ITERATIVE METHODS FOR STRUCTURED ALGORITHMIC DATA SCIENCE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Kevin Tian
June 2022

This dissertation is online at: https://purl.stanford.edu/qy752rq1707

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Aaron Sidford, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Nima Ahmadipouranari

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Moses Charikar

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Gregory Valiant

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Preface

This thesis studies the interplay between two aspects of algorithmic data science: iterative method theory and structured problem instances arising from applications. We organize the thesis into two parts, each of which couples a branch of the modern iterative method toolkit with a family of related structured problems in data science. Each part of the thesis develops new frameworks for designing iterative methods, gives new tools for the efficient implementation of said methods, and offers fresh perspectives on the relationships between our new iterative methods and various fundamental algorithmic problems in the relevant application domain.

Stochastic Variational Inequalities and Combinatorial Optimization. The first part of the thesis presents new algorithms and analysis frameworks for solving variational inequalities (VIs) in monotone operators (which generalizes convex optimization and equilbrium computation in minimax optimization), with a focus on algorithms for operators admitting a naturally stochastic structure.

In Chapters 2 and 3, we develop different aspects of stochastic VI theory: *acceleration* and *variance reduction*. In the case of Chapter 2, as a consequence of our new acceleration frameworks, we develop state-of-the-art algorithms for (well-conditioned) separable minimax and finite-sum optimization problems. In the case of Chapter 3, we give a host of new runtime tradeoffs for bilinear minimax optimization problems, advancing the frontier of approximate equilibrium computation in zero-sum matrix games for the first time since seminal works of [253, 415].

In Chapters 4, 5, and 6, we build upon the techniques of Chapters 2 and 3 by giving new algorithms for combinatorial optimization problems such as flow, matching, and transportation under different computational models. We achieve these results by combining advances in primal-dual acceleration, stochastic estimator design, and novel ways of implementing iterations of our methods in parallel, streaming, or dynamic settings. In this portion of the thesis, we give a state-of-the-art approximate solver for undirected maximum flow (Chapter 4), an implementation of low-pass, optimal space combinatorial optimization algorithms in a semi-streaming model (Chapter 5), and new reduction frameworks and faster update times for dynamic matching problems (Chapter 6).

Semidefinite Programming and High-Dimensional Statistics. The second part of the thesis investigates different aspects of the design of nearly-linear time semidefinite programming (SDP) solvers. In Chapter 7, we use ideas from the sketching literature and new theory for monotone SDPs to give a suite of new SDP tools catering to different geometries (e.g. Ky Fan or Schatten norm objectives) and constraint structure (e.g. packing or matrix dictionary recovery instances).

In Chapters 8 and 9, we showcase the power of our new SDP tools by providing robust algorithms for high-dimensional statistical estimation tasks under different contamination models. We consider contamination models capturing adversarial data poisoning and total variation distance misspecification in Chapter 8, where we give new almost-linear time algorithms for heavy-tailed clustering, generalized linear regression, and principal component analysis. These algorithms obtain near-optimal recovery guarantees in high dimensions as a function of the corruption parameter, and are based on combining the filtering paradigm with new SDP regret minimization tools. In Chapter 9, we robustify standard linear system solvers and regression algorithms in both overcomplete and undercomplete (sparse recovery) settings to run in nearly-linear time under the presence of an appropriate semi-random adversary which augments the measurement matrix (arbitrarily hindering standard notions of problem conditioning), without sacrificing statistical performance.

In Chapter 10, we consider a somewhat separate area of algorithmic high-dimensional statistics: developing an oracle complexity theory for structured logconcave sampling. Here too, we draw inspiration from aspects of iterative method theory such as proximal point methods and variance reduction to give a new framework for structured sampler design. We obtain state-of-the-art query complexities for well-conditioned, composite, and finite-sum structured densities; in the first case, we complement our new algorithms with near-matching lower bounds for popular frameworks.

Acknowledgments

If the reader chooses to only read one section of this thesis, I sincerely hope it is this one. This thesis would not have been possible if it were not for the joint efforts of a number of mentors, compatriots, and friends I have had the tremendous fortune to cross paths with over my lifetime. It is difficult to capture in words the influence they have had on both my academic and personal trajectories, but I will do my best in this section, and any omissions are solely my responsibility.

Mentors. I would like to start by thanking Aaron Sidford for being an amazing mentor, advisor, advocate and friend to me over the course of my Ph.D. Aaron took a leap of faith by accepting me as his student when I had never done research in algorithms or optimization before coming to Stanford. It took over a year of collaborating with Aaron before we got our first result, but Aaron continued to bring a level of enthusiasm and optimism to every meeting which set the tone for the whole group, and for how I strived to conduct theory research. Aaron taught me the value of dreaming big but also questioning our understanding of the basics at every stage. One of the biggest takeaways I have from working with Aaron is that every time I learn a new technique, it is a worthwhile endeavor (time permitting) to revisit every related problem armed with this new perspective to see if it yields deeper understanding. For all the lessons in both math and life you have taught me, and all the exciting research that has come out of our collaborations, thank you Aaron!

Next, I would like to thank Yin Tat Lee and Jerry Li, who have answered enough of my silly questions over Slack and Hangouts that they could be considered my informal co-advisors. I was fortunate to meet both Yin Tat and Jerry during Summer 2019 when I visited Seattle, and my tastes have substantially benefitted from working with both of them: almost all of the research in Part II of the thesis is a direct reflection of my work with Yin Tat (my guide into the world of sampling) and Jerry (likewise, for robust statistics). I am excited to continue our collaborations next year as I move to Seattle. Beyond both being incredibly sharp technically, Yin Tat and Jerry have given me very helpful advice during a number of tough times. Thank you, Yin Tat and Jerry!

Going back a bit further, I would like to thank James Zou and Greg Valiant for serving as my rotation mentors during my first few months at Stanford. Even though our research interests have diverged somewhat since then, they taught me about what it meant to do world-class research, and

have continued to provide guidance — I look up to both of them a lot.

Next, I thank my mentors from my time as an undergraduate, including Yuchun Guo, David Gifford and Dana Moshkovitz at MIT, and Jennifer Listgarten and Nicolo Fusi at Microsoft Research. I did quite a bit of flailing around trying to understand what kind of research I liked and how to make progress in general during those days — thank you all for your patience and mentorship. Yuchun in particular has been a great friend and advocate, and paved the way for my graduate studies.

I would like to thank Jon Kelner for hosting me at MIT during Summer 2021. Jon is one of the kindest academics I know, and taught me most of what I know about random matrix theory. I would also like to thank the other more senior academics I have benefitted from working with for all they have shared with me, including Tselil Schramm, Nima Anari, Sepehr Assadi, Chris Musco, Sivakanth Gopi, Adam Bouland, John Duchi, Ilias Diakonikolas, Daniel Kane, and Santosh Vempala.

The Stanford Theory Group is a wonderful place to spend 6 years thinking about hard problems, and part of what made me want to do TCS research in the first place was the universal friendliness I felt from theory-minded folk. I thank Moses Charikar, Li-Yang Tan, and Jan Vondrak for excellent courses they have taught, the other Stanford Theory faculty for creating an awesome environment, and Jay Subramanian, Megan Harris, Ruth Harris, and all the other Stanford CS administrators who have made the Ph.D. a smooth experience. I also have benefitted from Aaron's Stanford MS&E and ICME affiliations, and would like to thank the administration of these departments as well.

Finally, it is time to revisit my roots as an "academic" and thank all the teachers who shaped my tastes in math in the early years. I want to give special shoutouts to Mrs. Sarah Griffin and Mrs. Sally Barber, who helped me mature as a mathematician and a student, and have checked in over the years to make sure I am happy in life. I also want to thank Dr. Weizhen Gu for being one of my first extracurricular math teachers, and Sam Baethge, Dr. Max Warshauer, Dr. Nathaniel Dean, and Dr. Edward Early for letting me take my first steps into research at Texas Mathworks.

Academic compatriots. Theoretical research (and research of any form) can be an isolating experience if one is not careful, given the amount of time spent "in the trenches." I have been lucky to work with an amazing group of fellow students, each of whom has been incredibly helpful to talk to over the years as we jointly seek a deeper understanding. The Ph.D. would have been a lot less enjoyable without all of them, so I will use this space to shout them out.

Arun Jambulapati was one of the first students I worked closely with, and since then we have usually not gone very long before getting interested in a new direction together, having roped each other into a number of projects over the years. Arun has an incredible breadth and depth of knowledge, and our wide-ranging conversations have provided me stronger intuition for many topics. It's been a lot of fun, Arun, and I'm sure this is only the beginning.

Weihao Kong was the first senior student I worked on a project with, and has been very generous with his advice and friendship since then. Yair Carmon has a deep understanding of many different aspects of optimization theory, and was kind enough to include me on our first project [108] together.

He also has taught me a lot about writing style, and it never feels like you can be stuck on a problem for too long when you work with Yair. Those early days during the Ph.D. are the most important towards building a research network, and I am very grateful it started with Weihao and Yair.

I have been very fortunate to work closely with two junior students, Yujia Jin and Ruoqi Shen, on several projects throughout my Ph.D. Yujia and Ruoqi are both rising stars in the field, and I am thankful to them for all their efforts and insights, which have gotten us far.

Next, I would like to thank Swati Padmanabhan and Ewin Tang, close friends from my time in Seattle. Swati went through the exhilarating roller coaster that was the project [285, 286] with me and Arun; although our work there is not done yet, I'm sure we'll understand someday. Ewin has fielded many of my questions as I try to learn about quantum computing (which resulted in my first brief cameo on Wikipedia), and I also want to shout out her music recommendations.

I thank my other student collaborators over the years for our joint research efforts, including Teng Zhang, Qijia Jiang, Daniel Kongsgaard, Allen Liu, June Vuong, Sinho Chewi, and Daogao Liu. All are exceptional researchers and friends, and it has been a pleasure learning from them.

I also want to thank some of my other friends in graduate school, who helped make the academic grind a much more enjoyable process, including Shivam Garg (my gym buddy for many years), Kiran Shiragur, Yang Liu, Jay Mardia, Mark Sellke, Isabella Huang, Judy Shen, and Sitan Chen.

From the before times. The start of my Ph.D. was a very lonely experience at times, and the anxiety caused from the experience of dramatically switching areas during my first year would have been difficult to bear alone. Although it has become a much smoother process since then, when things get tough I still feel very fortunate to be able to lean on my friends "from the before times."

Jeffrey Chan has stuck with me as a "big brother" and role model for many different stages of my life. He edited all of my grad school and fellowship applications early on, as well as my first paper submission, hosted me in Berkeley on numerous occasions, and has provided an uncountable amount of advice about navigating the Ph.D. and adult life in general. Without his guidance and friendship, I would have surely been much more lost in this journey.

George Qi has been a stalwart friend to me since we were kids, and we have similarly stuck together through each stage of life since then. I am very grateful that we got to live together during the COVID times, though I am less grateful for all the ping pong balls he made me chase. Thanks for all the great memories, and the Rudy's, George.

Kevin Li and Zach Izzo were an absolute blast to live with during the Ph.D. and made Stanford feel like home away from home, especially during the long nights before deadlines. I am very thankful to have known them for so long (Kevin has been my roommate for periods during middle school, high school, college, and graduate school), and that we went through the Ph.D. journey together.

Moving to the Bay Area was an experience made much more comfortable by the presence of great friends nearby. For that, I thank Thom Lu, Jodie Chen, Connie Liu, Mike Wu, Jeff Sun, Kevin Peng, Lisa Huang, Serena Wang, Leonel Drukker, Rue Park, Joel Schneider, and Ben Bell.

I took multiple "academic vacations" to Seattle during the Ph.D. and those times were made even better by being able to hang out with Julia Guo, Matt Basile, Maddie Zhang, and Emmanuel Azuh. I'm really excited to move there and see you all again soon.

I would also like to thank Alex Jaffe, Kai Xiao, Weilian Chu, Tony Zeng, Eric Wang, Sonya Han, David Wang, Alex Hong, and Sameer Deshpande for being lifelong friends from college who I have kept in touch with over the years, even though I don't see them nearly as much as I'd like these days. Similarly, Ding Zhou, Lilly Shen, Leon Otis, Josh Dong, Danny Chen, Frances Chen, James Cong, Jackie Chen, Aaron Hui, Thaonhi Cung, Mary Jiang, David Yu, Priyanka Deshpande, Bobby Shen, Aditya Jain, Daniel (Shenghao) Wang, Jason Pang, Peter Hong, Ricky Chiang, Jessica Xiao, Jarry Xiao, Zeyi Lin, Prerna Bhat, Divya Ramamoorthy, Patrick Haley, and Connie Yan have been friends from my childhood who have kept in touch with me throughout all this time, which I deeply appreciate. During the COVID times, life was always more colorful when Lisa Hsiao and Kelly Zhou dropped by to visit us in Austin. Catching up with you all is always time well spent.

Finally, I thank the 1997-2016 San Antonio Spurs for giving me something to look forward to every year of my childhood. Pop, Tim, and Manu remain some of my biggest heroes in life.

The most important part. This is the most difficult part of the acknowledgments section for me to write, because I feel I will inevitably be unable to do it justice. I want to dedicate this thesis to all of the following amazing individuals, who have shaped me into the person I am.

From the very beginning, my parents Tian Hong and Xu Xiaoli have done everything they can to ensure that I live a good life. They sacrificed so much (more than I know) to give my sister and me a solid education and a happy childhood, putting our well-being first always. They taught me the value of honest, hard work, but also to always be proud of my efforts, rather than losing sleep over things outside of my control. They have been my biggest cheerleaders throughout my life, and I hope they know how much I look up to them as role models. Although the COVID times were difficult in some ways, I am grateful to have spent a year of my Ph.D. eating my parents' home-cooked meals every weekend. It is my great fortune of a lifetime to be able to call you my parents.

Sunny, thank you for all the fun memories we had growing up, and for being the most thoughtful sister I could ever ask for. Even though we're on different coasts now, it's always like a breath of fresh air when we get to see each other again. Despite being the older sibling, I feel like I still learn so much from you. I want you to know how proud I am to be your brother.

I would also like to acknowledge my extended family, especially my grandparents, Li Guirong, Xu Jintang, Zhang Fulan, and Tian Qingzhong. My maternal grandparents helped raise me during my early childhood, and were some of the most dedicated teachers I have ever had. My grandma Li Guirong patiently taught me times tables over long walks to and from elementary school, and my grandpa Xu Jintang painstakingly hand-translated math contests from English to Chinese so that he could teach me. Despite speaking no English, they spent over a decade in America taking care of my sister and me. I can only hope to give back to others a fraction of what they have given to me.

Amy, the most important event of my time in graduate school is when you came into my life. During the happy times, there is no one else I would rather celebrate with; during the hard times, you always know how to make me feel like everything will be alright. Long distance hasn't always been easy, but I could imagine no better partner through it all. You are my best friend, and life is sweeter now because I can always look forward to the next great adventure with you.

"When nothing seems to help, I go and look at a stonecutter hammering away at his rock perhaps a hundred times without as much as a crack showing in it. Yet at the hundred and first blow it will split in two, and I know that it was not that blow that did it, but all that had gone before."

— Jacob Riis

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Iterative methods are, in a broad sense, algorithmic procedures which make progress towards solving a computational task by performing a sequence of update rules, which are potentially much less cumbersome to implement than an "all-at-once" solution to the original problem. The analysis of iterative methods typically proceeds by first defining a global progress measure (a "potential"), and showing that local iterative updates advance this potential. In some cases, the potential used for analysis purposes directly reflects the problem the algorithm designer is trying to solve (e.g. when minimizing a loss function, the potential may be the function value itself). In other cases, e.g. when the most direct potential of interest either is difficult to directly measure or does not accurately reflect the type of progress being made by local iterates, designing an appropriate potential for analysis purposes can become a much more challenging endeavor.

Simple iterative methods such as (stochastic) gradient descent have emerged as a powerful workhorse enabling many recent advances in data science, operations research, and machine learning. The practical appeal of iterative methods is clear: by synthesizing local update rules into an algorithm with global performance guarantees, iterative methods effectively decouple the efficient implementation of the local update (such as matrix-vector multiplications, gradient computations, or other vectorizable operations) from the analysis of the overall algorithm. From an algorithm design perspective, what remains is to establish an understanding of performance measures obtained by the potpourri of iterative methods which practitioners may choose between. However, in many situations, there is not a "one-size-fits-all" type of analysis which generically applies to the myriad use cases of an iterative method framework. To advance the algorithmic frontier for a specific task at hand, an algorithm designer needs to take into account *specific structural aspects* of the problem in choosing an iterative method (and conducting a corresponding analysis, e.g. designing a potential).

Ubiquitous algorithms and algorithmic primitives building upon iterative update rules, such as stochastic gradient methods, convex programming, regression, and clustering, have well-understood

analyses in classical "textbook" regimes, under simplifying assumptions such as moderate dimensionality, knowledge of problem parameters, and an independently-sampled dataset. Frequently, these analyses are applied black-box in designing methodology for specific applications to obtain theoretical performance guarantees. However, in prominent modern applications going beyond these well-studied regimes, the theoretical guarantees of basic algorithms and their classical analyses may leave much on the table. This is particularly true when pursuing tradeoffs between goals such as robustness, accuracy, and computational efficiency as is often demanded by the practice of data-driven applications. This state of affairs suggests the following directions towards developing a modern theory of iterative methods.

First, can we develop a principled understanding of the power of classical frameworks for designing iterative methods, under design constraints imposed by the modern practice of data science?

Moreover, in cases when classical frameworks fall short, can we expand the modern computational theory of data science by proposing new algorithmic frameworks which provably address the shortcomings of the existing toolkit in the context of these aforementioned design constraints?

In many of the case studies contained in this thesis, the answer to these questions is yes, with an important qualification: to go beyond the black-box performance guarantees of analysis frameworks for iterative methods, we will crucially rely on a deeper understanding of *specific structural properties* of the algorithmic problem we study. This type of problem-specific structural understanding can be viewed as a guiding force in the theory of algorithm design in two ways. Most directly, it guides our selection of an appropriate iterative method and analysis for solving our desired problem. Taking a more comprehensive view, if the type of structure in question is sufficiently fundamental (e.g. shared across multiple natural computational problems), it may motivate the development of new iterative methods which are capable of taking advantage of said structure, and which thus have important implications for solving future problems with similar properties.

The goal of this thesis is to investigate algorithmic problems at the interface of iterative method design and the practice of modern data science through the lens of these two questions. Admittedly, the problems this thesis studies are fairly diverse, and at first glance may not seem particularly cohesive. However, the design of algorithms for solving these problems becomes a substantially more unified undertaking when approached through the principles discussed thus far, namely furthering the interplay between *iterative method theory* and *structured data science*. We organize the thesis into two distinct parts, which take different perspectives on the types of iterative method frameworks they investigate, the structure and mathematical properties of the problems they capitalize on, and the analytical toolkits they borrow from in synthesizing complete algorithms.

Each part of this thesis will begin with our developments of an aspect of modern iterative method theory: stochastic variational inequality solvers in Part I, and efficient semidefinite program solvers in Part II. In each case, the theory we develop aims to answer the following questions for the family of structured optimization problems at hand.

1. *Convergence analysis.* How can we develop analysis frameworks which effectively capture the convergence of our iterative methods (e.g. by choosing an appropriate potential function)?

2. *Efficient iterations.* How can we couple our framework with cheaply-implementable iterations (e.g. by replacing complex updates with simpler or more flexible alternatives, which afford a provably comparable iteration quality in terms of relevant progress measures)?

In each of Parts I and II, we will offer fresh perspectives for both of Questions 1 and 2: in the latter case of answering Question 2, we will frequently combine our developments in optimization theory with advancements in numerical linear algebra (i.e. streaming and sketching) or data structures.

Finally, the latter portion of each part of the thesis develops a close interplay between the new aspects of optimization theory we build and algorithms with improved performance guarantees for a fundamental application domain: combinatorial optimization in Part I, and high-dimensional statistics in Part II. By pushing forth this interplay with new iterative methods which are capable of taking advantage of structural aspects of the relevant application domain, our goal in this thesis is to showcase the power and flexibility of application-mindful iterative method design for solving problems in structured algorithmic data science.

Part I: Stochastic Variational Inequalities and Combinatorial Optimization

In the first part of this thesis, we develop new iterative methods for solving variational inequalities admitting stochastic formulations, as well as refined frameworks for analyzing these new methods. In Chapters 2 and 3, we directly apply these new methods to answer several open questions in the theory of first-order optimization methods. In Chapters 4, 5, and 6, we improve the state-of-the-art for a variety of well-studied problems in combinatorial optimization, such as maximum flow, optimal transport, and bipartite matching in different computational models, by combining our new iterative machinery with insights from the theoretical computer science literature, such as combinatorial constructions and efficient data structures.

We now briefly motivate how these developments came to be. Our approach to solving the algorithmic graph theory problems we study in Chapters 4, 5, and 6 is to first relax the underlying combinatorial optimization problem to an appropriate continuous formulation. The resulting continuous formulations share the following properties.

1. They are naturally formulated as zero-sum games between two players with geometrically-constrained domains (e.g. a Euclidean ball, a probability simplex, or a coordinatewise box).

2. They have combinatorial structure which can be harnessed to obtain tighter runtime characterizations, and are effectively captured by subsampling schemes (e.g. they are regression problems in an adjacency matrix with rows that are sparse or bounded, or become such a problem after applying a graph-structured preconditioner).

In order to capitalize upon these structural aspects, we develop the theory of stochastic variational inequalities. Variational inequalities (VIs) are a well-studied family of continuous optimization problems parameterized by a vector-valued operator $g : \mathcal{Z} \to \mathcal{Z}^*$, where $\mathcal{Z}^*$ is the dual space of $\mathcal{Z}$, see e.g. [458, 460, 459]. A *strong solution* to a VI in an operator g is a point $z^\star \in \mathcal{Z}$ such that

$$\langle g(z^\star), z^\star - z \rangle \leq 0 \text{ for all } z \in \mathcal{Z}. \tag{1.1}$$

The attentive reader well-versed in convex optimization may recognize (1.1) as the *first-order optimality condition* [458, 98]; indeed, when $g = \nabla f$ for a differentiable convex function f defined over a domain $\mathcal{Z}$, (1.1) is a necessary and sufficient condition for $z^\star$ to minimize f. This phenomenon is not unique to convex optimization: (1.1) has important implications for more general VIs, particularly when the operator g is *monotone* (for all $z, w \in \mathcal{Z}$, $\langle g(z) - g(w), z - w \rangle \geq 0$). Indeed, convex function gradients are monotone, and further another important example of a monotone operator is an appropriate gradient operator of a convex-concave function $f : \mathcal{Z} \to \mathbb{R}$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ is a product domain.[1] In this case, (1.1) implies that $z^\star$ is an equilibrium of f, and hence solves an appropriate minimax optimization problem.

In the special case of convex optimization, a substantial amount of effort in the continuous optimization community has been devoted to the understanding of *optimal query complexities* for solving (1.1) via first-order iterative methods, which in each iteration, query g at a point in the domain and perform a computationally-cheap update rule (e.g. direct vector addition). This has led to a comprehensive understanding of the complexity of solving convex optimization problems in various structured forms. A particularly important type of structure in the practice of optimization is *stochasticity*. Informally, this property is captured when g admits a natural formulation as an expectation over a random operator $\tilde{g} : \mathcal{Z} \to \mathcal{Z}^*$, and either updates or queries to $\tilde{g}$ are substantially cheaper than the analogs in g. A canonical example is when f is an average loss function over a sample or a population, and $\tilde{g} = \nabla f_i$ is the loss gradient evaluated at a specific individual i.

However, much less is known about the complexity of solving *stochastic variational inequalities beyond convex minimization*. This becomes highly relevant in our applications to combinatorial optimization, which are often encapsulated by an appropriate stochastic minimax problem, as alluded to by the structural properties discussed previously. Correspondingly, in Chapters 2 and 3, we develop new iterative methods for analyzing and solving more general stochastic variational inequalities. More concretely, Chapter 2 investigates the power of primal-dual formulations of variational inequalities for designing *accelerated algorithms*, making progress towards understanding Question 1 from the perspective of designing analysis frameworks for acceleration. Further, Chapter 3 builds theories of *variance reduction* and *numerical sparsity* for stochastic minimax problems, providing new tools for answering Question 2 for a suite of structured sparse updates.

[1] More precisely, in the case when $f(\cdot, y)$ is convex and $f(x, \cdot)$ is concave for any $x \in \mathcal{X}, y \in \mathcal{Y}$, we will define the monotone operator $g(x, y) = (\nabla_x f(x, y), -\nabla_y f(x, y))$.

Beyond yielding important primitives and design frameworks which will later be used in Chapters 4, 5, and 6, we note that results in Chapters 2 and 3 represent independently important progress in optimization theory, answering several basic open questions some of which have seen no improved bounds in over a decade (e.g. approximating a Nash equilbrium in a zero-sum game). More broadly, both minimax and stochastic optimization problems have become increasingly important for modeling purposes in modern data science, representing robustness to an adversary and scalability to large datasets respectively. As a result, we are optimistic that developing a general understanding of stochastic VI theory will have important downstream implications, a major motivation for the work in Part I of this thesis beyond specific applications in combinatorial optimization.

Finally, we briefly discuss the types of combinatorial optimization problems which are considered in Part I of the thesis, and the modern algorithmic design considerations they aim to address. Our results in Chapter 4, 5, and 6 showcase the interplay between iterative method design and combinatorial problem structure, as well as the flexibility of the stochastic VI solvers we build in adapting to various design constraints.

1. In Chapter 4, we consider the undirected maximum flow problem on capacitated graphs. We develop stochastic algorithms based on coordinate subsampling which, on a graph with m edges and n vertices, yield a flow of quality $1 - \epsilon$ times the optimum with a runtime[2] of $\widetilde{O}(m + \sqrt{mn}\epsilon^{-1})$, which remains the state-of-the-art for moderately small ϵ (and is improved further for graphs with sparse maximum flows). Our methods take advantage of inherent sparsity structure in graphs: by routing flows on a size-m graph via a smaller object (of dimension $\widetilde{O}(n)$), after a small number of passes over the graph, the method has a convergence rate which scales sublinearly in graph size, an appealing property for large-scale applications.

2. In Chapter 5, we consider combinatorial optimization problems such as transportation and matching in a *semi-streaming setting*, where we can only access the description of the graph using read-only passes over its adjacency matrix. We improve the pass complexities of semi-streaming iterative methods by applying acceleration tools to an appropriate continuous relaxation of the optimization problem. Moreover, by opening up the recursions governing our iterative method iterates, we demonstrate that our new algorithms can be implemented using an optimal $O(n)$ space. To achieve this, we combine an implicit representation of our iterative method iterates with dynamic data structure tools for sparsifying solutions on the fly.

3. In Chapter 6, we consider the application of iterative methods to *dynamic* combinatorial optimization problems, where one wishes to maintain an (approximate) solution on a slowly-changing graph, without recomputing from scratch. We demonstrate a simple framework for a dynamic model of bipartite matching, which reduces the solution maintenance problem

[2]Throughout the thesis, for ease of presentation, $\widetilde{O}$ hides polylogarithmic factors in appropriate problem parameters; in specific sections, we will be more precise in quantifying the relevant paramters.

to solving a small number of regularized regression subproblems. Intuitively, regularization equips the subproblem solutions with robustness properties which persist in a dynamic model, even against strong (adversarial) models of how the problem changes over time. Our hope is that this insight opens the door towards further applications of the powerful continuous optimization toolkit for dynamic algorithmic questions.

Part II: Semidefinite Programming and High-Dimensional Statistics

In the second part of this thesis, we present novel solvers achieving nearly-linear runtimes for various structured semidefinite programs, and their further algorithmic implications. Semidefinite programming (SDP) is a powerful primitive in convex optimization, and generalizes linear programming (LP) to matrix-valued variables and constraints. Formally, semidefinite programs (SDPs) are concerned with optimization problems of the form

$$\min \langle \mathbf{C}, \mathbf{X} \rangle \text{ subject to } \mathbf{X} \in \mathbb{S}^d_{\geq 0}, \ \langle \mathbf{A}_i, \mathbf{X} \rangle \geq b_i \text{ for all } i \in [n], \tag{1.2}$$

where $\mathbf{C}$, $\mathbf{X}$, $\{\mathbf{A}_i\}_{i\in[n]}$ are real, symmetric $d \times d$ matrices and $\mathbb{S}^d_{\geq 0}$ is the $d \times d$ positive semidefinite cone. However, the flexibility of using SDP solvers to attack an algorithmic problem can come at a significant computational price. The implementation of update rules for semidefinite programming solvers is notoriously expensive, often requiring full-matrix spectral operations (e.g. inversion or eigendecomposition), or careful analysis of the error propagation of approximate operations (see e.g. [399] for an extensive discussion of the difficulty of implicitly exponentiating a matrix). This computational overhead is a substantial challenge to address when deciding whether to use a SDP solver in practice. In Chapter 7, we give a suite of new SDP tools which both run in nearly-linear time, and effectively capture different types of problem structure. Two important types of structure we investigate are *monotonicity* (e.g. where the constraints $\{\mathbf{A}_i\}_{i\in[n]}$ in (1.2) are all nonnegative or nonpositive when viewed in an operator sense) and *geometry* (e.g. variants of (1.2) which have constraint geometry captured by an appropriate matrix norm, such as Schatten or Ky Fan norms, and hence can be phrased as regret minimization over an appropriate dual matrix set).

The solvers we give in Chapter 7 advance our understanding of Questions 1 and 2 on multiple fronts. From the perspective of Question 1, we suggest a new approach towards understanding *width-independent* convergence rates for monotone SDPs based on opening up potentials used in regret minimization, which results in multiple novel solvers. From the perspective of Question 2, we demonstrate new mathematical properties of low-rank sketches and the resulting approximate projections they implement, which allows for their use in iterates of our SDP solvers.

In Chapters 8 and 9, we build upon these new solvers either directly or indirectly (drawing inspiration from their design principles) to develop new methods for modern algorithmic questions in high-dimensional statistics. A common theme in these two chapters is the investigation of the

robustness of algorithms to model misspecification. Classical analyses of estimators and learning algorithms in statistics theory typically assume a "best-case" scenario from a modeling standpoint, which makes obtaining provable guarantees more tractable: a few examples follow.

1. Assume that the dataset is drawn independently from the same distribution (homogeneity).
2. Assume that the underlying distribution is "well-behaved" and "explicit" in various senses, such as belonging to a known family (e.g. Gaussian), exhibiting strong concentration properties (e.g. light tail behavior or bounded moments), or satisfying an explicit parameterization (e.g. being in approximately isotropic position, with a near-identity covariance).

However, due to the large-scale nature of modern data collection, imposing such strong assumptions about a statistical task may lead to meaningless algorithm performance guarantees, resulting in estimates which reveal very little about the actual problem at hand. As a result, it is important to develop algorithms which are flexible enough to datasets we are likely to actually observe, which may break typical assumptions. These assumption violations may occur at several different places along the data collection pipeline, including heterogeneity of sampling and (potentially adversarial) dataset contamination; alternatively, the ground truth may simply not have the properties we would like to posit (e.g. it is non-Gaussian, or satisfies weak concentration).

The field of robust statistics [31, 512, 278, 513] has classically addressed exactly these considerations, and indeed has resulted in robust estimators which can handle various forms of dataset contamination. Unfortunately, yet another aspect of modern data science throws a new wrinkle into the effectiveness of these classical robust estimators: *high dimensionality*. In particular, classical robust estimators when viewed as algorithms often result in runtimes exponential in the problem dimension. This has been alleviated in part by recent advances by the theoretical computer science community, e.g. [175, 337], but the resulting runtimes of many of these newly proposed algorithms are still far from linear in the problem size, and remain potentially prohibitive in high dimensions.

This thesis advances the state-of-the-art of robust algorithms for performing statistical tasks in high dimensions. We consider two fairly distinct examples of algorithmic robustness criteria "beyond the best case" motivated by practical modeling considerations.

1. In Chapter 8, we develop new algorithms for estimating generalized linear models, clustering mixture models, and performing principal component analysis under the *strong contamination model*. In this model, an adversary is allowed to replace a bounded fraction of (independent) draws with arbitrary points in the support of the underlying distribution, reflecting either (potentially adversarial) data poisoning or model misspecification in total variation distance.
2. In Chapter 9, we give new algorithms for overcomplete linear regression and sparse recovery, in the presence of a *monotone semi-random adversary*. Such an adversary can augment a well-behaved dataset with additional consistent information, which cannot affect the information-theoretic tractability of the problem, but can arbitrarily hinder the computational performance

of iterative methods which succeed under best case assumptions. This adversary captures the presence of heterogeneous data sources or the breaking of independence assumptions.

All of the algorithms presented in Chapters 8 and 9 crucially both run in *almost-linear time* in the dataset size, and further obtain *provably near-optimal statistical guarantees* under the relevant contamination model. Our algorithms build heavily upon the SDP solvers we develop in Chapter 7, to capture structural properties of the underlying problem instance. Intuitively, many of the niceness properties which are used by classical "best-case analyses" of the statistical problems we study are inherently *spectral*. For example, bounded moments may be certified through a small covariance matrix, and isotropicity and variants thereof are inherently statements about the conditioning of a dataset (how "spherical" the covariance matrix is). Moreover, algorithms which only take advantage of independence of samples through matrix concentration arguments can frequently be simulated through appropriate spectral certificates of concentration. As a result, even in contaminated situations beyond the best case, by carefully designing and solving appropriate SDP instances we can efficiently locate enough structure present in the problem to yield high-quality solutions.

Finally, in Chapter 10, we consider an area of algorithmic high-dimensional statistics which is somewhat of a departure from the other problems in Part II of this thesis: sampling from structured distributions under weak oracle access to the density. Specifically, the goal in Chapter 10 is to develop an *oracle complexity theory* for "logconcave sampling" problems of the following flavor:

$$\text{produce a sample from the density } \mu \propto \exp(-f), \text{ where } f : \mathbb{R}^d \to \mathbb{R} \text{ is convex.} \tag{1.3}$$

Problems of the form (1.3) frequently arise in computational Bayesian statistics (i.e. posterior sampling), as well as wide-ranging branches of modern data science such as differential privacy [63] and reinforcement learning [464]. In many applications, our access to μ is best modeled through a fairly implicit oracle, which queries gradients or values of the negative log-likelihood f at points (i.e. does not assume explicit knowledge of the density π or its normalizing constant). The goal of an oracle complexity theory is then to bound the number of oracle queries required to solve the problem (1.3).

Despite similarities between (1.3) and questions in convex optimization such as min f under value or gradient oracle access to f, which have been well-understood from an oracle complexity perspective for decades [414, 418], our understanding of the problem (1.3) from an analogous perspective is remarkably nascent. In Chapter 10, we push forward this research program by providing state-of-the-art samplers for basic structured formulations of (1.3) under standard oracle access and weak regularity assumptions, as well as lower bounds for well-studied approaches to sampler design.

Admittedly, the algorithms we develop for solving (1.3) are fairly distinct from the applications of SDP technology to robust statistical estimation found in the remainder of Part II of this thesis. However, the principles we use to design our new sampling frameworks draw heavy inspiration from the theory of iterative methods, and our analysis synthesizes tools from convex optimization

with structural insights from the extensive literature on high-dimensional statistics and applied probability. In this sense, Chapter 10 is very reflective of the themes of Part II at both a philosophical level, and also in the mathematical tools it uses. We hope the reader is understanding of the technical non sequitur the inclusion of Chapter 10 presents: the problem it studies is a personal favorite of the author, and we believe the resulting algorithms we design are sufficiently reflective of the goals of Part II that they merit representation in the thesis.

Organization

In Sections 1.1, 1.2, 1.3, and 1.4, we describe results contained in this thesis at a more detailed resolution. For brevity, we concentrate on defining the problems we solve and our results, providing only abbreviated discussions of motivations and techniques (deferring a more extensive discussion to the relevant chapters). Section 1.5 provides a bibliography on the content that this thesis represents.

1.1 Results: Theory of Stochastic Variational Inequalities

In Chapters 2 and 3, we give new algorithms for solving various fundamental stochastic VIs (1.1) in monotone operators. As a result, we obtain improved complexity bounds for several different optimization problems which admit appropriate stochastic VI formulations.

Throughout this discussion, fix a monotone operator $g : \mathcal{Z} \to \mathcal{Z}^*$, and a convex regularizer $r : \mathcal{Z} \to \mathbb{R}$. The classical proximal point method [459] is an implicit method for solving a VI by recursively performing the update

$$z_{t+1} \leftarrow \text{argmin}_{z \in \mathcal{Z}} \left\{ \langle \eta g(z), z - z_t \rangle + V_{z_t}^r(z) \right\}, \tag{1.4}$$

where $\eta > 0$ is a step size parameter, and V^r the *Bregman divergence* in r, a function which rewards proximity to z_t (e.g. when r is a standard squared Euclidean regularizer, V^r is simply the squared Euclidean distance). When r is strongly convex, it is well-known that (1.4) converges at a rate of roughly $\frac{1}{\eta T}$ in T iterations. However, the update (1.4) is considered implicit because it typically does not afford closed form solutions, and hence can only be approximately implemented. Indeed, as $\eta \to \infty$ solving (1.4) can be shown to solve the original VI in g as well. This framework hence presents a variety of tradeoffs an algorithm designer must consider.

1. How large should we choose the step size η to be able to solve the subproblems (1.4)? A large value of η leads to a stronger convergence rate, but more difficult subproblems to solve.

2. What optimization method should we use to implement solutions to each subproblem, and how can we take advantage of structure (e.g. stochastic formulations) present in g?

3. In cases when (1.4) may be difficult to directly implement, can we define an approximation to the update (1.4) whose analysis is more readily analyzed?

In Chapter 2, we provide a novel fine-grained analysis of the convergence rate of *extragradient methods*, a discretization strategy for simulating the subproblem (1.4) via fixed-point iteration [329, 415, 420] as suggested by Question 3. We then highlight the power of primal-dual formulations of optimization problems (which are well-captured by our new convergence criteria, termed "relative Lipschitzness"), and recover accelerated algorithms for smooth convex optimization and ℓ_∞ regression using classical extragradient methods. Finally, we demonstrate the flexibility of the relative Lipschitzness framework by adapting it to yield state-of-the-art accelerated algorithms for smooth finite-sum convex and separable minimax optimization problems.

In Chapter 3, in the spirit of Questions 1 and 2, we directly implement solvers for (1.4) for bilinear minimax problems over geometric domains (such as Euclidean balls and probability simplices) whose convergence rates are parameterized by η. Importantly, the resulting solvers obtain *sublinear* runtimes after a linear amount of preprocessing, and are based on a variance-reduced scheme for stochastic minimax optimization. By tuning the parameter η, the overall proximal point method yields improved rates for regression, classification, and equilbrium computation. We couple this new variance reduction framework for minimax optimization with gradient estimators based on subsampling *coordinates* of the constraint matrix, which results in significantly-improved runtimes for numerically sparse bilinear minimax problems.

1.1.1 Chapter 2: Acceleration via Primal-Dual Extragradient Methods

Extragradient methods are algorithms which approximately implement the update rule (1.4) by performing a linearized fixed-point iteration:

$$\begin{aligned} z_t' &\leftarrow \operatorname{argmin}_{z\in\mathcal{Z}} \left\{ \langle \eta g(z_t), z - z_t\rangle + V_{z_t}^r(z) \right\}, \\ z_{t+1} &\leftarrow \operatorname{argmin}_{z\in\mathcal{Z}} \left\{ \langle \eta g(z_t'), z - z_t\rangle + V_{z_t}^r(z) \right\}. \end{aligned} \tag{1.5}$$

For sufficiently simple (e.g. coordinatewise separable) regularizers r, the updates (1.5) are cheaply implementable. Classical analyses by [415, 420] show that when g is Lipschitz in a norm $\|\cdot\|$ and r is strongly convex in the same norm, the approximation (1.5) converges at a $\frac{1}{\eta T}$ rate for any $\eta \le L^{-1}$.

A direct application of the [415, 420] result to the case of smooth, strongly convex optimization (minimization of f where ∇f is L-Lipschitz and f is μ-strongly convex) yields a convergence rate of $\widetilde{O}(\frac{L}{\mu})$ gradient queries to a high-accuracy optimizer. Mysteriously, this convergence rate is *not* optimal: Nesterov's famous accelerated gradient descent algorithm [417] attains a convergence rate of $\widetilde{O}(\sqrt{\frac{L}{\mu}})$ for the same problem, which is the tight rate up to constants [418].

In Chapter 2, we first show that applying the rule (1.5) to smooth, strongly convex optimization does indeed yield the optimal rate of $\widetilde{O}(\sqrt{\frac{L}{\mu}})$, when one defines g as the gradient operator resulting

from a *primal-dual* minimax formulation of the optimization problem:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{X}^*}\frac{\mu}{2}\|x\|_2^2+\langle y,x\rangle-\tilde{f}^*(y), \tag{1.6}$$

where $\tilde{f}^*$ is the convex conjugate of $\tilde{f}:=f-\frac{\mu}{2}\|\cdot\|_2^2$. We give a tighter convergence guarantee for the updates (1.5) which goes beyond the standard Lipschitz g, strongly convex r setup considered in [415, 420]. We then show that running extragradient methods on the formulation (1.6) phrased as a variational inequality obtains an accelerated optimization rate through our tighter analysis.

Beyond recovering known results, we consider two families of modern optimization problems: separable minimax problems of the form

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}}f(x)+h(x,y)-g(y)\text{ where }f,g\text{ are convex and }h\text{ is convex-concave,} \tag{1.7}$$

and finite-sum convex optimization problems of the form

$$\min_{x\in\mathcal{X}}\frac{1}{n}\sum_{i\in[n]}f_i(x). \tag{1.8}$$

For each of (1.7) and (1.8), we design appropriate primal-dual formulations, and adaptations of (stochastic, in the latter case) extragradient methods which achieve accelerated rates of optimization. In the case of (1.7), we obtain the first algorithm to match a lower bound of [545] under smoothness and strong convexity. In the case of (1.8), we use our primal-dual formulation of (1.8) to design an accelerated method which improves upon the prior best rate obtained in [17] by up to a $\sqrt{n}$ factor.

1.1.2 Chapter 3: Stochastic Methods for Matrix Games

The theory of variance reduction for convex minimization has resulted in various improved algorithms for stochastic optimization problems [477, 472, 17]. The principle behind variance reduction is to use a small number of (more-expensive) full gradient computations as preprocessing to improve the quality of later stochastic gradient estimates. In particular, letting $\widetilde{\nabla}f$ be a (potentially poor) unbiased estimator for ∇f, the idea is to reduce its variance by using the estimate

$$\widetilde{\nabla}f(x)-\widetilde{\nabla}f(\bar{x})+\nabla f(\bar{x}),$$

which is similarly unbiased, but whose variance improves the closer $\bar{x}$ is to a minimizer of f.

However, for minimax problems, a similar theory of variance reduction was largely unexplored (except in the Euclidean regression setting: see [432]). This lack of development was an impediment, in some cases, to further improvements to the complexity of stochastic minimax optimization. To further the theory of minimax variance reduction, we consider in Chapter 3 the basic setting of

bilinear saddle point problems:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} y^\top \mathbf{A}x - b^\top y + c^\top x, \tag{1.9}$$

for some $\mathbf{A}\in\mathbb{R}^{m\times n}$ and geometrically-constrained domains $\mathcal{X}$, $\mathcal{Y}$. For example, in an ℓ_1-ℓ_1 "zero-sum game," $\mathcal{X}=\Delta^n$ and $\mathcal{Y}=\Delta^m$ are probability simplices with bounded ℓ_1 norm, which represent mixed strategies (we additionally consider analogous ℓ_2-ℓ_1 games, and ℓ_2-ℓ_2 games, which encapsulate linear classification and programming, and obtain similar improvements, but omit discussion here).

As a starting point, we develop a way of combining ideas from the theory of variance reduction with the tradeoffs afforded by proximal point methods (1.4), to obtain new algorithms with improved complexities for zero-sum games. Roughly speaking, when $\mathbf{A}$ is entrywise bounded, prior algorithms by [415] and [253] respectively obtained runtimes of

$$\widetilde{O}\left(\frac{\mathrm{nnz}(\mathbf{A})}{\epsilon}\right) \text{ and } \widetilde{O}\left(\frac{m+n}{\epsilon^2}\right)$$

for computing an ϵ-approximate equilbrium to (1.9), where $\mathbf{A}$ has $\mathrm{nnz}(\mathbf{A})$ nonzero entries. We instead show that we can solve the subproblem (1.4) at a rate of $\widetilde{O}((m+n)\eta^2)$, after a $\mathrm{nnz}(\mathbf{A})$-time preprocessing step, for any $\eta<\epsilon^{-1}$. Altogether, choosing the best η yields a rate of

$$\widetilde{O}\left(\left(\mathrm{nnz}(\mathbf{A})+(m+n)\eta^2\right)\cdot\frac{1}{\eta\epsilon}\right) \implies \widetilde{O}\left(\mathrm{nnz}(\mathbf{A})+\frac{\sqrt{\mathrm{nnz}(\mathbf{A})(m+n)}}{\epsilon}\right),$$

which generically improves upon [415] and improves [253] for moderately small ϵ. We give analogous speedups for *nonlinear* stochastic minimax problems under a gradient oracle model.

By designing new stochastic gradient estimators based on subsampling coordinates, we show how to further improve upon this rate for *numerically sparse* $\mathbf{A}$, namely dense matrices with few large entries. Intuitively, an estimator which samples coordinates avoids density issues by targeting small entries of $\mathbf{A}$ less frequently, and hence presents a finer-grained, more continuous characterization of $\mathbf{A}$'s inherent sparsity. For example, when every row and column of $\mathbf{A}$ has Euclidean norm $\widetilde{O}(1)$, we obtain an improved runtime of

$$\widetilde{O}\left(\left(\mathrm{nnz}(\mathbf{A})+\eta^2\right)\cdot\frac{1}{\eta\epsilon}\right) \implies \widetilde{O}\left(\mathrm{nnz}(\mathbf{A})+\frac{\sqrt{\mathrm{nnz}(\mathbf{A})}}{\epsilon}\right).$$

Direct uses of our stochastic methods for matrix games imply similar speedups for linear classification, linear regression, and various problems in computational geometry.

1.2 Results: Combinatorial Optimization

In Chapters 4, 5, and 6 we draw inspiration from our developments on stochastic VI theory, and obtain new algorithms for approximately solving constrained ℓ_1 and ℓ_∞ regression. The resulting methods have immediate implications for a variety of combinatorial optimization problems, which are reducible to regression. As an example, consider the *box-constrained ℓ_∞-regression problem*

$$\min_{x\in[-1,1]^m} \|\mathbf{A}x - b\|_\infty + c^\top x, \tag{1.10}$$

where $\mathbf{A} \in \mathbb{R}^{n\times m}$ has rows with bounded ℓ_1 norms. Solving (1.10) at an *accelerated rate*, i.e. computing an ϵ-approximate minimizer within a runtime scaling linearly or sublinearly in $\text{nnz}(\mathbf{A})$ but depending only linearly on ϵ^{-1}, was a notorious open problem in the continuous optimization community (it is straightforward to obtain a rate of $\text{nnz}(\mathbf{A}) \cdot \epsilon^{-2}$ using unaccelerated gradient methods). One concrete reason for this is lower bounds on the size of ℓ_∞-strongly convex regularizers over a box [483, 486], which typically govern the convergence rate of accelerated methods.

In a breakthrough work, Sherman in [483] showed how to formulate (1.10) as a "box-simplex game"

$$\min_{x\in[-1,1]^m} \max_{y\in\Delta^n} y^\top \mathbf{A}x - b^\top y + c^\top x \tag{1.11}$$

and design a somewhat ad hoc accelerated algorithm for solving (1.11) via a primal-dual extragradient method utilizing a *coupled* regularizer. Intuitively, Sherman's regularizer uses the dual variable y to perform a careful reweighting of a primal regularizer on x, bypassing the size lower bound.

To further our understanding of these local reweighting strategies and their implications for accelerating structured regression, we show in Chapter 2 how to construct a simpler primal-dual regularizer which retains Sherman's rate for solving (1.10), (1.11). Moreover, we give a flexible analysis strategy for our regression algorithm through the framework of a strengthening of our relative Lipschitzness criterion for extragradient algorithms, which we term *local relative Lipschitzness*.

In Chapter 4, we propose alternatives to Sherman's regularization strategy for solving (1.10) based on reweighted accelerated coordinate methods. The resulting runtimes improve upon [483] in many structured instances, and we show how to apply them to obtain faster approximate undirected maximum flow algorithms by using advances in combinatorial preconditioning [482, 318, 441].

In Chapter 5, we show that by recasting graph-structured problems such as bipartite matching, optimal transport, and transshipment as box-simplex games, we can obtain improved parallel complexities for approximately solving each by building upon our solver from Chapter 2. Moreover, we demonstrate that the iterates of our solver admit recursions which are amenable to implementation in low space in a semi-streaming model of access to the graph. As a result, we improve the state-of-the-art pass and/or space complexities for all of these problems as well.

In Chapter 6, we give a generic reduction framework from a decremental variant of bipartite

matching to solving a sequence of appropriately *regularized box-simplex games*, i.e. (1.11) with additional (small) regularization terms which encourage the simplex variable y to be "uniformly spread." We complete this picture by providing new accelerated solvers for said regularized games, via our relative Lipschitzness framework from Chapter 2.

1.2.1 Chapter 4: Faster Approximate ℓ_∞ Regression and Maximum Flow

The (undirected, capacitated) maximum flow problem on a graph $G = (V, E, w)$ (where $u \in \mathbb{R}^E_{\geq 0}$ are edge capacities) asks to find a flow $f \in \mathbb{R}^E$ which satisfies entrywise capacity constraints $|f| \leq u$, and routes as many units as possible from a source s to a sink t. Letting $\mathbf{B} \in \{-1, 0, 1\}^{E \times V}$ be the (signed) edge-vertex incidence matrix of G, and letting $d \in \mathbb{R}^V$ be a 2-sparse *demand vector* with $d_s = -1$, $d_t = 1$, it is straightforward to see that solving the "minimum congestion flow" problem

$$\min_{\mathbf{B}^\top f = d} \left\| \mathbf{diag}\,(u)^{-1} f \right\|_\infty$$

recovers the maximum flow. In [482, 318, 441] it was shown that an appropriate "soft relaxation" of the hard constraints $\mathbf{B}^\top f = d$ can be used without loss in approximation quality: it suffices to solve a box-constrained instance (1.10) instead. Intuitively, the box constraint on $x := \mathbf{diag}\,(u)^{-1} f$ follows from a reparameterization and binary search on the value of $\|x\|_\infty$, and $\mathbf{A}$ is $\mathbf{B}$ after left-multiplying by a *combinatorial preconditioner*, and has bounded row norms. The preconditioner is obtained by a routing scheme from G onto a smaller graph. Our runtime improvements for maximum flow follow by designing optimization procedures which capture the structure of the problem more tightly. In particular, our methods directly leverage sparsity properties of the preconditioner.

We begin by developing a faster algorithm for solving (1.10) via an accelerated coordinate descent algorithm with dynamic sampling probabilities, providing an alternative to [483]. The resulting algorithm yields a runtime of

$$\widetilde{O}\left(m + \frac{\sqrt{mn}}{\epsilon}\right)$$

for solving a maximum flow instance with n vertices and m edges, to $1 - \epsilon$ multiplicative accuracy, generically improving upon the $\widetilde{O}(m\epsilon^{-1})$ runtime of [483] for slightly-dense graphs. We further build upon a synthesis of stochastic VI acceleration methodology from Chapter 2, and data structures for supporting coordinate updates from Chapter 3, to obtain an improved maximum flow algorithm. When the optimal $x^\star$, the solution to (1.10), has $\|x^\star\|_2^2 = s$, our improvement runs in time

$$\widetilde{O}\left(m + \frac{n + \sqrt{ns}}{\epsilon}\right).$$

Since $s \leq m$ generically, this is never worse than our previous runtime (up to logarithmic factors), and represents an improvement for sparse maximum flows.

1.2.2 Chapter 5: Semi-Streaming Combinatorial Optimization

We demonstrate that various combinatorial optimization problems can be efficiently reduced to box-simplex games, following similar "soft penalization schemes" as were used in prior works on maximum flow. For example, consider the problem of *maximum cardinality bipartite matching* on an unweighted bipartite graph $G = (V, E)$, where we wish to find a flow vector $f \in \mathbb{R}^E_{\geq 0}$ which solves

$$\max_{\mathbf{B}^\top f \leq \mathbf{1}_V} \|f\|_1 . \tag{1.12}$$

Here, we overload notation from the prior section and let $\mathbf{B} \in \{0, 1\}^{E \times V}$ be the (unsigned) incidence matrix, and $\mathbf{1}_V$ is the all-ones vector on V. The condition $\mathbf{B}^\top f \leq \mathbf{1}_V$ means there is no more than one total unit of flow on any vertex, and $\|f\|_1$ is the matching size. It is well-known that an optimal f is *integral*, i.e. it is 0-1 valued, so this continuous relaxation is without loss.

By using the existence of combinatorial rounding schemes which greedily remove overflow (violations of the constraints $\mathbf{B}^\top f = 1$) without substantially affecting matching size, we show that the matching problem (1.12) is captured by an ℓ_1 regression problem

$$\min_{x \in \Delta^E} -\langle \mathbf{1}_E, x \rangle + \left\| \max(\mathbf{B}^\top x - \mathbf{1}_V, 0) \right\|_1 \tag{1.13}$$

which penalizes constraint violations directly in the objective. A primal-dual formulation of (1.13) can then be framed as a box-simplex game, and is hence amenable to the accelerated extragradient solvers developed in [483] or Chapter 2 of this thesis. Similar observations for the optimal transport and transshipment problems (e.g. shortest path) result in state-of-the-art parallel algorithms for each, obtaining depth scaling as $\widetilde{O}(\epsilon^{-1})$ where ϵ governs approximation quality, and near-linear work. Prior parallel complexities of near-linear work solvers for these problems scaled as $\widetilde{O}(\epsilon^{-2})$ [365, 358], and were based on unaccelerated methods. Our results follow because extragradient methods rely on only first-order ("gradient") information about the problem, which can be implemented using only efficiently parallelizable matrix-vector multiplications.

We next turn to solving these problems under a semi-streaming model of computation. In this model, the matrix $\mathbf{B}$ is not stored explicitly due to space considerations, and can only be accessed through read-only streams where edges are presented one at a time, termed *passes* over the graph. This presents a potential obstacle to solving (1.13) in low space: the iterates are naturally $|E|$-dimensional, and hence writing them is as expensive as storing the entire matrix $\mathbf{B}$, which we were trying to avoid! As our main result, we show our accelerated box-simplex game solver from Chapter 2 can entirely be implemented and deterministically sparsified in $O(|V|)$ space for these combinatorial applications, implying a $\widetilde{O}(\epsilon^{-1})$ pass complexity in a semi-streaming model and improving upon prior works on semi-streaming bipartite matching (e.g. [12, 13]) in either space, passes, or both.

1.2.3 Chapter 6: Dynamic Decremental Bipartite Matching

In recent years (cf. Chapters 4 and 5), a variety of *static* combinatorial optimization problems have benefitted tremendously from continuous perspectives and iterative method machinery. However, the development of *dynamic* graph algorithms, an important and widely-studied undertaking in the theoretical computer science literature, has remained fairly separate from continuous optimization developments. This begs the question: can we directly use faster solvers for continuous formulations of combinatorial problems to obtain improved complexities in a dynamic setting?

As a proof-of-concept, in Chapter 6 we show how to obtain such a reduction for a natural dynamic decremental model of bipartite matching (undergoing possibly adversarial edge deletions), abbreviated DDBM. We prove that to maintain a fractional matching with value at least $1 - \epsilon$ the optimum, it suffices to solve $\widetilde{O}(\epsilon^{-2})$ *regularized subproblems* roughly of the form

$$\min_{x \in \Delta^E} -\langle \mathbf{1}_E, x \rangle + \left\| \max(\mathbf{B}^\top x - \mathbf{1}_V, 0) \right\|_1 + \epsilon \sum_{e \in E} x_e \log x_e$$

to high accuracy. By directly plugging in high-precision accelerated solvers for regularized box-simplex games inspired by our extragradient analysis framework in Chapter 2, we deterministically solve the DDBM problem at an amoritized cost of $\widetilde{O}(\epsilon^{-3})$, over $m := |E|$ edge deletions. This improves upon the prior state-of-the-art of $\widetilde{O}(\epsilon^{-4})$ from [73] for the same problem.

Our framework is flexible enough that it decouples the DDBM problem from the cost of implementing subproblems, and hence is compatible with powerful techniques for solving variants of regularized box-simplex games. By using a recent breakthrough in the maximum flow literature [124] as our subproblem solver, we obtain an amoritized cost of $\widetilde{O}(m^{o(1)}\epsilon^{-2})$ over m edge deletions for the DDBM problem, an improvement upon our earlier result for sufficiently small ϵ.

1.3 Results: Structured Semidefinite Programming Solvers

In Chapter 7, we present nearly-linear time approximation algorithms for structured semidefinite programs (1.2), several of which are directly used in our applications in Chapters 8 and 9. Our algorithms are based on adaptations and generalizations of the *matrix multiplicative weights* framework, which we briefly overview here before stating our new results.

Matrix multiplicative weights (MMW) is an instantiation of regret minimization techniques over a subset of the positive semidefinite matrix cone, denoted $\mathbb{S}^d_{\geq 0}$ (where the matrices in question are $d \times d$). To give a typical example, let $\mathcal{X} := \{\mathbf{X} \in \mathbb{S}^d_{\geq 0} \mid \mathrm{Tr}(\mathbf{X}) = 1\}$ be the "spectraplex." Given a sequence of operator-norm bounded "gain matrices" $\{\mathbf{G}_t\}_{t \geq 1}$, MMW produces a sequence of response matrices $\{\mathbf{X}_t\}_{t \geq 1} \subset \mathcal{X}$ which aim to minimize the cumulative regret in linear measurements through

$\{\mathbf{G}_t\}_{t\geq 1}$ compared to the best action $\mathbf{U} \in \mathcal{X}$ in hindsight:

$$\sup_{\mathbf{U}\in\mathcal{X}} \left(\sum_{t\in[T]} \langle \mathbf{G}_t, \mathbf{U} \rangle \right) - \left(\sum_{t\in[T]} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \right). \tag{1.14}$$

Crucially, the response matrices $\{\mathbf{X}_t\}_{t\geq 1}$ must be chosen in an *online fashion*, without knowledge of future gain matrices $\mathbf{G}_s$ for $s > t$. When all gain matrices satisfy $\|\mathbf{G}_t\|_{\text{op}} \leq 1$, the matrix Hölder inequality implies that the regret can never be more than T, the number of iterations. The key guarantee of MMW is that it beats this trivial bound by obtaining *vanishing average regret*: it plays a sequence of response matrices such that the total regret (1.14) scales as $\widetilde{O}(\sqrt{T})$. A standard application of MMW, based on reducing (1.2) to feasibility problems which can be captured by regret minimization over $\mathcal{X}$, yields approximation algorithms for semidefinite programming: this reduction is detailed in Section 7.2. However, this situation comes with a few caveats.

1. Although the update rule employed by MMW is simple to state (based on solving entropy-regularized problems over $\mathcal{X}$, which admit explicit characterizations), actually implementing these updates exactly requires matrix eigendecomposition, which is an expensive operation. Are there cheaper alternatives which do not sacrifice performance?

2. There have been relatively few explorations of instantiating MMW for subsets of $\mathbb{S}^d_{\geq 0}$ beyond the spectraplex defined previously. Regret minimization over various other structured subsets of interest may entail more complicated update rules, and the resulting methods necessitate careful analyses when combined with faster, approximate implementations of said updates bypassing explicit eigendecomposition. Can we develop approximation-tolerant MMW variants over more complex sets?

3. The reduction of SDPs (1.14) to regret minimization over the spectraplex can be very lossy from the perspective of problem parameterization; typically, one needs to pay an overhead depending on the *problem width*, such as the size of constraints $\{\mathbf{A}_i\}_{i\in[n]}$ or the optimizer $\mathbf{X}^\star$. For structured SDPs, can we design more fine-grained reductions which bypass this lossiness via stronger guarantees (e.g. multiplicative value approximations with no dependence on width)?

In Chapter 7, we provide new variants of the MMW framework which make progress towards addressing each of these questions.

1.3.1 Chapter 7: Matrix Multiplicative Weights and Friends

We summarize the performance guarantees of our new variants of the MMW framework.

A rank-1 sketch for MMW. Standard MMW iterates can be expressed in closed form as $\mathbf{X}_t = \frac{\exp(\mathbf{Y}_t)}{\mathrm{Tr}\exp(\mathbf{Y}_t)}$, for a matrix $\mathbf{Y}_t$ efficiently computable from the sequence of $\{\mathbf{G}_s\}_{s\le t}$. However, computing $\mathbf{X}_t$ explicitly can be prohibitively expensive for applications. Towards addressing Question 1, one cheaper way to approximate $\mathbf{X}_t$ is to let $\mathbf{Q} \in \mathbb{R}^{k\times d}$ be a Johnson-Lindenstrauss (approximately norm-preserving) *sketch matrix*, and instead compute

$$\widetilde{\mathbf{X}}_t := \frac{\exp(\frac{1}{2}\mathbf{Y}_t)\mathbf{Q}^\top\mathbf{Q}\exp(\frac{1}{2}\mathbf{Y}_t)}{\langle\exp(\mathbf{Y}_t), \mathbf{Q}^\top\mathbf{Q}\rangle}.$$

Both the numerator and denominator of $\widetilde{\mathbf{X}}_t$ can be efficiently approximated using the Lanczos method [339] and polynomial approximations to the exponential. However, black-box applications of Johnson-Lindenstrauss constructions require choosing $k \approx \log d\cdot\epsilon^{-2}$ to guarantee an ϵ-approximation to inner products. We show that, perhaps surprisingly, when each gain matrix $\mathbf{G}_t$ is chosen obliviously to randomness used in computing the current iterate $\mathbf{X}_t$, a rank-1 sketch matrix suffices to retain standard MMW guarantees up to constant factors (letting $\mathbf{Q} = q^\top$ for a uniformly random unit vector q). As a result of our simple sketch, we improve upon prior state-of-the-art runtimes for regret minimization over the spectraplex by factors of $\Omega(\log^5 d)$ [20].

Ky Fan MMW. Let $k \in [d]$. Towards addressing Question 2, we develop new regret minimization procedures over the "k-Fantope,"

$$\mathcal{F}_k := \{\mathbf{X} \in \mathbb{S}^d_{\ge 0} \mid \mathrm{Tr}(\mathbf{X}) = k,\ \mathbf{X} \preceq \mathbf{I}\},$$

with iteration complexity dominated by an approximate k-PCA procedure, implementable in $\widetilde{O}(k)$ matrix-vector multiplications through gain matrices. The set $\mathcal{F}_k$ is important in applications because it is the set of dual certificates against the Ky Fan norm $\|\mathbf{G}\|_k$, defined to be the sum of the k largest eigenvalues (in magnitude) of $\mathbf{G}$: namely, for $\mathbf{G} \in \mathbb{S}^d_{\ge 0}$, $\|\mathbf{G}\|_k = \sup_{\mathbf{X}\in\mathcal{F}_k}\langle\mathbf{G}, \mathbf{X}\rangle$. When $k = 1$, our method recovers the standard MMW guarantee.

A key difficulty in designing a regret minimization procedure over $\mathcal{F}_k$ is the relatively complicated nature of entropic projections onto $\mathcal{F}_k$, which involves thresholding eigenvalues. As a side result required by our analysis, we give new performance guarantees on approximate k-PCA procedures such as simultaneous power iteration for approximating the spectrum of a matrix, refining results from [404, 136]. By opening up the interplay between our MMW variant and approximate spectrum computation, we obtain end-to-end approximation-tolerant algorithms for regret minimization over $\mathcal{F}_k$, and improve prior bounds in [136] for the same problem by factors of $\Omega(k^3)$.

Structured positive SDP. We consider variants of (1.2) with *monotonicity* structure, also referred to in the literature as mixed packing-covering (MPC) SDPs. The feasibility variant of MPC SDPs, in

full generality, asks to determine whether there exists $w \in \mathbb{R}^n_{\geq 0}$ such that

$$\sum_{i\in[n]} w_i \mathbf{P}_i \preceq \sum_{i\in[n]} w_i \mathbf{C}_i, \text{ for } \{\mathbf{P}_i\}_{i\in[n]}, \{\mathbf{C}_i\}_{i\in[n]} \subset \mathbb{S}^d_{\geq 0}. \tag{1.15}$$

We can think of $\{\mathbf{P}_i\}_{i\in[n]}$ as a set of matrices we wish to "pack" in a weighted sense, such that they are "covered" by the set of $\{\mathbf{C}_i\}_{i\in[n]}$. The monotone linear programming specialization of (1.15), where all of $\{\mathbf{P}_i\}_{i\in[n]}$, $\{\mathbf{C}_i\}_{i\in[n]}$ are diagonal matrices, admit approximation algorithms with very strong guarantees: they can test feasibility of (1.15) up to a $1+\epsilon$ multiplicative factor, depending polynomially on ϵ^{-1} and only *polylogarithmically* on width parameters of the problem. One compelling answer to Question 3 would thus be designing similar "width-independent" approximation algorithms for testing (1.15). However, a solution to this problem in full generality has resisted efforts from the theoretical computer science and continuous optimization communities, including a recent attempt by the author of this thesis [286].

Towards developing general solvers for (1.15), we give improved algorithms for various specializations of MPC SDPs in Chapter 7. We first consider the case when each $\mathbf{C}_i$ is a κ multiple of $\mathbf{P}_i$ for some $\kappa > 1$, which we term a "matrix dictionary recovery SDP." This specialization has immediate implications to structured preconditioning tasks (detailed in Section 9.2), and intuitively asks to approximate $\mathbf{I}$ with a linear combination of our matrix dictionary of $\{\mathbf{P}_i\}_{i\in[n]}$. We give a nearly-linear time algorithm for solving matrix dictionary SDPs with iteration complexity scaling only *linearly* in κ, the natural width parameter of the problem. We also design methods which generalize this result to approximating an arbitrary constraint matrix $\mathbf{B} \in \mathbb{S}^d_{\geq 0}$ using a specified matrix dictionary which admits efficient linear system solvers, e.g. graph Laplacians [491].

Finally, in the special case of (1.15) where all the $\mathbf{C}_i$ are scalars (also known as "pure packing SDPs"), we give an improved solver whose runtime roughly scales as ϵ^{-5}, improving the ϵ^{-6} dependency of [19, 442]. We obtain our improvement by interpreting width-independent solvers for pure packing SDPs through the lens of mirror descent, and show that this viewpoint extends straightforwardly to solve "Schatten norm packing SDPs" of the form

$$\max \|w\|_1 \text{ subject to } \left\| \sum_{i\in[n]} w_i \mathbf{P}_i \right\|_p \leq 1$$

at a width-independent rate, where $\|\cdot\|_p$ of a matrix argument is the Schatten-p norm. Our result is the first width-independent, nearly-linear time solver for Schatten norm packing SDPs.

1.4 Results: Algorithmic High-Dimensional Statistics

1.4.1 Chapter 8: High-Dimensional Robust Statistics in Almost-Linear Time

We design algorithms for three types of high-dimensional statistical estimation problems under *dataset contamination models*: clustering, generalized linear regression, and principal component analysis (PCA). Our contamination models are variants of the following: we observe a dataset $\{X_i\}_{i\in[n]}$ such that an $\alpha \in (0,1)$ fraction is independently drawn from a "ground truth" $\mathcal{D}$ supported on $\mathbb{R}^d$, and the remainder of the data is arbitrary (potentially adversarially chosen) points in $\mathbb{R}^d$.

Several algorithms in Chapter 8 are variants of *filtering*. Instantiations of the filtering paradigm in algorithmic robust statistics (originally introduced in [177, 360, 492]) combine two ideas: certifiable corruption and iterative dataset sanitization. The first idea asks to produce a (problem-specific) example of a *certificate of corruption*: in the absence of the certificate, a naïve solution (e.g. an empirical mean) should suffice for the estimation problem. Often, the proof of correctness of this certificate comes with a contrapositive which states that whenever it is present, it must be because of dataset contamination, and can hence be used as a "spectral signature" [508, 360] to define scores for each data point which are amplified on the adversarial points. For example, when performing mean estimation and the ground truth $\mathcal{D}$ is assumed to have a bounded covariance, an appropriate certificate is an amplified-variance direction v and the corresponding scores are correlations with v (relative to the empirical mean). These scores can then be used to iteratively sanitize the dataset, e.g. by downweighting a set of soft "inlier" scores being maintained over the data points.

Once an appropriate certificate is identified from statistical properties of the problem, basic instances of filtering strategies frequently suffice for obtaining polynomial-time algorithms for robust statistics. However, a much more challenging endeavor is developing robust estimators which both attain optimal statistical guarantees, and run in *nearly-optimal time*. Designing such a "best-of-both-words" estimator requires satisfactory answers to the following questions.

1. How can we compute scores through a certificate of corruption in nearly-linear time?
2. How can design a scheme which synthesizes iterations downweighting based on corruption certificates, and provably terminates in a near-constant number of iterations?

To answer these questions in the setting of robust mean estimation, [194] used an approximate top eigenvector computation to answer Question 1. Moreover, they developed a way of using the guarantees of *regret minimization procedures over the spectraplex* to ensure fast termination of the method. More specifically, [194] used the response matrices of MMW as their certificates of corruption, and used MMW regret guarantees to repeatedly decrease the operator norm of an empirical covariance matrix by a constant factor. Chapter 8 builds upon the program initiated by [194], and gives cheaply-computable robust estimators for multiple tasks beyond mean estimation.

Clustering heavy-tailed mixture models. Given a dataset of size n drawn from a *mixture model* $\mathcal{D} := \sum_{i\in[k]} \frac{1}{k}\mathcal{D}_i$, where each mixture component $\mathcal{D}_i$ is *Gaussian* and has identity-bounded covariance, we wish to recover which data point was drawn from which component to arbitrary constant precision (e.g. correctly classifying a .95 fraction of points). This problem is information-theoretically tractable assuming a sufficient separation condition between component means $\{\mu_i\}_{i\in[k]}$, and there exist efficient algorithms for solving it dominated by $\widetilde{O}(1)$ approximate k-PCA computations as long as $\min_{i\neq j\in[k]} \|\mu_i - \mu_j\|_2 = \Omega(k^{-\frac{1}{2}})$ [520, 7]. However, these algorithms break down in the face of contamination, do not generalize to heavy-tailed (non-Gaussian) data, and do not run in linear time (incurring a k-factor overhead).

We present a new clustering algorithm which bypasses all of these barriers assuming a similar separation level. In particular, we provide an almost-linear time algorithm (with runtime $O(n^{1+\epsilon_0}d)$ for any constant ϵ_0) for the following problem, termed *list-decodable mean estimation*. We are given a dataset $\{X_i\}_{i\in[n]}$, an $\alpha \in (0, \frac{1}{2})$ fraction of which is drawn from $\mathcal{D}$ (with identity-bounded covariance), and the remainder of which are arbitrary in $\mathbb{R}^d$. We wish to produce a list of length $O(\alpha^{-1})$ containing at least one element at distance $\widetilde{O}(-\alpha^{-\frac{1}{2}})$ from the mean of $\mathcal{D}$. By letting $\mathcal{D}$ vary through mixture components, and using the list-decoding procedure as a preprocessing step in a separated mixture model estimation problem, we obtain an almost-linear runtime for robustly clustering heavy-tailed mixture models as well. Even from just a runtime perspective (i.e. assuming uncorrupted Gaussian components), this is the first improvement upon [520, 7].

Our starting point is a simple, polynomial-time reduction from list-decodable mean estimation to approximate k-PCA subroutines, which we use to define certificates of corruption. We then combine this reduction with a Ky Fan norm SDP solver we develop in Chapter 7 to solve the general list-decodable mean estimation problem in $\widetilde{O}(ndk)$ time. Finally, to go beyond this k-PCA computation runtime barrier, we design an $O(n^{1+\epsilon_0}d)$-time algorithm for solving list-decodable mean estimation using "split-or-cluster" steps based on solely *one-dimensional* projections. This final method is inspired by both filtering based on MMW responses, and the multifilter algorithm of [179]. In particular, the algorithm maintains a partial clustering over the dataset which is iteratively refined, where we bound the total number of points across all clusters.

Parameter estimation in generalized linear models. Consider the following parameter estimation problem: we observe independent $\{(X_i, y_i)\}_{i\in[n]} \sim \mathcal{D}$ supported on $\mathbb{R}^d \times \mathbb{R}$, and given a link function $\gamma : \mathbb{R}^2 \times \mathbb{R}$, we wish to estimate

$$\theta^\star := \mathrm{argmin}_{\theta\in\mathbb{R}^d} \mathbb{E}_{(X,y)\mathcal{D}} \left[\gamma(\langle\theta, X\rangle, y)\right].$$

This regression problem arises as a maximum likelihood estimation problem in exponential families. When γ is *regular* (i.e. convex, and Lipschitz or smooth), and $\mathcal{D}$ has appropriately bounded moments, the empirical risk minimizer attains good performance. We consider a robust variant of this problem

where an ϵ fraction of our dataset is contaminated.

By combining a gradient filtering scheme based on semidefinite programming, and a new noise-tolerant accelerated gradient method we develop, we solve the Lipschitz link function variant of the robust regression problem to $\sqrt{\epsilon}$ Euclidean norm accuracy (up to a scaling by problem regularity parameters) in time $\widetilde{O}(nd\sqrt{\kappa})$. Here, κ is a conditioning parameter of the regression problem. This is a generic improvement upon the "robust gradient descent" method of [447] which attained runtime $\widetilde{O}(nd\kappa)$, and applies to both Lipschitz, smooth link functions such as the logistic loss, as well as the Moreau envelope approximations to solely Lipschitz link functions such as the hinge loss.

We further specialize to the case of *robust linear regression*, when the X-marginal of $\mathcal{D}$ is 2-*to*-4 *hypercontractive* (has fourth moments relatively bounded by second moments). Here, we obtain both an accelerated nearly-linear time algorithm, and another nearly-linear time algorithm which obtains an *improved estimation rate*, albeit at a quadratically larger sample complexity. Our algorithm estimates at a Euclidean error rate scaling as $\sqrt{\kappa\epsilon}$ (where κ is the conditoning of the X-marginal covariance matrix). This can be seen as an interpolation between (accelerated) robust gradient methods, which estimate at rate $\kappa\sqrt{\epsilon}$, and sum-of-squares methods [53], which attain a rate $\epsilon^{\frac{3}{4}}$ but require exactly solving a large SDP and impose slightly stronger problem regularity assumptions.

Sub-Gaussian principal component analysis. Given an ϵ-corrupted set of n samples from a d-dimensional, mean-zero sub-Gaussian distribution, return a $(1-\delta)$-approximate top eigenvector of the covariance matrix. We give a simple filtering-based polynomial-time algorithm which solves this problem for $\delta = O(\epsilon \log \epsilon^{-1})$, which is essentially optimal. Moreover, we give an application of a Schatten-p norm packing SDP we develop in Chapter 7 to the robust PCA problem. The result is a near-linear time algorithm for mildly-"gapped" covariances, achieving $\delta = O(\sqrt{\epsilon \log \epsilon^{-1} \log d})$.

We find these results interesting for two additional conceptual reasons. First, they draw an explicit connection between Schatten norm SDP objectives and robust PCA. We establish this connection by noting that, while an adversary can elevate a single large direction (fooling operator norm SDPs), it can only create a bounded number of such large directions, which a Schatten norm objective can detect. Second, they are an example of covariance property estimation without using the strong algebraic relationships afforded by assuming *exactly Gaussian* samples or other *strong algebraic properties*, which to our knowledge had remained open until our work.

1.4.2 Chapter 9: Semi-Random Linear Systems

Consider the problem of solving a consistent linear system $\mathbf{A}x^\star = b$, given $(\mathbf{A}, b) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$. When $\mathbf{A}$ is full-rank and $n \geq d$, the system is overconstrained, and fast iterative methods (e.g. accelerated gradient descent) converge in nearly-linear time if $\mathbf{A}^\top \mathbf{A}$ is $O(1)$-multiplicatively close to a multiple of the identity. Interestingly, by giving additional consistent linear measurements (augmenting $\mathbf{A}$ with more rows), we can arbitrarily ruin the condition number of $\mathbf{A}$, and correspondingly hinder the performance of gradient-based iterative methods.

We also consider the underconstrained *sparse recovery* counterpart of this problem, where $n < d$. To make the problem meaningful (i.e. for the solution to be uniquely determined), we must ensure that there are no candidate solutions in the nullspace of $\mathbf{A}$. This is typically formulated as follows: $x^\star$ is assumed to be s-sparse, and $\mathbf{A}$ is assumed to satisfy the $O(s)$-*restricted nullspace property* (RNP), which means it has no $O(s)$-sparse vectors in its kernel. The RNP is essentially the minimum assumption on $\mathbf{A}$ required for the sparse recovery problem to be information-theoretically tractable. When $\mathbf{A}$ is assumed to satisfy the stronger property of being $O(1)$-approximately norm preserving on all $O(s)$-sparse vectors, it is said to satisfy the stronger *restricted isometry property* (RIP). Under this stronger RIP assumption, there exist fast iterative methods which converge in nearly-linear time.

However, a similar state of affairs (as in the overconstrained case) occurs when applying nearly-linear time solvers to "augmented" RIP matrices: by adding new consistent measurements, the problem does not become information-theoretically harder, but the convergence of fast iterative methods which work well on fully-RIP matrices becomes poor. This latter phenomenon is in fact reported in practical applications of fast iterative methods to sparse recovery problems [166, 280, 446], motivating the development of nearly-linear time algorithms tolerant to some amount of model misspecification. We investigate both of these problems through the lens of a *monotone semi-random adversary*. Concretely, in the overconstrained setting, we assume that there exists a diagonal reweighting matrix $\mathbf{W}^\star = \mathbf{diag}\left(w^\star\right), w^\star \in \mathbb{R}^n_{\geq 0}$ such that $\mathbf{A}^\top \mathbf{W}^\star \mathbf{A}$ is well-conditioned. This is an even weaker assumption on "planted structure" than the presence of a subset of well-conditioned rows, since we may set $w^\star$ to be a zero-one indicator vector. In the underconstrained (sparse recovery) setting, we similarly assume there exists a $\mathbf{W}^\star$ such that $\mathbf{A}^\top \mathbf{W}^\star \mathbf{A}$ satisfies RIP. In both cases, we develop nearly-linear time solvers for the exact recovery problem (where we observe $\mathbf{A}x^\star$), as well as variants of the noisy recovery problem (where we only observe $\mathbf{A}x^\star + \xi$ for some noise vector $\xi \in \mathbb{R}^n$).

In the overconstrained case, our method is based on approximately solving a more general *optimal diagonal preconditioning* problem, which asks to find a nonnegative diagonal matrix $\mathbf{W}$ such that the condition number of $\mathbf{A}^\top \mathbf{W} \mathbf{A}$ is as small as possible. To do so, we apply the matrix dictionary recovery SDP solver from Chapter 7 to a dictionary consisting of outer products of $\mathbf{A}$'s rows. We also solve a similar "outer preconditioning" problem from numerical linear algebra, by using the diagonal basis dictionary and $\mathbf{A}^\top \mathbf{A}$ as the constraint matrix.

In the setting of sparse recovery, there are known hardness results (i.e. NP-hardness of certifying RIP [57]) which suggest it may be computationally difficult to obtain a global reweighting of $\mathbf{A}$ which is RIP. We sidestep this hardness result by designing a "locally reweighted" projected gradient descent method based on ℓ_1-constrained optimization. Our algorithm progresses whenever we can compute a reweighting of $g = \mathbf{A}^\top \Delta$ satisfying certain decomposition properties which can be easily computationally verified, where $\Delta = b - \mathbf{A}x$ are the residuals of a current iterate x. Such a reweighting always exists, because when $\mathbf{A}$ is fully RIP, we prove such a decomposition always

succeeds, and hence we can let the weighting be $\sqrt{w^*}$. We rephrase the computation of each local reweighting as a convex subproblem, which we solve efficiently using stochastic gradient methods.

1.4.3 Chapter 10: Towards an Oracle Complexity of Sampling

We define three variants of the meta-problem (1.3), each described by a density family parameterized by a type of structure the negative log-likelihood (denoted f throughout this exposition) is assumed to satisfy, and a type of oracle access to the density. In each case, we provide results which yield a further understanding of the oracle complexity of sampling from said density family. We briefly remark that each family we consider has a counterpart in convex optimization theory and is hence similarly motivated; further, each optimization counterpart has a well-established oracle complexity theory, with nearly-matching upper and lower bounds.

Well-conditioned sampling. We consider the setting where the function f is assumed to have a condition number bounded by κ (i.e. $\nabla^2 f$ everywhere has all eigenvalues in $[\mu, L]$, and $\kappa := \frac{L}{\mu}$). We assume gradient and value oracle access to f.

On the upper bound side, we prove new *concentration bounds* on the norm of the gradient, $\|\nabla f(x)\|_2$ where $x \sim \mu$, for well-conditioned f. This allows us to design samplers attaining an oracle query complexity of $\widetilde{O}(\kappa d)$ for sampling to high accuracy in total variation from an explicit warm start, improving a line of recent results which had previously achieved $\widetilde{O}(\kappa d + \kappa^{1.5}\sqrt{d})$ [210, 128].

On the lower bound side, we provide hardness results for sampling from well-conditioned logconcave distributions using two popular frameworks: the Metropolis-adjusted Langevin algorithm (MALA), and multi-step Metropolized Hamlitonian Monte Carlo (HMC). We show that when initialized at suitably-adversarial exponentially warm start, no parameterization of MALA can obtain a better mixing rate than $\widetilde{\Omega}(\kappa d)$, and give similar (slightly weaker) hardness results for HMC. Because it is currently unknown how to construct subexponentially warm starts, our result may be interpreted positively as further motivation to study the construction of such high-quality starting distributions, which would have further implications beyond specific algorithms e.g. MALA or HMC.

Composite logconcave sampling with a restricted Gaussian oracle. We consider the setting where the function f is decomposable as $f = h + g$, where h has a condition number bound of κ and g is "simple" in a precise sense, but possibly non-smooth. Even in very special cases of this composite sampling problem where g was e.g. the indicator of a coordinatewise box in the standad basis, designing nontrivial high-accuracy samplers[3] was an outstanding open problem. Our definition of simplicity of g assumes that we can query a "restricted Gaussian oracle" (RGO) $\mathcal{O}(\eta, v)$, which

$$\text{produces a sample from the density } \propto \exp\left(-g(x) - \frac{1}{2\eta}\|x - v\|_2^2\right).$$

[3] Any logconcave density can be sampled to high accuracy using polynomially many queries to a value oracle [373]; we use "nontrivial" to mean a runtime going beyond a black-box citation to this general result.

We give an algorithm which samples from μ to high accuracy in total variation using $\widetilde{O}(\kappa d)$ iterations, each querying the RGO for g and the gradient oracle for h once.

The astute reader may note a similarity between the RGO definition and the proximal point method described in (1.4). In fact, our composite sampler follows from combining two pieces. First, we develop a basic sampler which achieves $\widetilde{O}(\kappa^2 d)$ iteration complexity. Next, we give a generic "proximal point method" for logconcave sampling based on the notion of RGO access, which we use to improve the condition number dependence of our basic sampler.

Logconcave finite sums. We consider the setting where the function f is expressible as a finite sum: $f = \frac{1}{n}\sum_{i\in[n]} f_i$, and all f_i are κ well-conditioned. We assume gradient oracle access to each f_i. Note that we can always simulate gradient oracle access to f via n gradient queries; however, by subsampling gradients, it is possible that we can do even better than a naive "full gradient" method.

We first give a basic method which achieves a gradient oracle query complexity of $\widetilde{O}(n + \kappa^2 d)$. We then boost this algorithm through our aforementioned proximal reduction framework, to obtain an overall query complexity of $\widetilde{O}(n + \kappa\sqrt{nd} + \kappa d)$. Up to a $\widetilde{O}(\max(1, \sqrt{\frac{n}{d}}))$ factor, this is essentially the best one could hope for without an improvement in the $n = 1$ case. Finally, we show that the $n = 1$ case of our finite sum sampler actually implies that well-conditioned densities are sampleable in $\widetilde{O}(\kappa d)$ calls to only a *value oracle*, suggesting that we may not yet have a mature understanding why gradient oracles are helpful in the well-conditioned setting.

1.5 Bibliography of Represented Material

Part I

- Chapter 2 of this thesis is based on the works "Relative Lipschitzness in Extragradient Methods and a Direct Recipe for Acceleration" [147] (published in ITCS 2021) and "Sharper Rates for Separable Minimax and Finite Sum Optimization via Primal-Dual Extragradient Methods" [299] (published in COLT 2022), joint with Michael B. Cohen, Yujia Jin, and Aaron Sidford.

- Chapter 3 of this thesis is based on the works "Variance Reduction for Matrix Games" [111] (published in NeurIPS 2019) and "Coordinate Methods for Matrix Games" [112] (published in FOCS 2020), joint with Yair Carmon, Yujia Jin, and Aaron Sidford.

- Chapter 4 of this thesis is based on the work "Coordinate Methods for Accelerating ℓ_∞ Regression and Faster Approximate Maximum Flow" [486] (published in FOCS 2018), joint with Aaron Sidford.

- Chapter 5 of this thesis is based on the works "A Direct $\widetilde{O}(1/\epsilon)$ Iteration Parallel Algorithm for Optimal Transport" [293] (published in NeurIPS 2019) and "Semi-Streaming Bipartite

Matching in Fewer Passes and Optimal Space" [39] (published in SODA 2022), joint with Sepehr Assadi, Arun Jambulapati, Yujia Jin, and Aaron Sidford.

- Chapter 6 of this thesis is based on the work "Regularized Box-Simplex Games and Dynamic Decremental Bipartite Matching" [284] (published in ICALP 2022), joint with Arun Jambulapati, Yujia Jin, and Aaron Sidford.

Part II

- Chapter 7 of this thesis is based on (parts of) the works "A Rank-1 Sketch for Matrix Multiplicative Weights" [108] (published in COLT 2019), "List-Decodable Mean Estimation in Nearly-PCA Time" [180] (published in NeurIPS 2021), "Fast and Near-Optimal Diagonal Preconditioning" [287] (preprint under submission), and "Robust Sub-Gaussian Principal Component Analysis and Width-Independent Schatten Packing" [289] (published in NeurIPS 2020), joint with Yair Carmon, Ilias Diakonikolas, John C. Duchi, Arun Jambulapati, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, Christopher Musco, and Aaron Sidford.

- Chapter 8 of this thesis is based on (parts of) the works "Clustering Mixture Models in Almost-Linear Time via List-Decodable Mean Estimation" [181] (published in STOC 2022), "Robust Regression Revisited: Acceleration and Improved Estimation Rates" (published in NeurIPS 2021), [180], and [289], joint with Ilias Diakonikolas, Arun Jambulapati, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, and Tselil Schramm.

- Chapter 9 of this thesis is based on (parts of) the works "Semi-Random Sparse Recovery in Nearly-Linear Time" [319] (preprint under submission) and [287], joint with Arun Jambulapati, Jonathan A. Kelner, Jerry Li, Allen Liu, Christopher Musco, and Aaron Sidford.

- Chapter 10 of this thesis is based on the works "Logsmooth Gradient Concentration and Tighter Runtimes for Metropolized Hamiltonian Monte Carlo" [344] (published in COLT 2020), "Structured Logconcave Sampling with a Restricted Gaussian Oracle" [346] (published in COLT 2021), and "Lower Bounds on Metropolized Sampling Methods for Well-Conditioned Distributions" [345] (published in NeurIPS 2021), joint with Yin Tat Lee and Ruoqi Shen.

Copyright notices

Other

We acknowledge the author's other research collaborations throughout the Ph.D. which have taught the author a variety of technical tools and perspectives and influenced his research style. A few of these collaborations have resulted in the works "Learning Populations of Parameters" [503] (published in NeurIPS 2017), "CoVeR: Learning Covariate-Specific Vector Representations with Tensor Decompositions" [504] (published in ICML 2018), "Acceleration with a Ball Optimization Oracle" [109] (published in NeurIPS 2020), and "Improved Sampling-to-Counting Reductions in High-Dimensional Expanders and Faster Parallel Determinantal Sampling" [26], joint with Nima Anari, Callum Burgess, Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Weihao Kong, Yin Tat Lee, Aaron Sidford, Gregory Valiant, Thuy-Duong Vuong, Teng Zhang, and James Zou.

Part I

Stochastic Variational Inequalities and Combinatorial Optimization

Chapter 2

Acceleration via Primal-Dual Extragradient Methods

This chapter is based on [147, 299], with Michael B. Cohen, Yujia Jin, and Aaron Sidford.

2.1 Introduction

In this chapter, we study the classic extragradient algorithms of mirror prox [415] and dual extrapolation [420] for solving variational inequalities (VIs) in monotone operators. This family of problems includes convex optimization and finding the saddle point of a convex-concave game. Due to applications of the latter to adversarial and robust training, extragradient methods have received significant recent attention in the machine learning community, see e.g. [121, 391, 277]. Further, extragradient methods have been the subject of increasing study by the theoretical computer science and optimization communities due to recent extragradient-based runtime improvements for problems including maximum flow [483] and zero-sum games [111, 112].

Given a Lipschitz monotone operator and a bounded strongly-convex regularizer, mirror prox [415] and dual extrapolation [420] achieve $O(T^{-1})$ regret for solving the associated VI after T iterations. This rate is worst-case optimal when the Lipschitzness of the operator and strong convexity of the regularizer are with respect to the Euclidean norm [431]. However, in certain structured problems related to VIs, alternative analyses and algorithms can yield improved rates. For instance, when minimizing a smooth convex function (i.e. one with a Lipschitz gradient), it is known that accelerated rates of $O(T^{-2})$ are attainable, improving upon the standard $O(T^{-1})$ extragradient rate for the naive associated VI (see Appendix A.2). Further, algorithms inspired by extragradient methods have been developed recovering the $O(T^{-2})$ rate [192, 530].

Additionally, alternative analyses of extragradient methods, such as optimism [452] and area convexity [483] have shown that under assumptions beyond a Lipschitz operator and a strongly convex regularizer, improved rates can be achieved. These works leveraged modified algorithms which run efficiently under such non-standard assumptions. Further, the area convexity-based methods of [483] have had a number of implications, including faster algorithms for ℓ_∞ regression, maximum flow and multicommodity flow [483] as well as improved parallel algorithms for work-efficient positive linear programming [91] and optimal transport [293].

In this chapter we seek a better understanding of these structured problems and the somewhat disparate-seeming analyses and algorithms for solving them. We ask, *are the algorithmic changes enabling these results necessary? Can standard mirror prox and dual extrapolation be leveraged to obtain these results? Can we unify analyses for these problems, and further clarify the relationship between acceleration, extragradient methods, and primal-dual methods?* Finally, *can we use these novel perspectives to broaden the optimization toolbox through new methods with improved properties?*

2.1.1 Relative Lipschitzness in extragradient methods

Towards addressing these questions, we provide a general condition, which we term *relative Lipschitzness* (cf. Definition 1), and analyze the convergence of mirror prox and dual extrapolation for a monotone relatively Lipschitz operator.[1] This condition is derived directly from the standard analysis of the methods and is stated in terms of a straightforward relationship between the operator g and the regularizer r which define the algorithm. Our condition is inspired by both area convexity as well as the "relative smoothness" condition in convex optimization [64, 375], and can be thought of as a generalization of the latter to variational inequalities (see Lemma 3). Further, through this analysis we show that standard extragradient methods directly yield accelerated rates for smooth minimization and recover the improved rates of [483] for box-constrained ℓ_∞ regression, making progress on the questions outlined above. We also show our methods recover certain randomized accelerated rates and have additional implications, outlined below.

Extragradient methods directly yield acceleration. In Section 2.4, we show that applying algorithms of [415, 420] to a minimax formulation of $\min_{x\in\mathbb{R}^d} f(x)$, when f is smooth and strongly convex, yields accelerated minimization rates when analyzed via relative Lipschitzness. Specifically, we consider the following problem, termed the *Fenchel game* in [530]:

$$\min_{x\in\mathbb{R}^d}\max_{y\in\mathbb{R}^d} \langle y, x\rangle - f^*(y), \tag{2.1}$$

and show that when f is μ-strongly convex and L-smooth, $O(\sqrt{L/\mu})$ iterations of either mirror prox [415] or dual extrapolation [420] produces an average iterate which halves the function error

[1] A somewhat similarly-named property appeared in [374], which also studied mirror descent algorithms under relaxed conditions; however, their property $\|g(x)\|_*^2 \le \frac{MV_x(y)}{\|y-x\|^2}$ for all x, y, is different than the one we propose.

of f. By repeated application, this yields an accelerated linear rate of convergence and the optimal $O(T^{-2})$ rates for non-strongly convex, smooth function minimization by a reduction [551]. Crucially, to attain this rate we give a sharpened bound on the relative Lipschitzness of the gradient operator of (2.1) with respect to a primal-dual regularizer.

Our result advances a recent line of research, [3, 4, 192], on applying primal-dual analyses to shed light on the mysterious nature of acceleration. Specifically, [3, 4] show that the classical algorithm of [417] can be rederived via applying primal-dual "optimistic" dynamics, inspired by the framework of [452]. Further, [192] showed that an appropriate discretization of dynamics inspired by extragradient algorithms yields an alternative accelerated algorithm. While these results clarify the primal-dual nature of acceleration, additional tuning is ultimately required to obtain their final algorithms and analysis. We obtain acceleration as a direct application of known frameworks, i.e. standard mirror prox and dual extrapolation, applied to the formulation (2.1), and hope this helps demystify acceleration.

In Appendix A.5, we further show that analyzing extragradient methods tailored to strongly monotone operators via relative Lipschitzness, and applying this more fine-grained analysis to a variant of the objective (2.1), also yields an accelerated linear rate of convergence. The resulting proof strategy extends readily to accelerated minimization of smooth and strongly convex functions in general norms, as we discuss at the end of Section 2.4, and we believe it may be of independent interest.

Finally, we remark that there has been documented difficulty in accelerating the minimization of relatively smooth functions [265]; this was also explored more formally by [197]. It is noted in [265], as well as suggested in others (e.g. in the development of area convexity [483]) that this discrepancy may be due to acceleration fundamentally requiring conditions on relationships between groups of three points, rather than two. Our work, which presents an alternative three-point condition yielding accelerated rates, sheds light on this phenomenon and we believe it is an interesting future direction to explore the relationships between our condition and other alternatives in the literature which are known to yield acceleration.

Area convexity for bilinear box-simplex games. In Section 2.5, we draw a connection between relative Lipschitzness and the notion of an "area convex" regularizer, proposed by [483]. Area convexity is a property which weakens strong convexity, but is suitable for extragradient algorithms with a linear operator. It was introduced in the context of solving a formulation of approximate undirected maximum flow via box-constrained ℓ_∞ regression, or more generally approximating bilinear games between a box variable and a simplex variable. The algorithm of [483] applied to bilinear games was a variant of standard extragradient methods and analyzed via area convexity, which was proven via solving a subharmonic partial differential equation. We show that mirror prox, as analyzed by a local variant of relative Lipschitzness, yields the same rate of convergence as implied by area convexity, for box-simplex games. Our proof of this rate is straightforward and based on a simple

Cauchy-Schwarz argument after demonstrating local stability of iterates.

Randomized extragradient methods via local variance reduction. In general, the use of stochastic operator estimates in the design of extragradient algorithms for solving general VIs is not as well-understood as their use in the special case of convex function minimization. The best-known known stochastic methods for solving VIs [302] with bounded-variance stochastic estimators obtain $O(T^{-1/2})$ rates of convergence; this is by necessity, from known classical lower bounds on the rate of the special case of stochastic convex optimization [414]. Towards advancing the randomized extragradient toolkit, we ask: when can improved $O(T^{-1})$ rates of convergence be achieved by stochastic algorithms for solving specific VIs and fine-grained bounds on estimator variance (i.e. more local notions of variance)? This direction is inspired by analogous results in convex optimization, where reduced-variance and accelerated rates have been obtained, matching and improving upon their deterministic counterparts [300, 472, 18, 347, 425, 553].

For the special case of bilinear games, this question was recently addressed by the works [432, 111], using proximal reductions to attain improved rates. In this work, we give a framework for direct stochastic extragradient method design bypassing the variance bottleneck limiting prior algorithms to a $O(T^{-1/2})$ rate of convergence for problems with block-separable structure. We identify a particular criterion of randomized operators used in the context of extragradient algorithms (cf. Proposition 2) which enables $O(T^{-1})$ rates of convergence. Our approach is a form of "local variance reduction", where estimators in an iteration of the method share a random seed and we take expectations over the whole iteration in the analysis. Our improved estimator design exploits the separable structure of the problem; it would be interesting to design a more general variance reduction framework for randomized extragradient methods.

Formally, we apply our local variance reduction framework in Section 2.6 to show that an instance of our new randomized extragradient methods recover acceleration for coordinate-smooth functions, matching the known tight rates of [553, 425]. Along the way, we give a variation of relative Lipschitzness capturing an analagous property between a locally variance-reduced randomized gradient estimator and a regularizer, which we exploit to obtain our runtime. We note that a similar approach was taken in [486] to obtain faster approximate maximum flow algorithms in the bilinear minimax setting; here, we generalize this strategy and give conditions under which our variance reduction technique obtains improved rates for extragradient methods more broadly.

Additional contributions. A minor contribution of our work is that we show (Appendix A.3) that relative Lipschitzness implies new rates for minimax convex-concave optimization, taking a step towards closing the gap with lower bounds with *fine-grained* dependence on problem parameters. Under operator-norm bounds on blocks of the Hessian of a convex-concave function, as well as blockwise strong convexity assumptions, [545] showed a lower bound on the convergence rate to obtain an ϵ-approximate saddle point. When the blockwise operator norms of the Hessian are roughly equal, [366] gave an algorithm matching the lower bound up to a polylogarithmic factor,

using an alternating scheme repeatedly calling an accelerated proximal point reduction. Applying our condition with a strongly monotone variant of the mirror prox algorithm of [415] yields a new fine-grained rate for minimax optimization, improving upon the runtime of [366] for a range of parameters. Our algorithm is simple and the analysis follows directly from a tighter relative Lipschitzness bound; we note the same result can also be obtained by considering an operator norm bound of the problem after a rescaling of space, but we include this computation because it is a straightforward implication of our condition. We also remark that these results serve as a baseline rate for smooth minimax optimization, which is later sharpened by developments in Section 2.7.

Finally, in Appendix A.7, we discuss the relation of relative Lipschitzness to another framework for analyzing extragradient methods: namely, we note that our proof of the sufficiency of relative Lipschitzness recovers known bounds for optimistic mirror descent [452].

2.1.2 Sharper rates for separable minimax and finite sum optimization

In the second part of this chapter, we further study several fundamental families of optimization problems, namely (separable) minimax optimization, finite sum optimization, and minimax finite sum optimization (which generalizes both), and provide new state-of-the-art rates for these problem families using the framework developed in Section 2.1.1. These families have received widespread recent attention from the optimization community due to their prevalence in modeling tasks arising in modern data science. For example, minimax optimization has been used in both convex-concave settings and beyond to model robustness to (possibly adversarial) noise in many training tasks [378, 451, 244]. Moreover, finite sum optimization serves as a fundamental subroutine in many of the empirical risk minimization tasks of machine learning today [92]. Nonetheless, and perhaps surprisingly, there remain gaps in our understanding of the optimal rates for these problems. Toward closing these gaps, we provide new accelerated algorithms improving upon the state-of-the-art for each family of problems.

Our results build upon our earlier framework, in particular its application of primal-dual extragradient methods to recover accelerated rates for smooth, convex optimization in Appendix A.5. This application considers the problem[2]

$$\min_{x \in \mathcal{X}} f(x) + \frac{\mu}{2} \|x\|^2 \text{ for } L\text{-smooth and convex } f, \tag{2.2}$$

and its equivalent primal-dual formulation as an appropriate "Fenchel game"

$$\min_{x \in \mathcal{X}} \max_{x^* \in \mathcal{X}^*} \frac{\mu}{2} \|x\|^2 + \langle x^*, x \rangle - f^*(x^*), \text{ where } f^* \text{ is the convex conjugate of } f\,. \tag{2.3}$$

As discussed earlier, our framework shows that applying extragradient methods [415, 420] and analyzing them through a condition known as *relative Lipschitzness* recovers an accelerated gradient

[2]Throughout, $\mathcal{X}, \mathcal{Y}$ are unconstrained, Euclidean spaces (see Section 2.2).

query complexity for computing (2.2), which was known to be optimal [418].

Both the Fenchel game [3, 530] and the relative Lipschitzness property (independently proposed in [496]) have a longer history, discussed in Section 2.1.3. This chapter is particularly motivated by their synthesis in the earlier sections of this chapter, which used these tools to provide a general recipe for designing accelerated methods. This recipe consists of the following ingredients.

1. Choose a primal-dual formulation of an optimization problem and a regularizer, r.
2. Bound iteration costs, i.e. the cost of implementing mirror steps with respect to r.
3. Bound the relative Lipschitzness of the gradient operator of the problem with respect to r.

In our basic accelerated algorithm in Appendix A.5, this recipe was applied with (2.3) as the primal-dual formulation and $r(x, x^*) := \frac{\mu}{2}\|x\|^2 + f^*(x^*)$. Further, it was shown that each iteration could be implemented (implicitly) with $O(1)$ gradient queries and that the gradient operator Φ of the objective (2.3) is $O(\sqrt{L/\mu})$-relatively Lipschitz with respect to r. Combining these ingredients gave the accelerated rate for (2.3).

We broaden this primal-dual extragradient approach and add new recipes to the optimization cookbook. As a result, we obtain methods with improved rates for minimax optimization, finite sum optimization, and minimax finite sum optimization. We follow similar approaches as our basic recipe but change the ingredients with different primal-dual formulations, regularizers, extragradient methods, and analyses. In the following sections, we discuss each problem family, our new results and approach, and situate them in the relevant literature.

Separable minimax optimization.

In Section 2.7, we study separable convex-concave minimax optimization problems of the form[3]

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} F_{\text{mm}}(x, y) := f(x) + h(x, y) - g(y), \tag{2.4}$$

where f is L^{x}-smooth and μ^{x}-strongly convex, g is L^{y}-smooth and μ^{y}-strongly convex, and h is convex-concave and twice-differentiable with $\left\|\nabla^2_{xx}h\right\| \le \Lambda^{\text{xx}}$, $\left\|\nabla^2_{xy}h\right\| \le \Lambda^{\text{xy}}$, and $\left\|\nabla^2_{yy}h\right\| \le \Lambda^{\text{yy}}$. Our goal is to compute a pair of points (x, y) with bounded duality gap with respect to F_{mm}: $\text{Gap}_{F_{\text{mm}}}(x, y) \le \epsilon$ (see Section 2.2 for definitions).

The problem family (2.4) contains as a special case the following family of convex-concave minimax optimization problems with bilinear coupling (with $\Lambda^{\text{xx}} = \Lambda^{\text{yy}} = 0$ and $\Lambda^{\text{xy}} = \|A\|$):

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x) + \left(y^\top \mathbf{A}x - \langle b, y\rangle + \langle c, x\rangle\right) - g(y). \tag{2.5}$$

[3]Our results in Section 2.7 apply generally to non-twice differentiable, gradient Lipschitz h, but we use these assumptions for simplicity in the introduction. All norms are Euclidean (see Section 2.2 for relevant definitions).

Problem (2.5) has been widely studied in the optimization literature, dating at least to the classic work of [116], which used (2.5) to relax convex optimization with affine constraints related to imaging inverse problems. Problem (2.5) also encapsulates convex-concave quadratics and has been used to model problems in reinforcement learning [201] and decentralized optimization [333].

Our results. We give the following result on solving (2.4).

Theorem 1 (informal, cf. Theorem 25)**.** *There is an algorithm that, given $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ satisfying $\mathrm{Gap}_{F_{\mathrm{mm}}}(x_0, y_0) \leq \epsilon_0$, returns (x, y) with $\mathrm{Gap}_{F_{\mathrm{mm}}}(x, y) \leq \epsilon$ using T gradient evaluations to f, h, and g for*

$$T = O\left(\kappa_{\mathrm{mm}} \log\left(\frac{\kappa_{\mathrm{mm}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\mathrm{mm}} := \sqrt{\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}}.$$

In the special case of (2.5), Theorem 1 matches a lower bound of [545], which applies to the family of quadratic minimax problems obeying our smoothness and strong convexity bounds. More generally, Theorem 1 matches the lower bound whenever Λ^{xx} and Λ^{yy} are sufficiently small compared to the remaining parameters, improving prior state-of-the-art rates [533] in this regime.

By applying reductions based on explicit regularization used in [366], Theorem 1 also yields analogous accelerated rates depending polynomially on the desired accuracy when we either f, g, or both are not strongly convex. For conciseness, in this paper we focus on the strongly convex-concave regime discussed previously in this section.

Our approach. Our algorithm for solving (2.4) is based on the simple observation that minimax problems with the separable structure can be effectively "decoupled" by using convex conjugation on the components f and g. In particular, following a similar recipe as our smooth convex optimization application in Appendix A.5, we rewrite (an appropriate regularized formulation of) the problem (2.4) using convex conjugates as follows:

$$\min_{x \in \mathcal{X}, y^* \in \mathcal{Y}^*} \max_{y \in \mathcal{Y}, x^* \in \mathcal{X}^*} \frac{\mu^{\mathsf{x}}}{2}\|x\|^2 - \frac{\mu^{\mathsf{y}}}{2}\|y\|^2 + \langle x^*, x\rangle - \langle y^*, y\rangle + h(x, y) - f^*(x^*) + g^*(y^*).$$

Further, we define the regularizer $r(x, y, x^*, y^*) := \frac{\mu^{\mathsf{x}}}{2}\|x\|^2 + \frac{\mu^{\mathsf{y}}}{2}\|y\|^2 + f^*(x^*) + g^*(y^*)$. Finally, we apply an extragradient method for strongly monotone operators to our problem, using this regularizer. We demonstrate efficient implementability, and analyze the relative Lipschitzness of the problem's gradient operator with respect to r, yielding Theorem 1. In the final gradient oracle complexity, our method obtains the accelerated trade-off between primal and dual blocks for $\frac{\mu^{\mathsf{x}}}{2}\|x\|^2 + \langle x^*, x\rangle - f^*(x^*)$ and its $\mathcal{Y}$ analog, for the separable parts f and g respectively. It also obtains an unaccelerated rate for the h component, by bounding the relative Lipschitzness corresponding to h via our assumptions.

Prior work. Many recent works obtaining improved rates for minimax optimization under smoothness and strong convexity restrictions (including Appendix A.3 of this thesis) concentrate

on a more general family of problems of the form:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} F(x,y). \tag{2.6}$$

Typically, these works assume (for simplicity, assuming F is twice-differentiable), $\nabla^2_{xx}F$ is bounded between $\mu^{\mathsf{x}}\mathbf{I}$ and $\Lambda^{\mathsf{xx}}\mathbf{I}$ everywhere, $\nabla^2_{yy}F$ is bounded between $\mu^{\mathsf{y}}\mathbf{I}$ and $\Lambda^{\mathsf{yy}}\mathbf{I}$ everywhere, and $\nabla^2_{xy}F$ is operator norm bounded by Λ^{xy}. It is straightforward to see that (2.6) contains (2.4) as a special case, by setting $f \leftarrow \frac{\mu^{\mathsf{x}}}{2}\|\cdot\|^2$, $g \leftarrow \frac{\mu^{\mathsf{y}}}{2}\|\cdot\|^2$, and $h \leftarrow F - f + g$.

For (2.6), under gradient access to F, the works [366, 533], and Appendix A.3 presented different approaches yielding a variety of query complexities. Letting $\Lambda^{\max} := \max\left(\Lambda^{\mathsf{xx}}, \Lambda^{\mathsf{xy}}, \Lambda^{\mathsf{yy}}\right)$, these complexities scaled respectively as[4]

$$\widetilde{O}\left(\sqrt{\frac{\max\left(\Lambda^{\mathsf{xx}}, \Lambda^{\mathsf{xy}}, \Lambda^{\mathsf{yy}}\right)^2}{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right), \widetilde{O}\left(\sqrt{\frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}}} + \sqrt{\frac{\Lambda^{\mathsf{xy}}\Lambda^{\max}}{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right), \widetilde{O}\left(\frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right).$$

The state-of-the-art rate (ignoring logarithmic factors) is due to [533], which obtained the middle gradient query complexity above. Theorem 1 matches the rate obtained by Appendix A.3 and improves upon [366] in some regimes. Notably, Theorem 1 never improves upon [533] in the general regime, up to logarithmic factors. On the other hand, the method in Theorem 1 uses only a single loop, as opposed to the multi-loop methods in [366, 533] which lose logarithmic factors.

Up to logarithmic factors, there is a gap between [533] and the lower bound of [545] only when $\Lambda^{\mathsf{xy}} \ll \Lambda^{\max}$. We close this gap for minimax problems admitting the separable structure (2.5). In the special case of quadratic problems, prior work, [533], proposed a recursive approach which obtained a rate comparable to that of Theorem 1, albeit larger by subpolynomial factors.

Concurrent work. A pair of independent and concurrent works [332, 501] obtained variants of Theorem 1. Their results were stated under the restricted setting of bilinear coupling (2.5), but they each provided alternative results under (different) weakenings of our strong convexity assumptions. The algorithm of [501] is closer to the one developed in this chapter (also going through a primal-dual lifting), although the ultimate methods and analyses are somewhat different. Though our results were obtained independently, our presentation was informed by a reading of [332, 501] for a comparison.

Finite sum optimization

In Section 2.8, we study finite sum optimization problems of the form

$$\min_{x\in\mathcal{X}} F_{\text{fs}}(x) := \frac{1}{n}\sum_{i\in[n]} f_i(x), \tag{2.7}$$

[4] $\widetilde{O}$ hides logarithmic factors throughout, see Section 2.2.

where f_i is L_i-smooth for each $i \in [n]$, and $\frac{1}{n}\sum_{i\in[n]} f_i$ is μ-strongly convex. We focus on the strongly convex regime; through generic reductions [551], our results yield accelerated rates depending polynomially on the desired accuracy, without the strong convexity assumption.

Methods for solving (2.7) have garnered substantial interest because of their widespread applicability to empirical risk minimization problems over a dataset of n points, which encapsulate a variety of (generalized) regression problems in machine learning (see [92] and references therein).

Our results. We give the following result on solving (2.7).

Theorem 2 (informal, cf. Theorem 8, Corollary 4)**.** *There is an algorithm that, given $x_0 \in \mathcal{X}$ satisfying $F_{\text{fs}}(x_0) - F_{\text{fs}}(x_\star) \le \epsilon_0$ where $x_\star$ minimizes F_{fs}, returns $x \in \mathcal{X}$ with $\mathbb{E}F_{\text{fs}}(x) - F_{\text{fs}}(x_\star) \le \epsilon$ using T gradient evaluations (each to some f_i) for*

$$T = O\left(\kappa_{\text{fs}} \log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\text{fs}} := n + \sum_{i\in[n]} \frac{\sqrt{L_i}}{\sqrt{n\mu}}.$$

Our approach. Our algorithm for solving (2.7) builds upon an accelerated coordinate descent variant developed in Section 2.6, which developed an analysis of a randomized extragradient method to do so. We consider an equivalent primal-dual formulation of (a regularized variant of) (2.7), inspired by analogous developments in the ERM literature [477, 478]:

$$\min_{x\in\mathcal{X}} \max_{\{x_i^*\}_{i\in[n]}\subset\mathcal{X}^*} \frac{\mu}{2}\|x\|^2 + \frac{1}{n}\sum_{i\in[n]} \left(\langle x_i^*, x\rangle - f_i^*(x_i^*)\right).$$

Our algorithm then solves this regularized primal-dual game to high precision.

A key building block of our method is a randomized extragradient method which is compatible with strongly monotone problems. To this end, we extend the way the randomized extragradient method is applied in Section 2.6, which does not directly yield a high-precision guarantee. We proceed as follows: for roughly κ_{fs} iterations (defined in Theorem 2) of our method, we run the non-strongly monotone randomized mirror prox method of Section 2.6 to obtain a regret bound. We then subsample a random iterate, which we show halves an appropriate potential in expectation via our regret bound and strong monotonicity. We then recurse on this procedure to obtain a high-precision solver.

Prior work. Developing accelerated algorithms for (2.7) under our regularity assumptions has been the subject of a substantial amount of research effort in the community, see e.g. [364, 227, 478, 17] and references therein. Previously, the state-of-the-art gradient query complexities (up to

logarithmic factors) for (2.7) were obtained by [364, 227, 17],[5] and scaled as

$$\widetilde{O}\left(n + \sqrt{\frac{\sum_{i\in[n]} L_i}{\mu}}\right). \tag{2.8}$$

Rates such as (2.8), which scale as functions of $\sum_{i\in[n]} \frac{L_i}{\mu}$, arise in known *variance reduction*-based approaches [300, 168, 472, 17] due to their applications of a "dual strong convexity" lemma (e.g. Theorem 1, [300] or Lemma 2.4, [17]) of the form

$$\|\nabla f_i(x) - \nabla f_i(\bar{x})\|^2 \leq 2L_i \left(f_i(\bar{x}) - f_i(x) - \langle \nabla f_i(x), \bar{x} - x\rangle\right).$$

The analyses of e.g. [300, 17] sample $i \in [n]$ proportional to L_i, allowing them to bound the variance of a resulting gradient estimator by a quantity related to the divergence in F_{fs}.

The rate in (2.8) is known to be optimal in the uniform smoothness regime [539], but in a more general setting its optimality is unclear. Theorem 2 shows that the rate can be improved for sufficiently non-uniform L_i. In particular, Cauchy-Schwarz shows that the quantity κ_{fs} is never worse than (2.8), and improves upon it by a factor asymptotically between 1 and $\sqrt{n}$ when the $\{L_i\}_{i\in[n]}$ are non-uniform. Moreover, even in the uniform smoothness case, Theorem 2 matches the tightest rate in [17] up to an additive $\log \kappa_{\text{fs}}$ term, as opposed to an additional multiplicative logarithmic overhead incurred by the reduction-based approaches of [364, 227].

Our rate's improvement over (2.8) is comparable to a similar improvement that was achieved previously in the literature on coordinate descent methods. In particular, [347] first obtained a (generalized) partial derivative query complexity comparable to (2.8) under coordinate smoothness bounds, which was later improved to a query complexity comparable to Theorem 2 by [553, 425]. Due to connections between coordinate-smooth optimization and empirical risk minimization (ERM) previously noted in the literature [477, 478], it is natural to conjecture that the rate in Theorem 2 is achieveable for finite sums (2.7) as well. However, prior to our work (to our knowledge) this rate was not known, except in special cases e.g. linear regression [10].

Our method is based on using a primal-dual formulation of (2.7) to design our gradient estimators. It attains the rate of Theorem 8 by sampling summands proportional to $\sqrt{L_i}$, trading off primal and dual variances through a careful coupling. It can be viewed as a modified dual formulation to the coordinate descent algorithm in Section 2.6, which used primal-dual couplings inspired by the fine-grained accelerated algorithms of [553, 425]. We believe our result sheds further light on the duality between coordinate-smooth and finite sum optimization, and gives an interesting new approach for algorithmically leveraging primal-dual formulations of finite sum problems.

[5] There have been a variety of additional works which have also attained accelerated rates for either the problem (2.7) or its ERM specialization, see e.g. [167, 547, 338, 549]. However, to the best of our knowledge these do not improve upon the state-of-the-art rate of [17] in our setting.

Minimax finite sum optimization

In Section 2.9, we study a family of minimax finite sum optimization problems of the form

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} F_{\text{mmfs}}(x,y) := \frac{1}{n}\sum_{i\in[n]}\left(f_i(x)+h_i(x,y)-g_i(y)\right). \tag{2.9}$$

We assume f_i is L_i^{x}-smooth, g_i is L^{y}-smooth, and h_i is convex-concave and twice-differentiable with blockwise operator norm bounds Λ_i^{xx}, Λ_i^{xy}, and Λ_i^{yy} for each $i \in [n]$. We also assume the whole problem is μ^{x}-strongly convex and μ^{y}-strongly concave.

We propose the family (2.9) because it encapsulates (2.6) and (2.7), and is amenable to techniques from solving both. Moreover, (2.9) is a natural description of instances of (2.6) which arise from primal-dual formulations of empirical risk minimization problems, e.g. [547, 529]. It also generalizes natural minimax finite sum problems previously considered in e.g. [111].

Our results. We give the following result on solving (2.9).

Theorem 3 (informal, cf. Theorem 9)**.** *There is an algorithm that, given $(x_0, y_0) \in \mathcal{X}\times\mathcal{Y}$ satisfying $\text{Gap}_{F_{\text{mmfs}}}(x_0,y_0) \le \epsilon_0$, returns (x,y) with $\mathbb{E}\text{Gap}_{F_{\text{mmfs}}}(x,y) \le \epsilon$ using T gradient evaluations, each to some f_i, g_i, or h_i, where*

$$T = O\left(\kappa_{\text{mmfs}}\log\left(\kappa_{\text{mmfs}}\right)\log\left(\frac{\kappa_{\text{mmfs}}\epsilon_0}{\epsilon}\right)\right),$$
$$\text{for } \kappa_{\text{mmfs}} := n + \frac{1}{\sqrt{n}}\sum_{i\in[n]}\left(\sqrt{\frac{L_i^{\text{x}}}{\mu^{\text{x}}}}+\sqrt{\frac{L_i^{\text{y}}}{\mu^{\text{y}}}}+\frac{\Lambda_i^{\text{xx}}}{\mu^{\text{x}}}+\frac{\Lambda_i^{\text{xy}}}{\sqrt{\mu^{\text{x}}\mu^{\text{y}}}}+\frac{\Lambda_i^{\text{yy}}}{\mu^{\text{y}}}\right).$$

The rate in Theorem 3 captures (up to a logarithmic factor) both of the rates in Theorems 1 and 2, when (2.9) is appropriately specialized. It can be more generally motivated as follows. When n is not the dominant term in Theorem 2's bound, the remaining term is $\sqrt{n}$ times the average rate attained by Nesterov's accelerated gradient method [417] on each summand in (2.7). This improves upon the factor of n overhead which one might naively expect from computing full gradients. In similar fashion, Theorem 3 attains a rate (up to an additive n, and logarithmic factors) which is $\sqrt{n}$ times the average rate attained by Theorem 1 on each summand in (2.9).

Our approach. Our algorithm for solving (2.9) is a natural synthesis of the earlier algorithms suggested in Section 2.1.2. However, to obtain our results we apply additional techniques to bypass complications which arise from the interplay between the minimax method and the finite sum method, inspired by [111]. In particular, to obtain our tightest rate we would like to subsample the components in our gradient operator corresponding to $\{f_i\}_{i\in[n]}, \{g_i\}_{i\in[n]}, \{h_i\}_{i\in[n]}$ all at different frequencies when applying the randomized extragradient method. These different sampling distributions introduce dependencies between iterates, and make our randomized estimators no longer "unbiased" for the true gradient operator to directly incur the randomized extragradient analysis.

To circumvent this difficulty, we obtain our result via a partial decoupling, treating components corresponding to $\{f_i\}_{i\in[n]}$, $\{g_i\}_{i\in[n]}$ and those corresponding to $\{h_i\}_{i\in[n]}$ separately. For the first two aforementioned components, which are separable and hence do not interact, we pattern an expected relative Lipschitzness analysis for each block, similar to the finite sum optimization. For the remaining component $\{h_i\}_{i\in[n]}$, we develop a variance-reduced stochastic method which yields a relative variance bound. We put these pieces together in Proposition 5, a new randomized extragradient method analysis, to give a method with a convergence rate of roughly

$$n + \frac{1}{\sqrt{n}} \sum_{i\in[n]} \left(\sqrt{\frac{L_i^{\mathsf{x}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{L_i^{\mathsf{y}}}{\mu^{\mathsf{y}}}} \right) + (\kappa_{\text{mmfs}}^h)^2, \text{ where } \kappa_{\text{mmfs}}^h := \frac{1}{n} \sum_{i\in[n]} \left(\frac{\Lambda_i^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda_i^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda_i^{\mathsf{yy}}}{\mu^{\mathsf{y}}} \right).$$

The dependence on all pieces above is the same as in Theorem 3, except for the term corresponding to the $\{h_i\}_{i\in[n]}$. To improve this dependence, we wrap our solver in an "outer loop" proximal point method which solves a sequence of γ-regularized variants of (2.9). We obtain our final claim by trading off the terms n and $(\kappa_{\text{mmfs}}^h)^2$ through our choice of γ, which yields the accelerated convergence rate of Theorem 3.

Prior work. To our knowledge, there have been relatively few results for solving (2.9) under our fine-grained assumptions on problem regularity, although various stochastic minimax algorithms have been developed in natural settings [302, 432, 277, 111, 121, 112, 16]. For the general problem of solving $\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} \frac{1}{n} \sum_{i\in[n]} F_i(x, y)$ where F_i is L_i-smooth and convex-concave, and the whole problem is μ^{x}-strongly convex and μ^{y}-strongly concave, perhaps the most direct comparisons are Section 5.4 of [111] and Theorem 15 of [507]. In particular, [111] provided a high-precision solver using roughly

$$\widetilde{O}\left(n + \frac{1}{\sqrt{n}} \sum_{i\in[n]} \frac{L_i}{\mu} \right)$$

gradient queries, when $\mu^{\mathsf{x}} = \mu^{\mathsf{y}} = \mu$. This is recovered by Theorem 9 (possibly up to logarithmic factors) in the special setting of $f_i = g_i \leftarrow 0$, $\mu^{\mathsf{x}} = \mu^{\mathsf{y}} \leftarrow \mu$, and $\Lambda_i^{\mathsf{xx}} = \Lambda_i^{\mathsf{xy}} = \Lambda_i^{\mathsf{yy}} \leftarrow L_i$. More generally, [111] gave a result depending polynomially on the desired accuracy without the strongly convex-concave assumption, which follows from a variant of Theorem 9 after applying the explicit regularization in [366] that reduces to the strongly convex-concave case.

Moreover, Theorem 15 of [507] provided a high-precision solver using roughly

$$\widetilde{O}\left(n + \frac{1}{\sqrt{n}} \sum_{i\in[n]} \frac{L_i}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} \right)$$

gradient queries. Our work recovers (and sharpens dependences in) this result for minimax finite sum problems where each summand has the bilinear coupling (2.5). In the more general setting

where each summand only has a uniform smoothness bound, the [507] result can be thought of as the accelerated finite sum analog of the main claim in [366], which is incomparable to our Theorem 1. In a similar way, the rate of [507] is incomparable to Theorem 3, and each improves upon the other in different parameter regimes. We believe designing a single algorithm which obtains the best of both worlds for (2.9) is an interesting future direction.

2.1.3 Additional related work

We give a brief discussion of several lines of work which our results build upon, and their connection with the techniques used in this paper.

Acceleration via primal-dual extragradient methods. Our algorithms are based on *extragradient methods*, a framework originally proposed by [329] which was later shown to obtain optimal rates for solving Lipschitz variational inequalities in [415, 420]. There have been various implementations of extragradient methods including mirror prox [415] and dual extrapolation [420]; we focus on adapting the former in this work. Variations of extragradient methods have been studied in the context of primal-dual formulations of smooth convex optimization [3, 530], and are known to obtain optimal (accelerated) rates in this setting. In particular, the relative Lipschitzness analysis of acceleration in this chapter is motivated by developments in the bilinear setting, namely the area convexity framework of [483]. We further build upon these works by using primal-dual formulations to design accelerated algorithms in various settings beyond smooth convex optimization, namely (2.6), (2.7), and (2.9).

Acceleration under relative regularity assumptions. This chapter builds upon a framework for analyzing extragradient methods known as *relative Lipschitzness*, which was proposed independently by [496]. We demonstrate that this framework (and randomized variants thereof) obtains improved rates for primal-dual formulations beyond those studied in prior works.

Curiously, our applications of the relative Lipschitzness framework reveal that the regularity conditions our algorithms require are weaker than standard assumptions of smoothness in a norm. In particular, several technical requirements of specific components of our algorithms are satisfied by setups with regularity assumptions generalizing and strengthening the *relative smoothness* assumption of [64, 375]. This raises interesting potential implications in terms of the necessary regularity assumptions for non-Euclidean acceleration, because relative smoothness is known to be alone insufficient for obtaining accelerated rates in general [197]. Notably, [265] also developed an acceleration framework under a strengthened relative smoothness assumption, which requires strengthened bounds on divergences between three points. We further elaborate on these points in Section 2.7.3, when deriving relative Lipschitzness bounds through weaker assumptions in Lemma 13. We focus on the Euclidean setup in this paper, but we believe an analogous study of non-Euclidean setups is interesting and merits future exploration.

2.2 Preliminaries

General notation. We use $\widetilde{O}$ to hide logarithmic factors in problem regularity parameters, initial radius bounds, and target accuracies when clear from context. We denote $[n] := \{i \in \mathbb{N} \mid i \leq n\}$. Variables are in $\mathbb{R}^d$ unless otherwise noted. e_i is the i^{th} standard basis vector. Throughout the chapter, $\mathcal{X}$ (and $\mathcal{Y}$, when relevant) represent Euclidean spaces, and $\|\cdot\|$ will mean the Euclidean norm in appropriate dimension when applied to a vector, unless stated otherwise in the context. Regardless, when a norm $\|\cdot\|$ is clear from context, the dual norm is $\|\cdot\|_*$, defined as $\|x\|_* := \max_{\|y\| \leq 1} y^\top x$.

For a variable on a product space, e.g. $z \in \mathcal{X} \times \mathcal{Y}$, we refer to its blocks as $(z^{\mathsf{x}}, z^{\mathsf{y}})$ when clear from context. For a bilinear operator $\mathbf{A} : \mathcal{X} \to \mathcal{Y}^*$, unless specified otherwise, $\|\cdot\|$ will mean the (Euclidean) operator norm, i.e.

$$\|\mathbf{A}\| := \sup_{\|x\|=1} \|\mathbf{A}x\| = \sup_{\|x\|=1} \sup_{\|y\|=1} y^\top \mathbf{A} x.$$

Complexity model. Throughout the paper, we evaluate the complexity of methods by their gradient oracle complexity, and do not discuss the cost of vector operations (which typically are subsumed by the cost of the oracle). In Section 2.7, the gradient oracle returns ∇f, ∇g, or ∇h at any point; in Section 2.8 (respectively, Section 2.9), the oracle returns ∇f_i at a point for some $i \in [n]$ (respectively, ∇f_i, ∇g_i, or ∇h_i at a point for some $i \in [n]$).

Divergences. The Bregman divergence induced by differentiable, convex r is $V_x^r(x') := r(x') - r(x) - \langle \nabla r(x), x' - x \rangle$, for any $x, x' \in \mathcal{X}$. For all x, V_x^r is nonnegative and convex. Whenever we use no superscript r, we assume $r = \frac{1}{2}\|\cdot\|^2$ so that $V_x(x') = \frac{1}{2}\|x - x'\|^2$. Bregman divergences satisfy the equality

$$\langle \nabla r(w) - \nabla r(z), w - u \rangle = V_z^r(w) + V_w^r(u) - V_z^r(u). \tag{2.10}$$

We define the proximal operation in r by

$$\text{Prox}_x^r(\Phi) := \text{argmin}_{x' \in \mathcal{X}} \left\{ \langle \Phi, x' \rangle + V_x^r(x') \right\}.$$

Functions and operators. We say $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is convex-concave if its restrictions $h(\cdot, y)$ and $h(x, \cdot)$ are respectively convex and concave, for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The duality gap of a pair (x, y) is $\text{Gap}_h(x, y) := \max_{y' \in \mathcal{Y}} h(x, y') - \min_{x' \in \mathcal{X}} h(x', y)$; a saddle point is a pair $(x_\star, y_\star) \in \mathcal{X} \times \mathcal{Y}$ with zero duality gap.

We call operator $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ monotone if $\langle \Phi(z) - \Phi(z'), z - z' \rangle \geq 0$ for all $z, z' \in \mathcal{Z}$. We say $z_\star$ solves the variational inequality (VI) in Φ if $\langle \Phi(z_\star), z_\star - z \rangle \leq 0$ for all $z \in \mathcal{Z}$. We equip differentiable convex-concave h with the "gradient operator" $\Phi(x, y) := (\nabla_x h(x, y), -\nabla_y h(x, y))$. The gradient of convex f and the gradient operator of convex-concave h are both monotone (see Appendix A.9). Their VIs are respectively solved by any minimizers of f and saddle points of h.

Finally, for a function f, we call any x with $f(x) \leq f(x^\star) + \epsilon$ an *ϵ-approximate minimizer*, where $x^\star$ is the minimizing argument.

Regularity. We say function $f : \mathcal{X} \to \mathbb{R}$ is L-smooth if $\|\nabla f(x) - \nabla f(x')\|_* \leq L \|x - x'\|$ for all $x, x' \in \mathcal{X}$; if f is twice-differentiable, this is equivalent to $(x' - x)^\top \nabla^2 f(x)(x' - x) \leq L \|x' - x\|^2$ for all $x, x' \in \mathcal{X}$. We say differentiable function $f : \mathcal{X} \to \mathbb{R}$ is μ-strongly convex if $V_x^f(x') \geq \frac{\mu}{2} \|x - x'\|^2$ for all $x, x' \in \mathcal{X}$; if f is twice-differentiable, this is equivalent to $(x' - x)^\top \nabla^2 f(x)(x' - x) \geq \mu \|x' - x\|^2$ for all $x, x' \in \mathcal{X}$. We also say f is μ-strongly convex with respect to a distance-generating function r if $V_x^f(y) \geq \mu V_x^r(y)$ for all $x, y \in \mathcal{X}$.

Finally, we say operator $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ is m-strongly monotone with respect to convex $r : \mathcal{Z} \to \mathbb{R}$ if for all $z, z' \in \mathcal{Z}$,

$$\langle \Phi(z) - \Phi(z'), z - z' \rangle \geq m \langle \nabla r(z) - \nabla r(z'), z - z' \rangle = m \left(V_z^r(z') + V_{z'}^r(z) \right).$$

Convex conjugates. The (Fenchel dual) convex conjugate of a convex $f : \mathcal{X} \to \mathbb{R}$ is denoted

$$f^*(x^*) := \max_{x \in \mathcal{X}} \langle x, x^* \rangle - f(x).$$

We allow f^* to take the value ∞. We recall the following facts about convex conjugates.

Fact 1. *Let $f : \mathcal{X} \to \mathbb{R}$ be differentiable.*

1. *For all $x \in \mathcal{X}$, $\nabla f(x) \in \operatorname{argmax}_{x^* \in \mathcal{X}^*} \langle x^*, x \rangle - f^*(x^*)$.*
2. *$(f^*)^* = f$.*
3. *If f^* is differentiable, for all $x \in \mathcal{X}$, $\nabla f^*(\nabla f(x)) = x$.*
4. *If f is L-smooth, then for all $x, x' \in \mathcal{X}$,*

$$f(x') - f(x) - \langle \nabla f(x), x' - x \rangle \geq \frac{1}{2L} \|\nabla f(x') - \nabla f(x)\|^2.$$

If f is μ-strongly convex, f^* is $\frac{1}{\mu}$-smooth.

Proof. The first three items all follow from Chapter 11 of [458]. The first part of the fourth item is shown in Appendix A.1, and the second part is shown in [303]. □

For a function $f : \mathcal{X} \to \mathbb{R}$, we define the set $\mathcal{X}_f^* \subset \mathcal{X}^*$ to be the set of points realizable as a gradient, namely $\mathcal{X}_f^* := \{\nabla f(x) \mid x \in \mathcal{X}\}$. This will be come relevant in applications of Item 4 in Fact 1 throughout the paper, when ∇f is not surjective (onto $\mathcal{X}^*$).

Finally, we defer a review and several additional facts about convex conjugates to Appendix A.1.

Algorithm 1: MIRROR-PROX(z_0, T): Mirror prox [415]

Input: Distance generating r, λ-relatively Lipschitz monotone $g : \mathcal{Z} \to \mathcal{Z}^*$, initial point $z_0 \in \mathcal{Z}$;
for $0 \leq t < T$ **do**
  $w_t \leftarrow \text{Prox}^r_{z_t}(\frac{1}{\lambda} g(z_t))$;
  $z_{t+1} \leftarrow \text{Prox}^r_{z_t}(\frac{1}{\lambda} g(w_t))$;

2.3 Extragradient convergence under relative Lipschitzness

We give a brief presentation of mirror prox [415], and a convergence analysis under relative Lipschitzness. Our results also hold for dual extrapolation [420], which can be seen as a "lazy" version of mirror prox updating a state in dual space (see [98]); we defer details to Appendix A.4.

Definition 1 (Relative Lipschitzness)**.** *For convex $r : \mathcal{Z} \to \mathbb{R}$, we call operator $g : \mathcal{Z} \to \mathcal{Z}^*$ λ-relatively Lipschitz with respect to r if for every three $z, w, u \in \mathcal{Z}$,*

$$\langle g(w) - g(z), w - u \rangle \leq \lambda \left(V^r_z(w) + V^r_w(u) \right)$$

We say g is λ-relatively Lipschitz with respect to r over $\mathcal{Z}_{\text{alg}} \subseteq \mathcal{Z}$ if the above inequality holds for all $z, w, u \in \mathcal{Z}_{\text{alg}}$.

For example, we have the following bound when $g = \nabla r$, which follows directly from nonnegativity of Bregman divergences and (2.10).

Lemma 1. *Let $r : \mathcal{Z} \to \mathbb{R}$ be convex. Then, ∇r is 1-relatively Lipschitz with respect to r over $\mathcal{Z}$.*

Definition 1 can be thought of as an alternative to a natural nonlinear analog of the area convexity condition of [483] displayed below:

$$\langle g(w) - g(z), w - u \rangle \leq \lambda \left(r(z) + r(w) - r(u) - 3r\left(\frac{z + w + u}{3} \right) \right).$$

Our proposed alternative is well-suited for the standard analyses of extragradient methods such as mirror prox and dual extrapolation. For the special case of bilinear minimax problems in a matrix $\mathbf{A}$, the left hand side of Definition 1 measures the area of a triangle in a geometry induced by $\mathbf{A}$.

Relative Lipschitzness encapsulates the more standard assumptions that g is Lipschitz and r is strongly convex (Lemma 2), as well as the more recent assumptions that f is convex and relatively smooth with respect to r [64, 375] (Lemma 3). We defer proofs to Appendix A.1.

Lemma 2. *If g is L-Lipschitz and r is μ-strongly convex in $\|\cdot\|$, g is L/μ-relatively Lipschitz with respect to r.*

Lemma 3. *If f is L-relatively smooth with respect to r, i.e. $V_x^f(y) \le LV_x^r(y)$ for all x and y, then g, defined by $g(x) := \nabla f(x)$ for all x, is L-relatively Lipschitz with respect to r.*

We now give an analysis of Algorithm 1 showing the average "regret" $\langle g(w_t), w_t - u\rangle$ of iterates decays at a $O(T^{-1})$ rate. This strengthens Lemma 3.1 of [415].

Proposition 1. *The iterates $\{w_t\}$ of Algorithm 1 satisfy for all $u \in \mathcal{Z}$,*

$$\sum_{0\le t<T} \langle g(w_t), w_t - u\rangle \le \lambda V_{z_0}^r(u).$$

Proof. First-order optimality conditions of w_t, z_{t+1} with respect to u imply (see Lemma 216)

$$\begin{aligned} \frac{1}{\lambda}\langle g(z_t), w_t - z_{t+1}\rangle &\le V_{z_t}^r(z_{t+1}) - V_{w_t}^r(z_{t+1}) - V_{z_t}^r(w_t), \\ \frac{1}{\lambda}\langle g(w_t), z_{t+1} - u\rangle &\le V_{z_t}^r(u) - V_{z_{t+1}}^r(u) - V_{z_t}^r(z_{t+1}). \end{aligned} \tag{2.11}$$

Adding and manipulating gives, via relative Lipschitzness (Definition 1),

$$\begin{aligned} \frac{1}{\lambda}\langle g(w_t), w_t - u\rangle &\le V_{z_t}^r(u) - V_{z_{t+1}}^r(u) + \frac{1}{\lambda}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle - V_{w_t}^r(z_{t+1}) - V_{z_t}^r(w_t) \\ &\le V_{z_t}^r(u) - V_{z_{t+1}}^r(u). \end{aligned} \tag{2.12}$$

Finally, summing and telescoping (2.12) yields the desired conclusion. □

We briefly comment on how to use Proposition 1 to approximately solve convex-concave games in a function $f(x, y)$. By applying convexity and concavity appropriately to the regret guarantee (and dividing by T, the iteration count), one can replace the left hand side of the guarantee with the duality gap of an average point $\bar{w}$ against a point u, namely $f(w^{\mathsf{x}}, u^{\mathsf{y}}) - f(u^{\mathsf{x}}, w^{\mathsf{y}})$. By maximizing the right hand side over u, this can be converted into an overall duality gap guarantee. For some of our applications in following sections, u will be some fixed point (rather than a best response) and the regret statement will be used in a more direct manner to prove guarantees.

2.4 Acceleration via relative Lipschitzness

We show that directly applying Algorithm 1 to the optimization problem (2.1) recovers an accelerated rate for first-order convex function minimization (for simplicity, we focus on the ℓ_2 norm here; our methods extend to general norms, discussed in Appendix A.5). Our main technical result, Lemma 4, shows the gradient operator of (2.1) is relatively Lipschitz in the natural regularizer induced by f, which combined with Proposition 1 gives our main result, Theorem 4. Crucially, our method regularizes the dual variable with f^*, the Fenchel dual of f, which we show admits efficient implementation, allowing us to obtain our improved bound on the relative Lipschitzness parameter.

Lemma 4 (Relative Lipschitzness for the Fenchel game)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and μ-strongly convex in the Euclidean norm $\|\cdot\|_2$. Let $g(x, y) = (y, \nabla f^*(y) - x)$ be the gradient operator of the convex-concave problem* (2.1)*, and define the distance-generating function $r(x, y) := \frac{\mu}{2}\|x\|_2^2 + f^*(y)$. Then, g is $1 + \sqrt{\frac{L}{\mu}}$-relatively Lipschitz with respect to r.*

Proof. Consider three points $z = (z^{\mathsf{x}}, z^{\mathsf{y}}),\ w = (w^{\mathsf{x}}, w^{\mathsf{y}}),\ u = (u^{\mathsf{x}}, u^{\mathsf{y}})$. By direct calculation,

$$\langle g(w) - g(z), w - u\rangle = \langle w^{\mathsf{y}} - z^{\mathsf{y}}, w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle -w^{\mathsf{x}} + z^{\mathsf{x}} + \nabla f^*(w^{\mathsf{y}}) - \nabla f^*(z^{\mathsf{y}}), w^{\mathsf{y}} - u^{\mathsf{y}}\rangle\,. \tag{2.13}$$

By Cauchy-Schwarz and L^{-1}-strong convexity of f^* (cf. Lemma 214) respectively, we have

$$\begin{aligned}\langle w^{\mathsf{y}} - z^{\mathsf{y}}, w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle z^{\mathsf{x}} - w^{\mathsf{x}}, w^{\mathsf{y}} - u^{\mathsf{y}}\rangle &\le \|w^{\mathsf{y}} - z^{\mathsf{y}}\|_2 \|w^{\mathsf{x}} - u^{\mathsf{x}}\|_2 + \|z^{\mathsf{x}} - w^{\mathsf{x}}\|_2 \|w^{\mathsf{y}} - u^{\mathsf{y}}\|_2 \\ &\le \sqrt{\frac{L}{\mu}}\left(\frac{\mu}{2}\|w^{\mathsf{x}} - z^{\mathsf{x}}\|_2^2 + \frac{\mu}{2}\|w^{\mathsf{x}} - u^{\mathsf{x}}\|_2^2 + \frac{1}{2L}\|w^{\mathsf{y}} - z^{\mathsf{y}}\|_2^2 + \frac{1}{2L}\|w^{\mathsf{y}} - u^{\mathsf{y}}\|_2^2\right) \\ &\le \sqrt{\frac{L}{\mu}}\left(V_z^r(w) + V_w^r(u)\right).\end{aligned} \tag{2.14}$$

The second line used Young's inequality twice. Furthermore, by convexity of f^* from z^{y} to u^{y},

$$\begin{aligned}&\langle \nabla f^*(w^{\mathsf{y}}) - \nabla f^*(z^{\mathsf{y}}), w^{\mathsf{y}} - u^{\mathsf{y}}\rangle \\ &= \langle \nabla f^*(z^{\mathsf{y}}), u^{\mathsf{y}} - z^{\mathsf{y}}\rangle - \langle \nabla f^*(w^{\mathsf{y}}), u^{\mathsf{y}} - w^{\mathsf{y}}\rangle - \langle \nabla f^*(z^{\mathsf{y}}), w^{\mathsf{y}} - z^{\mathsf{y}}\rangle \\ &\le f^*(u^{\mathsf{y}}) - f^*(z^{\mathsf{y}}) - \langle \nabla f^*(w^{\mathsf{y}}), u^{\mathsf{y}} - w^{\mathsf{y}}\rangle - \langle \nabla f^*(z^{\mathsf{y}}), w^{\mathsf{y}} - z^{\mathsf{y}}\rangle \\ &= V_{z^{\mathsf{y}}}^{f^*}(w^{\mathsf{y}}) + V_{w^{\mathsf{y}}}^{f^*}(u^{\mathsf{y}}) \le V_z^r(w) + V_w^r(u).\end{aligned} \tag{2.15}$$

The last inequality used separability of r and nonnegativity of divergences. Summing the bounds (2.14) and (2.15) and recalling (2.13) yields the conclusion, where we use Definition 1. □

We also state a convenient fact about the form our iterates take.

Lemma 5. *In the setting of Lemma 4, let $z_t = (x_t, y_t)$, $w_t = (x_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})$ be iterates produced by running Algorithm 1 on the pair g, r. Suppose $y_0 = \nabla f(v_0)$ for some v_0. Then, $y_{t+\frac{1}{2}}$ and y_{t+1} can be recursively expressed as $y_{t+\frac{1}{2}} = \nabla f(v_{t+\frac{1}{2}})$, $y_{t+1} = \nabla f(v_{t+1})$, for*

$$v_{t+\frac{1}{2}} \leftarrow v_t + \frac{1}{\lambda}(x_t - v_t),\ v_{t+1} \leftarrow v_t + \frac{1}{\lambda}\left(x_{t+\frac{1}{2}} - v_{t+\frac{1}{2}}\right).$$

Proof. We prove this inductively; consider some iteration t. Assuming $y_t = \nabla f(v_t)$, by definition

$$\begin{aligned}y_{t+\frac{1}{2}} &= \operatorname{argmin}_y \left\{\left\langle \frac{1}{\lambda}\left(\nabla f^*(y_t) - x_t\right), y\right\rangle + V_{y_t}^{f^*}(y)\right\} \\ &= \operatorname{argmax}_y \left\{\left\langle \frac{1}{\lambda}(x_t - v_t) + v_t, y\right\rangle - f^*(y)\right\} = \nabla f\left(v_t + \frac{1}{\lambda}(x_t - v_t)\right).\end{aligned}$$

Here, we used standard facts about convex conjugates (see Lemma 212). A similar argument shows that we can compute implicitly $y_{t+1} = \nabla f(v_t + \frac{1}{\lambda}(x_{t+\frac{1}{2}} - v_{t+\frac{1}{2}}))$. □

We now prove Theorem 4, i.e. that we can halve function error in $O\left(\sqrt{\frac{L}{\mu}}\right)$ iterations of Algorithm 1. Simply iterating Theorem 4 yields a linear rate of convergence for smooth, strongly convex functions, yielding an ϵ-approximate minimizer in $O\left(\sqrt{\frac{L}{\mu}} \log \frac{f(x_0)-f(x^*)}{\epsilon}\right)$ iterations.

Theorem 4. *In the setting of Lemma 4, run $T \geq 4\lambda$ iterations of Algorithm 1 initialized at $z_0 = (x_0, \nabla f(x_0))$ on the pair g, r with $\lambda = 1 + \sqrt{\frac{L}{\mu}}$, and define*

$$\bar{v} = \frac{1}{T} \sum_{0 \leq t < T} v_{t+\frac{1}{2}} \text{ where } w_t = \left(x_{t+\frac{1}{2}}, \nabla f(v_{t+\frac{1}{2}})\right).$$

Then we have $f(\bar{v}) - f(x^*) \leq \frac{1}{2}(f(x_0) - f(x^*))$, where x^* minimizes f.

Proof. First, we remark that this form of w_t follows from Lemma 5, and correctness of λ follows from Lemma 4. By an application of Proposition 1, letting $u = (x^*, \nabla f(x^*))$,

$$\begin{aligned}\frac{1}{T} \sum_{0 \leq t < T} \langle g(w_t), w_t - u \rangle &\leq \frac{\lambda}{T} \cdot V_{z_0}^r(u) \leq \frac{1}{4}\left(\frac{\mu}{2}\|x_0 - x^*\|_2^2 + V_{\nabla f(x_0)}^{f^*}(\nabla f(x^*))\right) \\ &= \frac{1}{4}\left(\frac{\mu}{2}\|x_0 - x^*\|_2^2 + f(x_0) - f(x^*)\right) \leq \frac{1}{2}\left(f(x_0) - f(x^*)\right).\end{aligned}$$

The second line used the definition of divergence in f^* (see Lemma 213) and strong convexity of f, which implies $f(x_0) \geq f(x^*) + \frac{\mu}{2}\|x_0 - x^*\|_2^2$. Moreover, by the definition of g and $\nabla f(x^*) = 0$,

$$\begin{aligned}\frac{1}{T} \sum_{0 \leq t < T} \langle g(w_t), w_t - u \rangle &= \frac{1}{T} \sum_{0 \leq t < T} \left\langle \nabla f(v_{t+\frac{1}{2}}), x_{t+\frac{1}{2}} - x^* \right\rangle + \left\langle v_{t+\frac{1}{2}} - x_{t+\frac{1}{2}}, \nabla f(v_{t+\frac{1}{2}}) \right\rangle \\ &\geq \frac{1}{T} \sum_{0 \leq t < T} f(v_{t+\frac{1}{2}}) - f(x^*) \geq f(\bar{v}) - f(x^*).\end{aligned}$$

The last line used convexity twice. Combining these two derivations yields the conclusion. □

For convenience, we state the full algorithm of Theorem 4 as Algorithm 2.

In Appendix A.5, we give an alternative proof of acceleration leveraging relative Lipschitzness, as well as a variant of extragradient methods suited for strongly monotone operators (cf. Appendix A.4), by applying these tools to the saddle point problem (to be contrasted with (2.1))

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^d} \frac{\mu}{2}\|x\|_2^2 + \langle y, x \rangle - h^*(y), \text{ where } h(x) := f(x) - \frac{\mu}{2}\|x\|_2^2.$$

This alternative proof strategy readily generalizes the accelerated rate of Theorem 4 to general norms. While the rates attained in Appendix A.5 are slightly less sharp (losing a $\frac{L}{\mu}$ factor in the

Algorithm 2: EG-ACCEL(x_0, ϵ): Extragradient accelerated smooth minimization

Input: $x_0 \in \mathbb{R}^d$, f L-smooth and μ-strongly convex in $\|\cdot\|_2$, and $\epsilon_0 \geq f(x_0) - f(x^*)$;
Output: ϵ-approximate minimizer of f;
$\lambda \leftarrow 1 + \sqrt{L/\mu}$, $x^{(0)} \leftarrow x_0$, $T \leftarrow 4\lceil\lambda\rceil$, $K \leftarrow \lceil \log_2 \frac{\epsilon_0}{\epsilon} \rceil$;
for $0 \leq k < K$ **do**
  $x_0 \leftarrow x^{(k)}$, $v_0 \leftarrow x_0$;
  for $0 \leq t < T$ **do**
    $x_{t+\frac{1}{2}} \leftarrow x_t - \frac{1}{\mu\lambda}\nabla f(v_t)$ and $v_{t+\frac{1}{2}} \leftarrow v_t + \frac{1}{\lambda}(x_t - v_t)$;
    $x_{t+1} \leftarrow x_t - \frac{1}{\mu\lambda}\nabla f(v_{t+\frac{1}{2}})$ and $v_{t+1} \leftarrow v_t + \frac{1}{\lambda}(x_{t+\frac{1}{2}} - v_{t+\frac{1}{2}})$;
  $x^{(k+1)} \leftarrow \frac{1}{T}\sum_{0\leq t<T} v_{t+\frac{1}{2}}$;
Return: $x^{(K)}$;

logarithm) when compared to Theorem 4, the analysis is arguably simpler. This is in the sense that Appendix A.5 shows a potential function decreases at a linear rate in every iteration, rather than requiring $O(\sqrt{L/\mu})$ iterations to halve it.

2.5 Area convexity rates for box-simplex games via relative Lipschitzness

In this section, we show that a local variant of Definition 1 recovers the improved convergence rate achieved by [483] for box-constrained ℓ_∞-regression, and more generally box-simplex bilinear games. Specifically, we will use the following result, a simple extension to Proposition 1 which states that relative Lipschitzness only must hold with respect to triples of algorithm iterates.

Corollary 1. *Suppose Algorithm 1 is run with a monotone operator g and a distance generating r satisfying, for all iterations t,*

$$\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle \leq \lambda \left(V_{z_t}^r(w_t) + V_{w_t}(z_{t+1})\right). \tag{2.16}$$

Then, the conclusion of Proposition 1 holds.

Proof. Observe that the only applications of relative Lipschitzness in the proof of Proposition 1 are of the form (2.16) (namely, in (2.12)). Thus, the same conclusion still holds. □

In other words, letting $\mathcal{Z}_{\text{alg}}$ be a superset of all the iterates of the algorithm, it suffices for relative Lipschitzness to hold only over $\mathcal{Z}_{\text{alg}}$ (see Definition 1). We use Corollary 1 to give an alternative algorithm and analysis recovering the rates implied by the use of area convexity in [483], for box-simplex games, which we now define.

Problem 1 (Box-simplex game)**.** *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix and let $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ be vectors. The associated box-simplex game, and its induced monotone operator g, are*

$$\min_{x \in [-1,1]^n} \max_{y \in \Delta^m} f(x,y) := y^\top \mathbf{A} x - \langle b, y \rangle + \langle c, x \rangle \,, \; g(x,y) := \begin{pmatrix} \mathbf{A}^\top y + c \\ b - \mathbf{A} x \end{pmatrix}. \tag{2.17}$$

Here, $\Delta^m := \{y \in \mathbb{R}^m_{\geq 0} : \sum_{i \in [m]} y_i = 1\}$ is the nonnegative probability simplex in m dimensions.

By a simple reduction that at most doubles the size of the input (stacking $\mathbf{A}$, b with negated copies, cf. Section 3.1 of [486]), Problem 1 is a generalization of the box-constrained ℓ_∞-regression problem

$$\min_{x \in [-1,1]^m} \|\mathbf{A} x - b\|_\infty .$$

The work of [483] proposed a variant of extragradient algorithms, based on taking primal-dual proximal steps in the following regularizer:[6]

$$r(x,y) := y^\top |\mathbf{A}|(x^2) + 10 \|\mathbf{A}\|_{\infty \to \infty} \sum_{i \in [m]} y_i \log y_i. \tag{2.18}$$

Here, $|\mathbf{A}|$ is the entrywise absolute value of $\mathbf{A}$. The convergence rate of this algorithm was proven in [483] via an analysis based on "area convexity" of the pair (g, r), which required a somewhat sophisticated proof based on solving a partial differential equation over a triangle. We now show that the same rate can be obtained by the extragradient algorithms of [415, 420], and analyzed via local relative Lipschitzness (2.16).[7] We first make the following simplication without loss of generality.

Lemma 6. *For all $x \in [-1,1]^n$ the value of $\max_{y \in \Delta^m} f(x,y)$ in (2.17) is unchanged if we remove all coordinates of b with $b_i \geq \min_{i^* \in [m]} b_{i^*} + 2\|\mathbf{A}\|_{\infty \to \infty}$, and the corresponding rows of $\mathbf{A}$. Therefore, in designing an algorithm to solve (2.17) to additive error with linear pre-processing it suffices to assume that $b_i \in [0, 2\|\mathbf{A}\|_{\infty \to \infty}]$ for all $i \in [m]$.*

Proof. For any $x \in [-1,1]^n$, letting $i^* \in \operatorname{argmin}_{i \in [m]} b_i$ we have

$$\max_{y \in \Delta^m} y^\top (\mathbf{A} x - b) = \max_{i \in [m]} [\mathbf{A} x - b]_i \geq -\|\mathbf{A}\|_{\infty \to \infty} \|x\|_\infty - \min_{i^* \in [m]} b_i \geq -\|\mathbf{A}\|_{\infty \to \infty} - b_{i^*}.$$

However, $[\mathbf{A} x - b]_i \leq \|\mathbf{A}\|_{\infty \to \infty} - b_i$ for all $i \in [m]$. Consequently, any coordinate $i \in [m]$ that satisfies $b_i \geq b_{i^*} + 2\|\mathbf{A}\|_{\infty \to \infty}$ has $[\mathbf{A} x - b]_i \leq [\mathbf{A} x - b]_{i_*}$ and the value of $\max_{y \in \Delta^m} f(x,y)$ is unchanged if this entry of b_i and the corresponding row of $\mathbf{A}$ is removed. Further, note that $\langle y, \mathbf{1} \rangle$ is a constant for all $y \in \Delta^m$. Consequently, in linear time we can remove all the coordinates i with

[6] We let $\|\mathbf{A}\|_{\infty \to \infty} := \sup_{\|x\|_\infty = 1} \|\mathbf{A} x\|_\infty$, i.e. the ℓ_∞ operator norm of $\mathbf{A}$ or max ℓ_1 norm of any row.

[7] Although our analysis suffices to recover the rate of [483] for ℓ_∞ regression, the analysis of [483] is in some sense more robust (and possibly) more broadly applicable than ours, as it does not need to reason directly about how much the iterates vary in a step. Understanding or closing this gap is an interesting open problem.

$b_i \geq \min_{i^* \in [m]} b_{i^*} + 2\|\mathbf{A}\|_{\infty\to\infty}$ and shift all the coordinates by an additive constant so that the minimum coordinate of a remaining b_i is 0 without affecting additive error of any x. □

We now prove our main result regarding the use of mirror prox to solve box-simplex games, using the area convex regularizer analyzed (with a slightly different algorithm) in [483].

Theorem 5. *Assume the preprocessing of Lemma 6 so that $b \in [0, 2\|\mathbf{A}\|_{\infty\to\infty}]^m$. Consider running Algorithm 1 or Algorithm 65 on the operator in (2.17) with $\lambda = 3$, using the regularizer in (2.18). The resulting iterates satisfy (2.16), and thus satisfy the conclusion of Proposition 1.*

Proof. Fix a particular iteration t. We first claim that the simplex variables w_t^y and z_{t+1}^y obey the following multiplicative stability property: entrywise,

$$w_t^y, z_{t+1}^y \in \left[\frac{1}{2} z_t^y, 2 z_t^y\right]. \tag{2.19}$$

We will give the proof for w_t^y as the proof for z_{t+1}^y follows from the same reasoning. Recall that

$$w_t = \text{argmin}_{w \in \Delta^n \times [-1,1]^m} \left\{ \left\langle \frac{1}{\lambda} g(z_t), w \right\rangle + V_{z_t}^r(w) \right\},$$

and therefore, defining $(x)^2$ and $(z_t^x)^2$ as the entrywise square of these vectors,

$$w_t^y = \text{argmin}_{y \in \Delta^m} \langle \gamma_t^y, y \rangle + 10\|\mathbf{A}\|_{\infty\to\infty} \sum_{i\in[m]} y_i \log \frac{y_i}{[z_t^y]_i} \text{where } \gamma_t^y := \frac{1}{\lambda}(b - \mathbf{A} z_t^x) + |\mathbf{A}| \left[(x)^2 - (z_t^x)^2\right].$$

Consequently, applying log and exp entrywise we have

$$w_t^y \propto \exp\left(\log z_t^y - \frac{1}{10\|\mathbf{A}\|_{\infty\to\infty}} \gamma_t^y\right).$$

This implies the desired (2.19), where we use that $\|\gamma_t^y\|_\infty \leq 3\|\mathbf{A}\|_{\infty\to\infty}$, and $\exp(0.6) \leq 2$. Next, we have by a straightforward calculation (Lemma 3.4, [483] or Lemma 6, [293]) that

$$\nabla^2 r(x, y) \succeq \begin{pmatrix} \mathbf{diag}\left(|\mathbf{A}_{:j}|^\top y\right) & 0 \\ 0 & \|\mathbf{A}\|_{\infty\to\infty} \mathbf{diag}\left(\frac{1}{y_i}\right) \end{pmatrix}. \tag{2.20}$$

By expanding the definition of Bregman divergence, we have

$$V_{z_t}^r(w_t) = \int_0^1 \int_0^\alpha \|w_t - z_t\|^2_{\nabla^2 r(z_t + \beta(w_t - z_t))} \, d\beta d\alpha.$$

Fix some $\beta \in [0, 1]$, and let $z_\beta := z_t + \beta(w_t - z_t)$. Since the coordinates of z_β also satisfy the stability

property (2.19), by the lower bound of (D.12), we have

$$\begin{aligned}\|w_t - z_t\|^2_{\nabla^2 r(z_\beta)} &\geq \sum_{i\in[m],j\in[n]} |\mathbf{A}_{ij}| \left([z^y_\beta]_i \, [w^x_t - z^x_t]^2_j + \frac{1}{[z^y_\beta]_i} \, [w^y_t - z^y_t]^2_i \right) \\ &\geq \frac{1}{2} \sum_{i\in[m],j\in[n]} |\mathbf{A}_{ij}| \left([z^y_t]_i \, [w^x_t - z^x_t]^2_j + \frac{1}{[z^y_t]_i} \, [w^y_t - z^y_t]^2_i \right).\end{aligned}$$

By using a similar calculation to lower bound $V^r_{w_t}(z_{t+1})$, we have by Young's inequality the desired

$$\begin{aligned}V^r_{z_t}(w_t) + V^r_{w_t}(z_{t+1}) &\geq \frac{1}{4} \sum_{i\in[m],j\in[n]} |\mathbf{A}_{ij}| \left([z^y_t]_i \, [w^x_t - z^x_t]^2_j + \frac{1}{[z^y_t]_i} \, [w^y_t - z^y_t]^2_i \right) \\ &+ \frac{1}{4} \sum_{i\in[m],j\in[n]} |\mathbf{A}_{ij}| \left([z^y_t]_i \left[w^x_t - z^x_{t+1}\right]^2_j + \frac{1}{[z^y_t]_i} \left[w^y_t - z^y_{t+1}\right]^2_i \right) \\ &\geq \frac{1}{3} \sum_{i\in[m],j\in[n]} \mathbf{A}_{ij} \left([w^y_t - z^y_t]_i \left[w^x_t - z^x_{t+1}\right]_j - \left[w^y_t - z^y_{t+1}\right]_i [w^x_t - z^x_t]_j \right) \\ &= \frac{1}{\lambda} \left\langle g(w_t) - g(z_t), w_t - z_{t+1} \right\rangle.\end{aligned}$$

□

The range of the regularizer r is bounded by $O(\|\mathbf{A}\|_{\infty\to\infty} \log m)$, and hence the iteration complexity to find an ϵ additively-approximate solution to the box-simplex game is $O(\frac{\|\mathbf{A}\|_{\infty\to\infty} \log m}{\epsilon})$. Finally, we comment that the iteration complexity of solving the subproblems required by extragradient methods in the regularizer r to sufficiently high accuracy is logarithmically bounded in problem parameters via a simple alternating minimization scheme proposed by [483]. Here, we note that the error guarantee e.g. Proposition 1 is robust up to constant factors to solving each subproblem to ϵ additive accuracy, and appropriately using approximate optimality conditions (for an example of this straightforward extension, see Corollary 1 of [293]).

2.6 Randomized coordinate acceleration via expected relative Lipschitzness

We show relative Lipschitzness can compose with randomization. Specifically, we adapt Algorithm 2 to coordinate smoothness, recovering the accelerated rate first obtained in [553, 425]. We recall f is L_i-coordinate-smooth if its coordinate restriction is smooth, i.e. $|\nabla_i f(x + ce_i) - \nabla_i f(x)| \leq L_i|c|$ $\forall x \in \mathcal{X},\ c \in \mathbb{R}$; for twice-differentiable coordinate smooth f, $\nabla^2_{ii} f(x) \leq L_i$.

Along the way, we build a framework for randomized extragradient methods via "local variance reduction" in Proposition 2. In particular, we demonstrate how for separable domains our technique can yield $O(T^{-1})$ rates for stochastic extragradient algorithms, bypassing a variance barrier

encountered by prior methods [302]. Throughout, let $f : \mathbb{R}^d \to \mathbb{R}$ be L_i-smooth in coordinate i, and μ-strongly convex in $\|\cdot\|_2$, and define the distance generating function $r(x, y) = \frac{\mu}{2}\|x\|_2^2 + f^*(y)$.

Our approach modifies that of Section 2.4 in the following ways. First, our iterates are defined via stochastic estimators which "share randomness" (use the same coordinate in both updates). Concretely, fix some iterate $z_t = (x_t, \nabla f(v_t))$. For a distribution $\{p_i\}_{i\in[d]}$, sample $i \sim p_i$ and let

$$\begin{aligned}
&g_i(z_t) := \left(\frac{1}{p_i}\nabla_i f(v_t), v_t - x_t\right),\ w_t^{(i)} = \left(x_{t+\frac{1}{2}}^{(i)}, \nabla f(v_{t+\frac{1}{2}})\right) := \mathrm{Prox}_{z_t}^r\left(\frac{1}{\lambda}g_i(z_t)\right),\\
&g_i(w_t^{(i)}) := \left(\frac{1}{p_i}\nabla_i f(v_{t+\frac{1}{2}}), v_{t+\frac{1}{2}} - \left(x_t + \frac{1}{p_i}\Delta_t^{(i)}\right)\right) \text{ for } \Delta_t^{(i)} := x_{t+\frac{1}{2}}^{(i)} - x_t,\\
&z_{t+1}^{(i)} = \left(x_{t+1}^{(i)}, \nabla f(v_{t+1}^{(i)})\right) := \mathrm{Prox}_{z_t}^r\left(\frac{1}{\lambda}g_i(w_t^{(i)})\right).
\end{aligned} \tag{2.21}$$

By observation, $g_i(z_t)$ is unbiased for $g(z_t)$; however, the same cannot be said for $g_i(w_t^{(i)})$, as the random coordinate was used in the definition of $w_t^{(i)}$. Nonetheless, examining the proof of Proposition 1, we see that the conclusion

$$\langle g(\bar{w}_t), \bar{w}_t - u\rangle \le V_{z_t}^r(u) - \mathbb{E}\left[V_{z_{t+1}^{(i)}}^r(u)\right]$$

still holds for some point $\bar{w}_t$, as long as

$$\begin{aligned}
\mathbb{E}\left[\left\langle g_i(w_t^{(i)}), w_t^{(i)} - u\right\rangle\right] &= \langle g(\bar{w}_t), \bar{w}_t - u\rangle,\\
\mathbb{E}\left[\left\langle g_i(w_t^{(i)}) - g_i\left(z_t\right), w_t^{(i)} - z_{t+1}^{(i)}\right\rangle\right] &\le \lambda\mathbb{E}\left[V_{z_t}^r(w_t^{(i)}) + V_{w_t^{(i)}}^r(z_{t+1}^{(i)})\right].
\end{aligned} \tag{2.22}$$

We make this concrete in the following claim, a generalization of Proposition 1 which handles randomized operator estimates as well as an expected variant of relative Lipschitzness. We remark that as in Corollary 1, the second condition in (2.22) only requires relative Lipschitzness to hold for the iterates of the algorithm, rather than globally.

Proposition 2. *Suppose in every iteration of Algorithm 1, steps are conducted with respect to randomized gradient operators* $\left\{g_i(z_t), g_i(w_t^{(i)})\right\}$ *satisfying* (2.22) *for some* $\{\bar{w}_t\}$. *Then, for all* $u \in \mathcal{Z}$,

$$\mathbb{E}\left[\sum_{0\le t<T}\langle g\left(\bar{w}_t\right), \bar{w}_t - u\rangle\right] \le \lambda V_{z_0}^r(u).$$

Proof. The proof follows identically to that of Proposition 1, where we iterate taking expectations over (2.12), each time applying the two conditions in (2.22). □

For the rest of this section, we overload g_i to mean the choices used in (2.21). This choice is motivated via the following two properties, required by (2.22) (and shown in Appendix A.6).

Lemma 7. *Let* $\bar{w}_t := (x_t + \sum_{i\in[d]} \Delta_t^{(i)}, \nabla f(v_{t+\frac{1}{2}}))$. *Then* $\forall u$, *taking expectations over iteration* t,

$$\mathbb{E}\left[\left\langle g_i(w_t^{(i)}), w_t^{(i)} - u\right\rangle\right] = \langle g(\bar{w}_t), \bar{w}_t - u\rangle .$$

Lemma 8 (Expected relative Lipschitzness)**.** *Let* $\lambda = 1 + S_{1/2}/\sqrt{\mu}$, *where* $S_{1/2} := \sum_{i\in[d]} \sqrt{L_i}$. *Then, for the iterates* (2.21) *with* $p_i = \sqrt{L_i}/S_{1/2}$, *taking expectations over iteration* t,

$$\mathbb{E}\left[\left\langle g_i(w_t^{(i)}) - g_i\left(z_t\right), w_t^{(i)} - z_{t+1}^{(i)}\right\rangle\right] \le \lambda \mathbb{E}\left[V_{z_t}^r(w_t^{(i)}) + V_{w_t^{(i)}}^r(z_{t+1}^{(i)})\right].$$

Crucially, our proof of these results uses the fact that our randomized gradient estimators are 1-sparse in the x component, and the fact that we "shared randomness" in the definition of the gradient estimators. Moreover, our iterates are efficiently implementable, under the "generalized partial derivative oracle" of prior work [347, 553, 425], which computes $\nabla_i f(ax + by)$ for $x, y \in \mathbb{R}^d$ and $a, b \in \mathbb{R}$. In many settings of interest, these oracles can be implemented with a dimension-independent runtime; we defer a discussion to previous references.

Lemma 9 (Iterate maintenance)**.** *We can implement each iteration of Algorithm 1 using two generalized partial derivative oracle queries and constant additional work.*

We defer a formal statement to Appendix A.6, as Lemma 222. Combining Lemma 7 and Lemma 8, (2.22) is satisfied with $\lambda = 1 + S_{1/2}/\sqrt{\mu}$. Finally, all of these pieces directly imply the following, via the proof of Theorem 4 and iterating expectations. We give our full method as Algorithm 1.

Theorem 6 (Coordinate acceleration)**.** *Algorithm 1 produces an* ϵ*-approximate minimizer of* f *in*

$$O\left(\sum_{i\in[d]} \sqrt{\frac{L_i}{\mu}} \log\left(\frac{f(x_0) - f(x^*)}{\epsilon}\right)\right) \text{ iterations in expectation,}$$

with iteration complexity given by Lemma 9.

Proof. This follows from the proof of Theorem 4, using Proposition 2 in place of Proposition 1. □

2.7 Separable minimax optimization

In this section, we provide efficient algorithms for computing an approximate saddle point of the following separable minimax optimization problem:

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} F_{\text{mm}}(x, y) \text{ for } F_{\text{mm}} := f(x) + h(x, y) - g(y). \tag{2.23}$$

Here and throughout this section $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{Y} \to \mathbb{R}$ are differentiable, convex functions and $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a differentiable, convex-concave function. For the remainder, we focus on algorithms for solving the following regularized formulation of (2.23):

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} F_{\text{mm-reg}}(x, y) \text{ for } F_{\text{mm-reg}}(x, y) := f(x) + h(x, y) - g(y) + \frac{\mu^{\mathsf{x}}}{2} \|x\|^2 - \frac{\mu^{\mathsf{y}}}{2} \|y\|^2 . \tag{2.24}$$

To instead solve an instance of (2.23) where f is μ^{x}-strongly convex and g is μ^{y}-strongly convex, we may instead equivalently solve (2.24) by reparameterizing $f \leftarrow f - \frac{\mu^{\mathsf{x}}}{2} \|\cdot\|^2$, $g \leftarrow g - \frac{\mu^{\mathsf{y}}}{2} \|\cdot\|^2$. As it is notationally convenient for our analysis, we focus on solving the problem (2.24) and then give the results for (2.23) at the end of this section in Corollary 3.

In designing methods for solving (2.24) we make the following additional regularity assumptions.

Assumption 1 (Minimax regularity)**.** *We assume the following about* (2.24)*.*

1. *f is L^{x}-smooth and g is L^{y}-smooth.*
2. *h has the following blockwise-smoothness properties: for all $u, v \in \mathcal{X} \times \mathcal{Y}$,*

$$\begin{aligned} \|\nabla_x h(u) - \nabla_x h(v)\| &\le \Lambda^{\mathsf{xx}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda^{\mathsf{xy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\| , \\ \|\nabla_y h(u) - \nabla_y h(v)\| &\le \Lambda^{\mathsf{xy}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda^{\mathsf{yy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\| . \end{aligned} \tag{2.25}$$

Note that when h is twice-differentiable, (2.51) equates to everywhere operator norm bounds on blocks of $\nabla^2 h$. Namely, for all $w \in \mathcal{X} \times \mathcal{Y}$,

$$\left\|\nabla^2_{xx} h(w)\right\|_{\text{op}} \le \Lambda^{\mathsf{xx}}, \ \left\|\nabla^2_{xy} h(w)\right\|_{\text{op}} \le \Lambda^{\mathsf{xy}}, \text{ and } \left\|\nabla^2_{yy} h(w)\right\|_{\text{op}} \le \Lambda^{\mathsf{yy}}.$$

In the particular case when $h(x, y) = y^\top \mathbf{A} x - b^\top y + c^\top x$ is bilinear, clearly $\Lambda^{\mathsf{xx}} = \Lambda^{\mathsf{yy}} = 0$ (as remarked in the introduction). In this case, we may then set $\Lambda^{\mathsf{xy}} := \|\mathbf{A}\|_{\text{op}}$.

The remainder of this section is organized as follows.

1. In Section 2.7.1, we state a primal-dual formulation of (2.24) which we will apply our methods to, and prove that its solution yields a solution to (2.24).
2. In Section 2.7.2, we give our algorithm and prove it is efficiently implementable.
3. In Section 2.7.3, we prove the convergence rate of our algorithm.
4. In Section 2.7.4, we state and prove our main result, Theorem 25.

2.7.1 Setup

To solve (2.24), we will instead find a saddle point to the expanded primal-dual function

$$F_{\text{mm-pd}}(z) := \left\langle z^{\mathsf{f}^*}, z^{\mathsf{x}} \right\rangle - \left\langle z^{\mathsf{g}^*}, z^{\mathsf{y}} \right\rangle + \frac{\mu^{\mathsf{x}}}{2} \left\| z^{\mathsf{x}} \right\|^2 - \frac{\mu^{\mathsf{y}}}{2} \left\| z^{\mathsf{y}} \right\|^2 + h(z^{\mathsf{x}}, z^{\mathsf{y}}) - f^*(z^{\mathsf{f}^*}) + g^*(z^{\mathsf{g}^*}). \quad (2.26)$$

We denote the domain of $F_{\text{mm-pd}}$ by $\mathcal{Z} := \mathcal{X} \times \mathcal{Y} \times \mathcal{X}^* \times \mathcal{Y}^*$. For $z \in \mathcal{Z}$, we refer to its blocks by $(z^{\mathsf{x}}, z^{\mathsf{y}}, z^{\mathsf{f}^*}, z^{\mathsf{g}^*})$. The primal-dual function $F_{\text{mm-pd}}$ is related to $F_{\text{mm-reg}}$ in the following way.

Lemma 10. *Let $z_\star$ be the saddle point to* (2.26)*. Then, $(z^{\mathsf{x}}_\star, z^{\mathsf{y}}_\star)$ is a saddle point to* (2.24)*.*

Proof. By performing the maximization over z^{f^*} and minimization over z^{g^*}, we see that the problem of computing a saddle point to the objective in (2.26) is equivalent to

$$\min_{z^{\mathsf{x}} \in \mathcal{X}} \max_{z^{\mathsf{y}} \in \mathcal{Y}} \frac{\mu^{\mathsf{x}}}{2} \left\| z^{\mathsf{x}} \right\|^2 - \frac{\mu^{\mathsf{y}}}{2} \left\| z^{\mathsf{y}} \right\|^2 + h(z^{\mathsf{x}}, z^{\mathsf{y}}) + \left(\max_{z^{\mathsf{f}^*} \in \mathcal{X}^*} \left\langle z^{\mathsf{f}^*}, z^{\mathsf{x}} \right\rangle - f^*(z^{\mathsf{f}^*}) \right) - \left(\max_{z^{\mathsf{g}^*} \in \mathcal{Y}^*} \left\langle z^{\mathsf{g}^*}, z^{\mathsf{y}} \right\rangle - g^*(z^{\mathsf{g}^*}) \right).$$

By Item 2 in Fact 1, this is the same as (2.24). □

We next define Φ, the gradient operator of $F_{\text{mm-pd}}$. Before doing so, it will be convenient to define $r : \mathcal{Z} \to \mathbb{R}$, which combines the (unsigned) separable components of $F_{\text{mm-pd}}$:

$$r(z) := \frac{\mu^{\mathsf{x}}}{2} \left\| z^{\mathsf{x}} \right\|^2 + \frac{\mu^{\mathsf{y}}}{2} \left\| z^{\mathsf{y}} \right\|^2 + f^*(z^{\mathsf{f}^*}) + g^*(z^{\mathsf{g}^*}). \quad (2.27)$$

The function r will also serve as a regularizer in our algorithm. With this definition, we decompose Φ into three parts, roughly corresponding to the contribution from r, the contributions from the bilinear portion of the primal-dual representations of f and g, and the contribution from h. In particular, we define

$$\begin{aligned} \Phi^r(z) &:= \nabla r(z) = \left(\mu^{\mathsf{x}} z^{\mathsf{x}}, \mu^{\mathsf{y}} z^{\mathsf{y}}, \nabla f^*(z^{\mathsf{f}^*}), \nabla g^*(z^{\mathsf{g}^*}) \right) \\ \Phi^{\text{bilin}}(z) &:= (z^{\mathsf{f}^*}, z^{\mathsf{g}^*}, -z^{\mathsf{x}}, -z^{\mathsf{y}}), \\ \Phi^h(z) &:= (\nabla_x h(z^{\mathsf{x}}, z^{\mathsf{y}}), -\nabla_y h(z^{\mathsf{x}}, z^{\mathsf{y}}), 0, 0)\,. \end{aligned} \quad (2.28)$$

It is straightforward to check that Φ, the gradient operator of $F_{\text{mm-pd}}$, satisfies

$$\Phi(z) := \Phi^r(z) + \Phi^{\text{bilin}}(z) + \Phi^h(z). \quad (2.29)$$

Finally, we note that by construction Φ is 1-strongly monotone with respect to r.

Lemma 11 (Strong monotonicity)**.** *The operator Φ (as defined in* (I.20)*) is 1-strongly monotone with respect to the function $r : \mathcal{Z} \to \mathbb{R}$ as in* (7.66)*.*

Proof. Consider the decomposition of $\Phi = \Phi^r + \Phi^{\text{bilin}} + \Phi^h$ defined in (2.28) and (I.20). By definition and Items 1 to 3 from Fact 35, we know the operators Φ^h and Φ^{bilin} are monotone, and $\Phi^r = \nabla r$ is 1-strongly monotone with respect to r. Combining the three operators and using additivity of monotonicity in Item 4 of Fact 35 yields the claim. □

2.7.2 Algorithm

Our algorithm will be an instantiation of *strongly monotone mirror prox*, stated as Algorithm 3 below and analyzed in Appendix A.4, an alternative to the mirror prox algorithm originally proposed by [415].

Algorithm 3: SM-MIRROR-PROX(λ, T, z_0): Strongly monotone mirror prox

Input: Convex $r : \mathcal{Z} \to \mathbb{R}$, m-strongly monotone $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ (with respect to r), $z_0 \in \mathcal{Z}$
Parameter(s): $\lambda > 0$, $T \in \mathbb{N}$
for $0 \le t < T$ **do**
  $z_{t+1/2} \leftarrow \text{Prox}^r_{z_t}(\frac{1}{\lambda}\Phi(z_t))$
  $z_{t+1} \leftarrow \text{argmin}_{z \in \mathcal{Z}}\{\frac{1}{\lambda}\langle \Phi(z_{t+1/2}), z\rangle + \frac{m}{\lambda} V^r_{z_{t+1/2}}(z) + V^r_{z_t}(z)\}$

We prove the following result in Appendix A.4.

Proposition 3. *If Φ is λ-relatively Lipschitz with respect to r over $\mathcal{Z}_{\text{alg}}$ containing all iterates of Algorithm 3, and its VI is solved by $z_\star$, the iterates of Algorithm 3 satisfy*

$$V^r_{z_t}(z_\star) \le \left(1 + \frac{m}{\lambda}\right)^{-t} V^r_{z_0}(z_\star), \text{ for all } t \in [T].$$

Our algorithm in this section, Algorithm 4, will simply apply Algorithm 3 to the operator-regularizer pair (Φ, r) defined in (I.20) and (7.66). We give this implementation as pseudocode below, and show that it is a correct implementation of Algorithm 3 in the following lemma.

Lemma 12. *Algorithm 4 implements Algorithm 3 with $m = 1$ on (Φ, r) defined in* (I.20), (7.66).

Proof. Let $\{z_t, z_{t+1/2}\}_{0 \le t \le T}$ be the iterates of Algorithm 3. We will inductively show that Algorithm 4 preserves the invariants

$$z_t = \left(z_t^{\mathsf{x}}, z_t^{\mathsf{y}}, \nabla f\left(z_t^{\mathsf{f}}\right), \nabla g\left(z_t^{\mathsf{g}}\right)\right), \; z_{t+1/2} = \left(z_{t+1/2}^{\mathsf{x}}, z_{t+1/2}^{\mathsf{y}}, \nabla f\left(z_{t+1/2}^{\mathsf{f}}\right), \nabla g\left(z_{t+1/2}^{\mathsf{g}}\right)\right),$$

for the iterates of Algorithm 4. Once we prove this claim, it is clear from inspection that Algorithm 4 implements Algorithm 3, upon recalling the definitions (I.20), (7.66).

The base case of our induction follows from our initialization so that $(\nabla f(z_0^{\mathsf{f}}), \nabla g(z_0^{\mathsf{g}})) \leftarrow (\nabla f(x_0), \nabla f(y_0))$. Next, suppose for some $0 \le t < T$, we have $z_t^{\mathsf{f}^*} = \nabla f(z_t^{\mathsf{f}})$ and $z_t^{\mathsf{g}^*} = \nabla g(z_t^{\mathsf{g}})$. By

Algorithm 4: MINIMAX-SOLVE($F_{\text{mm-reg}}, x_0, y_0$): Separable minimax optimization

Input: (2.24) satisfying Assumption 1, $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$
Parameter(s): $\lambda > 0$, $T \in \mathbb{N}$
$(z_0^{\mathsf{x}}, z_0^{\mathsf{y}}) \leftarrow (x_0, y_0)$, $(z_0^{\mathsf{f}}, z_0^{\mathsf{g}}) \leftarrow (x_0, y_0)$
for $0 \le t < T$ **do**

$$\Phi^{\mathsf{x}} \leftarrow \mu^{\mathsf{x}} z_t^{\mathsf{x}} + \nabla f(z_t^{\mathsf{f}}) + \nabla_x h(z_t^{\mathsf{x}}, z_t^{\mathsf{y}})$$
$$\Phi^{\mathsf{y}} \leftarrow \mu^{\mathsf{y}} z_t^{\mathsf{y}} + \nabla g(z_t^{\mathsf{g}}) - \nabla_y h(z_t^{\mathsf{x}}, z_t^{\mathsf{y}}).$$

▷gradient step:

$z_{t+1/2}^{\mathsf{x}} \leftarrow z^{\mathsf{x}} - \frac{1}{\lambda\mu^{\mathsf{x}}}\Phi^{\mathsf{x}}$
$z_{t+1/2}^{\mathsf{y}} \leftarrow z^{\mathsf{y}} - \frac{1}{\lambda\mu^{\mathsf{y}}}\Phi^{\mathsf{y}}$
$z_{t+1/2}^{\mathsf{f}} \leftarrow (1 - \frac{1}{\lambda})z_t^{\mathsf{f}} + \frac{1}{\lambda}z_t^{\mathsf{x}}$ and $z_{t+1/2}^{\mathsf{g}} \leftarrow (1 - \frac{1}{\lambda})z_t^{\mathsf{g}} + \frac{1}{\lambda}z_t^{\mathsf{y}}$

$$\Phi^{\mathsf{x}} \leftarrow \mu^{\mathsf{x}} z_{t+1/2}^{\mathsf{x}} + \nabla f(z_{t+1/2}^{\mathsf{f}}) + \nabla_x h(z_{t+1/2}^{\mathsf{x}}, z_{t+1/2}^{\mathsf{y}})$$
$$\Phi^{\mathsf{y}} \leftarrow \mu^{\mathsf{y}} z_{t+1/2}^{\mathsf{y}} + \nabla g(z_{t+1/2}^{\mathsf{g}}) - \nabla_y h(z_{t+1/2}^{\mathsf{x}}, z_{t+1/2}^{\mathsf{y}})$$

▷extragradient step:

$z_{t+1}^{\mathsf{x}} \leftarrow \frac{1}{1+\lambda}z_{t+1/2}^{\mathsf{x}} + \frac{\lambda}{1+\lambda}z_t^{\mathsf{x}} - \frac{1}{(1+\lambda)\mu^{\mathsf{x}}}\Phi^{\mathsf{x}}$
$z_{t+1}^{\mathsf{y}} \leftarrow \frac{1}{1+\lambda}z_{t+1/2}^{\mathsf{y}} + \frac{\lambda}{1+\lambda}z_t^{\mathsf{y}} - \frac{1}{(1+\lambda)\mu^{\mathsf{y}}}\Phi^{\mathsf{y}}$
$z_{t+1}^{\mathsf{f}} \leftarrow \frac{\lambda}{1+\lambda}z_t^{\mathsf{f}} + \frac{1}{1+\lambda}z_{t+1/2}^{\mathsf{x}}$ and $z_{t+1}^{\mathsf{g}} \leftarrow \frac{\lambda}{1+\lambda}z_t^{\mathsf{g}} + \frac{1}{1+\lambda}z_{t+1/2}^{\mathsf{y}}$

the updates in Algorithm 3,

$$\begin{aligned} z_{t+1/2}^{\mathsf{f}^*} &\leftarrow \operatorname{argmin}_{z^{\mathsf{f}^*} \in \mathcal{X}^*} \left\{ \frac{1}{\lambda} \left\langle \nabla f^*(z_t^{\mathsf{f}^*}) - z_t^{\mathsf{x}}, z^{\mathsf{f}^*} \right\rangle + V_{z_t^{\mathsf{f}^*}}^{f^*}(z^{\mathsf{f}^*}) \right\} \\ &= \operatorname{argmin}_{z^{\mathsf{f}^*} \in \mathcal{X}^*} \left\{ \frac{1}{\lambda} \left\langle z_t^{\mathsf{f}} - z_t^{\mathsf{x}}, z^{\mathsf{f}^*} \right\rangle - \left\langle z_t^{\mathsf{f}}, z^{\mathsf{f}^*} \right\rangle + f^*(z^{\mathsf{f}^*}) \right\} \\ &= \operatorname{argmax}_{z^{\mathsf{f}^*} \in \mathcal{X}^*} \left\{ \left\langle \left(1 - \frac{1}{\lambda}\right) z_t^{\mathsf{f}} + \frac{1}{\lambda} z_t^{\mathsf{x}}, z^{\mathsf{f}^*} \right\rangle - f^*(z^{\mathsf{f}^*}) \right\} = \nabla f \left(\left(1 - \frac{1}{\lambda}\right) z_t^{\mathsf{f}} + \frac{1}{\lambda} z_t^{\mathsf{x}} \right). \end{aligned}$$

The second line used our inductive hypothesis and Item 3 in Fact 1, and the last used Item 1 in Fact 1. Hence, the update to $z_{t+1/2}^{\mathsf{f}}$ in Algorithm 4 preserves our invariant; a symmetric argument yields $z_{t+1/2}^{\mathsf{g}^*} = \nabla g(z_{t+1/2}^{\mathsf{g}})$ where $z_{t+1/2}^{\mathsf{g}} := (1 - \frac{1}{\lambda})z_t^{\mathsf{g}} + \frac{1}{\lambda}z_t^{\mathsf{y}}$.

Similarly, we show we may preserve this invariant for z_{t+1}:

$$\begin{aligned} z_{t+1}^{\mathsf{f}^*} &\leftarrow \operatorname{argmin}_{z^{\mathsf{f}^*} \in \mathcal{X}^*} \left\{ \frac{1}{\lambda} \left\langle z_{t+1/2}^{\mathsf{f}} - z_{t+1/2}^{\mathsf{x}}, z^{\mathsf{f}^*} \right\rangle - \frac{1}{\lambda} \left\langle z_{t+1/2}^{\mathsf{f}}, z^{\mathsf{f}^*} \right\rangle - \left\langle z_t^{\mathsf{f}}, z^{\mathsf{f}^*} \right\rangle + \left(1 + \frac{1}{\lambda}\right) f^*(z^{\mathsf{f}^*}) \right\} \\ &= \operatorname{argmax}_{a \in \mathcal{X}^*} \left\{ \left\langle z_t^{\mathsf{f}} + \frac{1}{\lambda} z_{t+1/2}^{\mathsf{x}}, z^{\mathsf{f}^*} \right\rangle - \left(1 + \frac{1}{\lambda}\right) f^*(z^{\mathsf{f}^*}) \right\} = \nabla f \left(\frac{\lambda}{1+\lambda} z_t^{\mathsf{f}} + \frac{1}{1+\lambda} z_{t+1/2}^{\mathsf{x}} \right). \end{aligned}$$

Hence, we may set $z_{t+1}^{\mathsf{f}} := \frac{\lambda}{1+\lambda} z_t^{\mathsf{f}} + \frac{1}{1+\lambda} z_{t+1/2}^{\mathsf{x}}$ and similarly, $z_{t+1}^{\mathsf{g}} := \frac{\lambda}{1+\lambda} z_t^{\mathsf{g}} + \frac{\lambda}{1+\lambda} z_{t+1/2}^{\mathsf{y}}$. □

As an immediate corollary of Lemma 12, we have the following characterization of our iterates, recalling the definitions of $\mathcal{X}_f^*$ and $\mathcal{Y}_g^*$ from Section 2.2.

Corollary 2. *Define the product space $\mathcal{Z}_{\text{alg}} := \mathcal{X} \times \mathcal{Y} \times \mathcal{X}_f^* \times \mathcal{Y}_g^*$, where $\mathcal{X}_f^* := \{\nabla f(x) \mid x \in \mathcal{X}\}$ and $\mathcal{Y}_g^* := \{\nabla g(y) \mid y \in \mathcal{Y}\}$. Then all iterates of Algorithm 4 lie in $\mathcal{Z}_{\text{alg}}$.*

For a point $z \in \mathcal{Z}_{\text{alg}}$, we define the points $z^{\mathsf{f}} := \nabla f^*(z^{\mathsf{f}^*})$ and $z^{\mathsf{g}} := \nabla g^*(z^{\mathsf{g}^*})$. By Item 3 of Fact 1, this implies $z^{\mathsf{f}}, z^{\mathsf{g}} \in \mathcal{X}$ since z^{f^*} and z^{g^*} are appropriate gradients.

2.7.3 Convergence analysis

In order to use Proposition 3 to analyze Algorithm 4, we require a strong monotonicity bound and a relative Lipschitzness bound on the pair (Φ, r); the former is already given by Lemma 11. We build up to the latter bound by first giving the following consequences of Assumption 1, inspired by a similar proof strategy as used in Lemma 4.

Lemma 13 (Minimax smoothness implications)**.** *Let convex $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{Y} \to \mathbb{R}$, and convex-concave $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ satisfy Assumption 1. Then, the following hold.*

1. *$|\langle \nabla f(v) - \nabla f(w), x - y \rangle| \le \alpha L^{\mathsf{x}} V_v^f(w) + \alpha^{-1} V_x(y)$ for all $v, w, x, y \in \mathcal{X}$ and $\alpha > 0$.*
2. *$|\langle \nabla g(v) - \nabla g(w), x - y \rangle| \le \alpha L^{\mathsf{y}} V_v^g(w) + \alpha^{-1} V_x(y)$ for all $v, w, x, y \in \mathcal{Y}$ and $\alpha > 0$.*
3. *Φ^h is 1-relatively Lipschitz with respect to $r_\alpha^h : \mathcal{Z} \to \mathbb{R}$ defined for all $z \in \mathcal{Z}$ and $\alpha > 0$ by $r_\alpha^h(z) := \frac{1}{2}\left(\Lambda^{\mathsf{xx}} + \alpha\Lambda^{\mathsf{xy}}\right)\|z^{\mathsf{x}}\|^2 + \frac{1}{2}\left(\Lambda^{\mathsf{yy}} + \alpha^{-1}\Lambda^{\mathsf{xy}}\right)\|z^{\mathsf{y}}\|^2$.*

Proof. We will prove Items 1 and 3, as Item 2 follows symmetrically to Item 1.

Proof of Item 1. We compute:

$$\begin{aligned}
|\langle \nabla f(v) - \nabla f(w), x - y \rangle| &\le \|\nabla f(v) - \nabla f(w)\| \, \|x - y\| \\
&\le \frac{\alpha}{2}\|\nabla f(v) - \nabla f(w)\|^2 + \frac{1}{2\alpha}\|x - y\|^2 \\
&\le \alpha L^{\mathsf{x}} V_{\nabla f(w)}^{f^*}(\nabla f(v)) + \alpha^{-1} V_x(y) = \alpha L^{\mathsf{x}} V_v^f(w) + \alpha^{-1} V_x(y).
\end{aligned}$$

The first inequality was Cauchy-Schwarz, the second was Young's inequality, and the third used Items 3 and 4 in Fact 1. The last equality follows from Fact 1.

Proof of Item 3. Let $w, v, z \in \mathcal{Z}$ be arbitrary. We have,

$$\begin{aligned}&\left\langle \Phi^h(w) - \Phi^h(z), w - v \right\rangle \\ &\quad = \left\langle \nabla_x h(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_x h(z^{\mathsf{x}}, z^{\mathsf{y}}), w^{\mathsf{x}} - v^{\mathsf{x}} \right\rangle - \left\langle \nabla_y h(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_y h(z^{\mathsf{x}}, z^{\mathsf{y}}), w^{\mathsf{y}} - v^{\mathsf{y}} \right\rangle.\end{aligned}$$

Applying Cauchy-Schwarz, Young's inequality, and Assumption 1 yields

$$\begin{aligned}\left\langle \nabla_x h(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_x h(z^{\mathsf{x}}, z^{\mathsf{y}}), w^{\mathsf{x}} - v^{\mathsf{x}} \right\rangle &\leq \left\| \nabla_x h(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_x h(z^{\mathsf{x}}, z^{\mathsf{y}}) \right\| \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\| \\ &\leq \left(\Lambda^{\mathsf{xx}} \left\| w^{\mathsf{x}} - z^{\mathsf{x}} \right\| + \Lambda^{\mathsf{xy}} \left\| w^{\mathsf{y}} - z^{\mathsf{y}} \right\| \right) \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\| \\ &\leq \frac{\Lambda^{\mathsf{xx}}}{2} \left\| w^{\mathsf{x}} - z^{\mathsf{x}} \right\|^2 + \frac{\Lambda^{\mathsf{xx}}}{2} \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\|^2 + \Lambda^{\mathsf{xy}} \left\| w^{\mathsf{y}} - z^{\mathsf{y}} \right\| \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\|.\end{aligned}$$

Symmetrically,

$$\left\langle \nabla_y h(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_y h(z^{\mathsf{x}}, z^{\mathsf{y}}), w^{\mathsf{y}} - v^{\mathsf{y}} \right\rangle \leq \frac{\Lambda^{\mathsf{yy}}}{2} \left\| w^{\mathsf{y}} - z^{\mathsf{y}} \right\|^2 + \frac{\Lambda^{\mathsf{yy}}}{2} \left\| w^{\mathsf{y}} - v^{\mathsf{y}} \right\|^2 + \Lambda^{\mathsf{xy}} \left\| w^{\mathsf{x}} - z^{\mathsf{x}} \right\| \left\| w^{\mathsf{y}} - v^{\mathsf{y}} \right\|.$$

Applying Young's inequality again yields

$$\begin{aligned}\Lambda^{\mathsf{xy}} \left\| w^{\mathsf{y}} - z^{\mathsf{y}} \right\| \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\| &\leq \frac{\alpha \Lambda^{\mathsf{xy}}}{2} \left\| w^{\mathsf{x}} - v^{\mathsf{x}} \right\|^2 + \frac{\Lambda^{\mathsf{xy}}}{2\alpha} \left\| w^{\mathsf{y}} - z^{\mathsf{y}} \right\|^2, \\ \text{and } \Lambda^{\mathsf{xy}} \left\| w^{\mathsf{x}} - z^{\mathsf{x}} \right\| \left\| w^{\mathsf{y}} - v^{\mathsf{y}} \right\| &\leq \frac{\alpha \Lambda^{\mathsf{xy}}}{2} \left\| w^{\mathsf{x}} - z^{\mathsf{x}} \right\|^2 + \frac{\Lambda^{\mathsf{xy}}}{2\alpha} \left\| w^{\mathsf{y}} - v^{\mathsf{y}} \right\|^2.\end{aligned}$$

Combining these inequalities yields the desired bound of

$$\begin{aligned}\left\langle \Phi^h(w) - \Phi^h(z), w - v \right\rangle &\leq \left(\Lambda^{\mathsf{xx}} + \alpha \Lambda^{\mathsf{xy}} \right) \left(V_{z^{\mathsf{x}}}(w^{\mathsf{x}}) + V_{w^{\mathsf{x}}}(v^{\mathsf{x}}) \right) + \left(\Lambda^{\mathsf{yy}} + \alpha \Lambda^{\mathsf{xy}} \right) \left(V_{z^{\mathsf{y}}}(w^{\mathsf{y}}) + V_{w^{\mathsf{y}}}(v^{\mathsf{y}}) \right) \\ &= V_z^{r_\alpha^h}(w) + V_w^{r_\alpha^h}(v).\end{aligned}$$

□

Leveraging Lemma 13 and Lemma 1, we prove relative Lipschitzness of Φ with respect to r in Lemma 14. Interestingly, the implications in Lemma 13 are sufficient for this proof, and this serves as a (potentially) weaker replacement for Assumption 1 in yielding a convergence rate for our method.

This is particularly interesting when the condition in Item 1 is replaced with a non-Euclidean divergence, namely $|\langle \nabla f(v) - \nabla f(w), x - y \rangle| \leq \alpha L^{\mathsf{x}} V_v^f(w) + \alpha^{-1} V_x^\omega(y)$ for some convex $\omega : \mathcal{X} \to \mathbb{R}$. Setting, setting $v = y, w = x, \alpha = \frac{1}{L^{\mathsf{x}}}$ in this condition yields $V_x^f(y) \leq L V_x^\omega(y)$. Hence, this extension to Item 1 generalizes *relative smoothness* between f and ω, a condition introduced by [64, 375]. It has been previously observed [265, 197] that relative smoothness alone does not suffice for accelerated rates. Item 1 provides a new strengthening of relative smoothness which, as shown by its (implicit) use in Section 2.4, suffices for acceleration. We believe a more thorough investigation comparing these conditions is an interesting avenue for future work.

Lemma 14 (Relative Lipschitzness)**.** *Define* $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ *as in* (I.20)*, and define* $r : \mathcal{Z} \to \mathbb{R}$ *as in* (7.66)*. Then* Φ *is* λ*-relatively Lipschitz with respect to* r *over* $\mathcal{Z}_{\text{alg}}$ *defined in Corollary 2 for*

$$\lambda = 1 + \sqrt{\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}}. \tag{2.30}$$

Proof. Let $w, v, z \in \mathcal{Z}_{\text{alg}}$. We wish to show (cf. Definition 1)

$$\langle \Phi(w) - \Phi(z), w - v \rangle \leq \lambda \left(V_z^r(w) + V_w^r(v) \right).$$

Since $\Phi = \Phi^r + \Phi^{\text{bilin}} + \Phi^h$ (cf. (I.20)), we bound the contribution of each term individually. The conclusion follows from combining (2.31), (2.32), and (2.33).

Bound on Φ^r: By applying Lemma 1 to r,

$$\langle \Phi^r(w) - \Phi^r(z), w - v \rangle = \langle \nabla r(w) - \nabla r(z), w - v \rangle \leq V_z^r(w) + V_w^r(v). \tag{2.31}$$

Bound on Φ^{bilin}: For all $a \in \mathcal{Z}_{\text{alg}}$, we may write for some $a^{\mathsf{f}} \in \mathcal{X}$ and $a^{\mathsf{g}} \in \mathcal{Y}$,

$$\begin{aligned} \Phi^{\text{bilin}}(a) &= (a^{\mathsf{f}^*}, a^{\mathsf{g}^*}, -a^{\mathsf{x}}, -a^{\mathsf{y}}) = (\nabla f(a^{\mathsf{f}}), \nabla g(a^{\mathsf{g}}), -a^{\mathsf{x}}, -a^{\mathsf{y}}) \\ \text{and } a &= (a^{\mathsf{x}}, a^{\mathsf{y}}, a^{\mathsf{f}^*}, a^{\mathsf{g}^*}) = (a^{\mathsf{x}}, a^{\mathsf{y}}, \nabla f(a^{\mathsf{f}}), \nabla g(a^{\mathsf{g}})). \end{aligned}$$

Consequently,

$$\begin{aligned} \left\langle \Phi^{\text{bilin}}(w) - \Phi^{\text{bilin}}(z), w - v \right\rangle &= \left\langle \nabla f(w^{\mathsf{f}}) - \nabla f(z^{\mathsf{f}}), w^{\mathsf{x}} - v^{\mathsf{x}} \right\rangle + \left\langle \nabla g(w^{\mathsf{f}}) - \nabla g(z^{\mathsf{f}}), w^{\mathsf{y}} - v^{\mathsf{y}} \right\rangle \\ &\quad - \left\langle w^{\mathsf{x}} - z^{\mathsf{x}}, \nabla f(w^{\mathsf{f}}) - \nabla f(v^{\mathsf{f}}) \right\rangle - \left\langle w^{\mathsf{y}} - z^{\mathsf{y}}, \nabla g(w^{\mathsf{g}}) - \nabla g(v^{\mathsf{g}}) \right\rangle. \end{aligned}$$

Applying Lemma 13 (Item 1 and Item 2) to each term, with $\alpha = (\mu^{\mathsf{x}} L^{\mathsf{x}})^{-\frac{1}{2}}$ for terms involving f and $\alpha = (\mu^{\mathsf{y}} L^{\mathsf{y}})^{-\frac{1}{2}}$ for terms involving g yields

$$\begin{aligned} \left\langle \Phi^{\text{bilin}}(w) - \Phi^{\text{bilin}}(z), w - v \right\rangle &\leq \sqrt{\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}}} \left(V_{w^{\mathsf{f}}}^f(z^{\mathsf{f}}) + V_{v^{\mathsf{f}}}^f(w^{\mathsf{f}}) \right) + \sqrt{\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}}} \left(\mu^{\mathsf{x}} V_{w^{\mathsf{x}}}(v^{\mathsf{x}}) + \mu^{\mathsf{x}} V_{z^{\mathsf{x}}}(w^{\mathsf{x}}) \right) \\ &\quad + \sqrt{\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}} \left(V_{w^{\mathsf{g}}}^g(z^{\mathsf{g}}) + V_{v^{\mathsf{g}}}^g(w^{\mathsf{g}}) \right) + \sqrt{\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}} \left(\mu^{\mathsf{y}} V_{w^{\mathsf{y}}}(v^{\mathsf{y}}) + \mu^{\mathsf{y}} V_{z^{\mathsf{y}}}(w^{\mathsf{y}}) \right). \end{aligned}$$

Applying Item 3 in Fact 1 and recalling the definition of r (7.66) yields

$$\left\langle \Phi^{\text{bilin}}(w) - \Phi^{\text{bilin}}(z), w - v \right\rangle \leq \left(\sqrt{\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}} \right) \left(V_z^r(w) + V_w^r(v) \right). \tag{2.32}$$

Bound on Φ^h: Applying Lemma 13 (Item 3 with $\alpha = \sqrt{\mu^{\mathsf{x}}/\mu^{\mathsf{y}}}$), we have that Φ^h is 1-relatively Lipschitz with respect to $r^h_\alpha : \mathcal{Z} \to \mathbb{R}$ defined for all $z \in \mathcal{X}$ and $\alpha > 0$ by

$$\begin{aligned} r^h_\alpha(z) &:= \frac{1}{2}\left(\Lambda^{\mathsf{xx}} + \alpha\Lambda^{\mathsf{xy}}\right)\|z^{\mathsf{x}}\|^2 + \frac{1}{2}\left(\Lambda^{\mathsf{yy}} + \alpha^{-1}\Lambda^{\mathsf{xy}}\right)\|z^{\mathsf{y}}\|^2 \\ &= \left(\frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right)\cdot\frac{\mu^{\mathsf{x}}}{2}\|z^{\mathsf{x}}\|^2 + \left(\frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right)\cdot\frac{\mu^{\mathsf{y}}}{2}\|z^{\mathsf{y}}\|^2. \end{aligned}$$

Leveraging the nonnegativity of Bregman divergences, we conclude

$$\begin{aligned} \left\langle \Phi^h(w) - \Phi^h(z), w - v\right\rangle &\le V_z^{r^h_\alpha}(w) + V_w^{r^h_\alpha}(v) \\ &\le \left(\frac{\Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}}\right)\left(V_z^r(w) + V_w^r(v)\right). \end{aligned} \tag{2.33}$$

□

Finally, we provide simple bounds regarding initialization and termination of Algorithm 4.

Lemma 15. *Let $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, and define*

$$z_0 := (x_0, y_0, \nabla f(x_0), \nabla g(y_0)). \tag{2.34}$$

Suppose $\mathrm{Gap}_{F_{\text{mm-reg}}}(x_0, y_0) \le \epsilon_0$. *Then, letting $z_\star$ be the solution to* (2.26),

$$V^r_{z_0}(z_\star) \le \left(1 + \frac{L^{\mathsf{x}}}{\mu_x} + \frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}}\right)\epsilon_0.$$

Proof. By the characterization in Lemma 10, we have by Item 1 in Fact 1:

$$z_\star = (x_\star, y_\star, \nabla f(x_\star), \nabla g(y_\star)).$$

Hence, we bound

$$\begin{aligned} V^r_{z_0}(z_\star) &= \mu^{\mathsf{x}} V_{x_0}(x_\star) + V^f_{x_\star}(x_0) + \mu^{\mathsf{y}} V_{y_0}(y_\star) + V^g_{y_\star}(y_0) \\ &\le \mu^{\mathsf{x}} V_{x_0}(x_\star) + \frac{L^{\mathsf{x}}}{2}\|x_0 - x_\star\|^2_{\mathcal{X}} + \mu^{\mathsf{y}} V_{y_0}(y_\star) + \frac{L^{\mathsf{y}}}{2}\|y_0 - y_\star\|^2_{\mathcal{Y}} \\ &= \left(\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}} + 1\right)\mu^{\mathsf{x}} V_{x_0}(x_\star) + \left(\frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}} + 1\right)\mu^{\mathsf{y}} V_{y_0}(y_\star) \\ &\le \left(\frac{L^{\mathsf{x}}}{\mu^{\mathsf{x}}} + \frac{L^{\mathsf{y}}}{\mu^{\mathsf{y}}} + 1\right)\epsilon_0. \end{aligned}$$

The first line used Item 3 in Fact 1, and the second used smoothness of f and g (Assumption 1).

To obtain the last line, define the functions

$$F^{\mathsf{x}}_{\text{mm-reg}}(x) := \max_{y\in\mathcal{Y}} F_{\text{mm-reg}}(x,y) \text{ and } F^{\mathsf{y}}_{\text{mm-reg}}(y) := \min_{x\in\mathcal{X}} F_{\text{mm-reg}}(x,y).$$

Fact 36 shows $F^{\mathsf{x}}_{\text{mm-reg}}$ is μ^{x}-strongly convex and $F^{\mathsf{y}}_{\text{mm-reg}}$ is μ^{y}-strongly concave, so

$$\begin{aligned}\text{Gap}_{F_{\text{mm-reg}}}(x_0,y_0) &= \left(F^{\mathsf{x}}_{\text{mm-reg}}(x_0) - F^{\mathsf{x}}_{\text{mm-reg}}(x_\star)\right) + \left(F^{\mathsf{y}}_{\text{mm-reg}}(y_\star) - F^{\mathsf{y}}_{\text{mm-reg}}(y_0)\right)\\ &\geq \mu^{\mathsf{x}} V_{x_0}(x^\star) + \mu^{\mathsf{y}} V_{y_0}(y^\star).\end{aligned}$$

□

Lemma 16. *Let $z \in \mathcal{Z}$ have*

$$V_z^r(z_\star) \leq \left(\frac{\mu^{\mathsf{x}} + L^{\mathsf{x}} + \Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\mu^{\mathsf{y}} + L^{\mathsf{y}} + \Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}\right)\cdot\frac{\epsilon}{2},$$

for $z_\star$ the solution to (2.26)*. Then,*

$$\text{Gap}_{F_{\text{mm-reg}}}(z^{\mathsf{x}}, z^{\mathsf{y}}) \leq \epsilon.$$

Proof. We follow the notation of Lemma 15. From Fact 36 we know $F^{\mathsf{x}}_{\text{mm-reg}}$ is $\mathcal{L}^{\mathsf{x}}$-smooth and $F^{\mathsf{y}}_{\text{mm-reg}}$ is $\mathcal{L}^{\mathsf{y}}$-smooth, where

$$\mathcal{L}^{\mathsf{x}} := \mu^{\mathsf{x}} + L^{\mathsf{x}} + \Lambda^{\mathsf{xx}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{y}}} \text{ and } \mathcal{L}^{\mathsf{y}} := \mu^{\mathsf{y}} + L^{\mathsf{y}} + \Lambda^{\mathsf{yy}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{x}}},$$

under Assumption 1. Moreover, by Lemma 10 and the definition of saddle points, $x_\star := z^{\mathsf{x}}_\star$ is the minimizer to $F^{\mathsf{x}}_{\text{mm-reg}}$, and $y_\star := z^{\mathsf{y}}_\star$ is the maximizer to $F^{\mathsf{y}}_{\text{mm-reg}}$. We conclude via

$$\begin{aligned}\text{Gap}_{F_{\text{mm-reg}}}(z^{\mathsf{x}}, z^{\mathsf{y}}) &= \left(F^{\mathsf{x}}_{\text{mm-reg}}(x) - F^{\mathsf{x}}_{\text{mm-reg}}(x_\star)\right) + \left(F^{\mathsf{y}}_{\text{mm-reg}}(y_\star) - F^{\mathsf{y}}_{\text{mm-reg}}(z^{\mathsf{y}})\right)\\ &\leq \left(\mu^{\mathsf{x}} + L^{\mathsf{x}} + \Lambda^{\mathsf{xx}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{y}}}\right)\|x - x_\star\|^2\\ &+ \left(\mu^{\mathsf{y}} + L^{\mathsf{y}} + \Lambda^{\mathsf{yy}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{x}}}\right)\|y - y_\star\|^2\\ &\leq 2\left(\frac{\mu^{\mathsf{x}} + L^{\mathsf{x}} + \Lambda^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\mu^{\mathsf{y}} + L^{\mathsf{y}} + \Lambda^{\mathsf{yy}}}{\mu^{\mathsf{y}}} + \frac{(\Lambda^{\mathsf{xy}})^2}{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}\right)V_z^r(z_\star) \leq \epsilon.\end{aligned}$$

The first inequality was smoothness of $F^{\mathsf{x}}_{\text{mm-reg}}$ and $F^{\mathsf{y}}_{\text{mm-reg}}$ (where we used that the gradients at $x_\star$ and $y_\star$ vanish because the optimization problems they solve are over unconstrained domains), and the last inequality was nonnegativity of Bregman divergences. □

2.7.4 Main result

We now state and prove our main claim.

Theorem 7. *Suppose $F_{\text{mm-reg}}$ in (2.24) satisfies Assumption 1, and suppose we have $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ such that $\text{Gap}_{F_{\text{mm-reg}}}(x_0, y_0) \leq \epsilon_0$. Algorithm 4 with λ as in (2.30) returns $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $\text{Gap}_{F_{\text{mm-reg}}}(x, y) \leq \epsilon$ in T iterations, using a total of $O(T)$ gradient calls to each of f, g, h, where*

$$T = O\left(\kappa_{\text{mm}} \log\left(\frac{\kappa_{\text{mm}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\text{mm}} := \sqrt{\frac{L^{\text{x}}}{\mu^{\text{x}}}} + \sqrt{\frac{L^{\text{y}}}{\mu^{\text{y}}}} + \frac{\Lambda^{\text{xx}}}{\mu^{\text{x}}} + \frac{\Lambda_{xy}}{\sqrt{\mu^{\text{x}}\mu^{\text{y}}}} + \frac{\Lambda^{\text{yy}}}{\mu^{\text{y}}}. \tag{2.35}$$

Proof. By Lemma 10, the points $x_\star$ and $y_\star$ are consistent between (2.24) and (2.26). The gradient complexity of each iteration follows from observation of Algorithm 4.

Next, by Lemma 12, Algorithm 4 implements Algorithm 3 on the pair (I.20), (7.66). By substituting the bounds on λ and m in Lemmas 14 and 11 into Proposition 3 (where we define $\mathcal{Z}_{\text{alg}}$ as in Corollary 2), it is clear that after T iterations (for a sufficiently large constant in the definition of T), we will have $V^r_{z_T}(z_\star)$ is bounded by the quantity in Lemma 16, where we use the initial bound on $V^r_{z_0}(z^\star)$ from Lemma 15. The conclusion follows from setting $(x, y) \leftarrow (z^{\text{x}}_T, z^{\text{y}}_T)$. □

As an immediate corollary, we have the following result on solving (2.23).

Corollary 3. *Suppose for F_{mm} in (2.23) solved by $(x_\star, y_\star)$, $(f - \frac{\mu^{\text{x}}}{2}\|\cdot\|^2, g - \frac{\mu^{\text{y}}}{2}\|\cdot\|^2, h)$ satisfies Assumption 1. There is an algorithm taking $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ satisfying $\text{Gap}_{F_{\text{mm}}}(x_0, y_0) \leq \epsilon_0$, which performs T iterations for T in (2.35), returns $(x, y) \in \mathcal{X} \times \mathcal{Y}$ satisfying $\text{Gap}_{F_{\text{mm}}}(x, y) \leq \epsilon$, and uses a total of $O(T)$ gradient calls to each using $O(1)$ gradient calls to each of f, g, h.*

2.8 Finite sum optimization

In this section, we give an algorithm for efficiently finding an approximate minimizer of the following finite sum optimization problem:

$$F_{\text{fs}}(x) := \frac{1}{n}\sum_{i\in[n]} f_i(x). \tag{2.36}$$

Here and throughout this section $f_i : \mathcal{X} \to \mathbb{R}$ is a differentiable, convex function for all $i \in [n]$. For the remainder, we focus on algorithms for solving the following regularized formulation of (2.36):

$$\min_{x\in\mathcal{X}} F_{\text{fs-reg}}(x) \text{ for } F_{\text{fs-reg}}(x) := \frac{1}{n}\sum_{i\in[n]} f_i(x) + \frac{\mu}{2}\|x\|^2. \tag{2.37}$$

As in Section 2.7, to solve an instance of (2.36) where each f_i is μ-strongly convex, we may instead equivalently solve (2.37) by reparameterizing $f_i \leftarrow f_i - \frac{\mu}{2}\|\cdot\|^2$ for all $i \in [n]$. We further remark

that our algorithms extend to solve instances of (2.36) where F_{fs} is μ-strongly convex in $\|\cdot\|$, but individual summands are not. We provide this result at the end of the section in Corollary 4.

In designing methods for solving (2.37) we make the following additional regularity assumptions.

Assumption 2. *For all $i \in [n]$, f_i is L_i-smooth.*

The remainder of this section is organized as follows.

1. In Section 2.8.1, we state a primal-dual formulation of (2.37) which we will apply our methods to, and prove that its solution also yields a solution to (2.37).
2. In Section 2.8.2, we give our algorithm and prove it is efficiently implementable.
3. In Section 2.8.3, we prove the convergence rate of our algorithm.
4. In Section 2.8.4, we state and prove our main result, Theorem 8.

2.8.1 Setup

To solve (2.37), we instead find a saddle point to the primal-dual function

$$F_{\text{fs-pd}}(z) := \frac{1}{n}\sum_{i\in[n]}\left(\left\langle z^{\mathsf{f}_i^*}, z^{\mathsf{x}}\right\rangle - f_i^*(z^{\mathsf{f}_i^*})\right) + \frac{\mu}{2}\|z^{\mathsf{x}}\|^2. \tag{2.38}$$

We denote the domain of $F_{\text{fs-pd}}$ by $\mathcal{Z} := \mathcal{X} \times (\mathcal{X}^*)^n$. For $z \in \mathcal{Z}$, we refer to its blocks by $(z^{\mathsf{x}}, \{z^{\mathsf{f}_i^*}\}_{i\in[n]})$. The primal-dual function $F_{\text{fs-pd}}$ is related to $F_{\text{fs-reg}}$ in the following way.

Lemma 17. *Let $z_\star$ be the saddle point to* (2.38)*. Then, $z_\star^{\mathsf{x}}$ is a minimizer of* (2.37)*.*

Proof. By performing the maximization over each $z^{\mathsf{f}_i^*}$, we see that the problem of computing a minimizer to the objective in (2.38) is equivalent to

$$\min_{z^{\mathsf{x}}\in\mathcal{X}} \frac{\mu}{2}\|z^{\mathsf{x}}\|^2 + \frac{1}{n}\sum_{i\in[n]}\left(\max_{z^{\mathsf{f}_i^*}\in\mathcal{X}^*}\left\langle z^{\mathsf{f}_i^*}, z^{\mathsf{x}}\right\rangle - f_i^*(z^{\mathsf{f}_i^*})\right).$$

By Item 2 in Fact 1, this is the same as (2.37). □

As in Section 2.7.1, it will be convenient to define the convex function $r : \mathcal{Z} \to \mathbb{R}$, which combines the (unsigned) separable components of $F_{\text{fs-pd}}$:

$$r(z) := \frac{\mu}{2}\|z^{\mathsf{x}}\|^2 + \frac{1}{n}\sum_{i\in[n]} f_i^*(z^{\mathsf{f}_i^*}). \tag{2.39}$$

Again, r serves as a regularizer in our algorithm. We next define Φ, the gradient operator of $F_{\text{fs-pd}}$:

$$\Phi(z) := \left(\frac{1}{n} \sum_{i \in [n]} z^{\mathsf{f}_i^*} + \mu z^{\mathsf{x}}, \left\{ \frac{1}{n} \left(\nabla f_i^*(z^{\mathsf{f}_i^*}) - z^{\mathsf{x}} \right) \right\}_{i \in [n]} \right). \tag{2.40}$$

By construction, Φ is 1-strongly monotone with respect to r.

Lemma 18 (Strong monotonicity)**.** *Define $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ as in* (2.40)*, and define $r : \mathcal{Z} \to \mathbb{R}$ as in* (2.39)*. Then Φ is 1-strongly-monotone with respect to r.*

Proof. The proof is identical to Lemma 11 without the Φ^h term: the bilinear component cancels in the definition of strong monotonicity, and the remaining part is exactly the gradient of r. □

2.8.2 Algorithm

Our algorithm is an instantiation of *randomized mirror prox* stated as Algorithm 5 below, an extension to mirror prox allowing for randomized gradient estimators. This algorithm was also used in our developments in Section 2.6, but we restate it explicitly here for convenience. We note that the operators Φ_i need only be defined on iterates of the algorithm.

Algorithm 5: RAND-MIRROR-PROX($\{\Phi_i\}_{i \in [n]}, w_0$): Randomized mirror prox [147]

Input: Convex $r : \mathcal{Z} \to \mathbb{R}$, probability distribution $p : [n] \to \mathbb{R}_{\geq 0}$ with $\sum_{i \in [n]} p_i = 1$, operators $\{\Phi_i\}_{i \in [n]} : \mathcal{Z} \to \mathcal{Z}^*$, $z_0 \in \mathcal{Z}$;
Parameter(s): $\lambda > 0$, $S \in \mathbb{N}$;
for $0 \leq s < S$ **do**
  Sample $i \sim p$;
  $w_{s+1/2} \leftarrow \text{Prox}^r_{w_t}(\frac{1}{\lambda} \Phi_i(w_s))$;
  $w_{s+1} \leftarrow \text{Prox}^r_{w_t}(\frac{1}{\lambda} \Phi_i(w_{s+1/2}))$;

We restate the following result (originally given as Proposition 2 earlier) in slightly more generality, giving a guarantee on Algorithm 5.

Proposition 4. *Suppose $\{\Phi_i\}_{i \in [n]}$ are defined so that in each iteration s, for all $u \in \mathcal{Z}$, there exists a point $\bar{w}_s \in \mathcal{Z}$ and a monotone operator $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ such that (where all expectations fix w_s, and condition only on the randomness in iteration s)*

$$\begin{aligned} \mathbb{E}_{i \sim p} \left[\langle \Phi_i(w_{s+1/2}), w_{s+1/2} - u \rangle \right] &= \langle \Phi(\bar{w}_s), \bar{w}_s - u \rangle \ \text{ for all } u \in \mathcal{Z}, \\ \mathbb{E}_{i \sim p} \left[\langle \Phi_i(w_{s+1/2}) - \Phi_i(w_s), w_{s+1/2} - w_{s+1} \rangle \right] &\leq \lambda \mathbb{E}_{i \sim p} \left[V^r_{w_s}(w_{s+1/2}) + V^r_{w_{s+1/2}}(w_{s+1}) \right]. \end{aligned} \tag{2.41}$$

Then (where the expectation below is taken over the randomness of the entire algorithm):

$$\mathbb{E}\left[\frac{1}{S}\sum_{0\le s<S}\langle\Phi(\bar{w}_s),\bar{w}_s-u\rangle\right]\le\frac{\lambda V^r_{w_0}(u)}{S},\ \textit{for all } u\in\mathcal{Z}.$$

The first condition in (2.41) is an "unbiasedness" requirement on the operators $\{\Phi_i\}_{i\in[n]}$ with respect to the operator Φ, for which we wish to conclude a regret guarantee. The second posits that relative Lipschitzness (Definition 1) holds in an expected sense. We recall that Algorithm 5 requires us to specify a set of sampling probabilities $\{p_i\}_{i\in[n]}$. We define

$$p_i:=\frac{\sqrt{L_i}}{2\sum_{j\in[n]}\sqrt{L_j}}+\frac{1}{2n}\text{ for all } i\in[n]. \tag{2.42}$$

This choice crucially ensures that all $p_i\ge\frac{1}{2n}$, and that all $\frac{\sqrt{L_i}}{p_i}\le 2\sum_{j\in[n]}\sqrt{L_j}$.

Our algorithm, Algorithm 6, recursively applies Algorithm 5 to the operator-pair (Φ,r) defined in (2.40) and (2.39), for an appropriate specification of $\{\Phi_i\}_{i\in[n]}$. We give this implementation as pseudocode in Algorithms 6 and 7 below, and show that Algorithm 7 is a correct implementation of Algorithm 5 with respect to our specified $\{\Phi_i\}_{i\in[n]}$ in the remainder of the section.

Algorithm 6: FINITE-SUM-SOLVE$(F_{\text{fs-reg}},x_0)$: Finite sum optimization

Input: (2.37) satisfying Assumption 2, $x_0\in\mathcal{X}$;
Parameter(s): $T\in\mathbb{N}$;
$z_0^{\mathsf{x}}\leftarrow x_0$, $z_0^{\mathsf{f_i}}\leftarrow x_0$, $z_0^{\mathsf{f_i^*}}\leftarrow\nabla f_i(x_0)$ for all $i\in[n]$;
for $0\le t<T$ **do**
  $z_{t+1}\leftarrow$ FINITE-SUM-ONE-PHASE$(F_{\text{fs-reg}},z_t)$;

We next describe the operators $\{\Phi_i\}_{i\in[n]}$ used in our implementation of Algorithm 5. Fix some $0\le s<S$, and consider some iterates $\{w,w_{\mathsf{aux}}(j)\}:=\{w_s,w_{s+1/2}\}$ of Algorithm 5 (where we use the notation (j) to mean the iterate that would be taken if $j\in[n]$ was sampled in iteration s, and we drop the subscript s for simplicity since we only focus on one iteration). We denote the $\mathcal{X}$ block of $w_{\mathsf{aux}}(j)$ by $w^{\mathsf{x}}_{\mathsf{aux}}$, since (as made clear in the following) conditioned on w, $w^{\mathsf{x}}_{\mathsf{aux}}$ is always the same regardless of the sampled $j\in[n]$. For all $j\in[n]$, we then define the operators

$$\begin{aligned}\Phi_j(w)&:=\left(\frac{1}{n}\sum_{i\in[n]}w^{\mathsf{f_i^*}}+\mu w^{\mathsf{x}},\left\{\frac{1}{np_j}(\nabla f_j^*(w^{\mathsf{f_j^*}})-w^{\mathsf{x}})\cdot\mathbf{1}_{i=j}\right\}\right),\\ \Phi_j(w_{\mathsf{aux}}(j))&:=\left(\frac{1}{n}\sum_{i\in[n]}w^{\mathsf{f_i^*}}+\frac{1}{np_j}\left(w^{\mathsf{f_j^*}}_{\mathsf{aux}}(j)-w^{\mathsf{f_j^*}}\right)+\mu w^{\mathsf{x}}_{\mathsf{aux}},\left\{\frac{1}{np_j}\left(\nabla f_j^*\left(w^{\mathsf{f_j^*}}_{\mathsf{aux}}(j)\right)-w^{\mathsf{x}}_{\mathsf{aux}}\right)\cdot\mathbf{1}_{i=j}\right\}\right),\end{aligned} \tag{2.43}$$

Algorithm 7: FINITE-SUM-ONE-PHASE($F_{\text{fs-reg}}, w_0$): Finite sum optimization subroutine

Input: (2.37) satisfying Assumption 2, $w_0 \in \mathcal{Z}$ specified by $w_0^{\mathsf{x}}, \{w_0^{\mathsf{f_i}}\}_{i\in[n]} \in \mathcal{X}$;
Parameter(s): $\lambda \geq 2$, $S \in \mathbb{N}$;
Sample $0 \leq \sigma < S$ uniformly at random;
for $0 \leq s \leq \sigma$ **do**
  Sample $j \in [n]$ according to p defined in (2.42);
  $w_{s+1/2}^{\mathsf{x}} \leftarrow w_s^{\mathsf{x}} - \frac{1}{\lambda\mu}(\mu w_s^{\mathsf{x}} + \frac{1}{n}\sum_{i\in[n]} \nabla f_i(w_s^{\mathsf{f_i}}))$;
  $w_{s+1/2}^{\mathsf{f_j}} \leftarrow (1 - \frac{1}{\lambda n p_j}) w_s^{\mathsf{f_j}} + \frac{1}{\lambda n p_j} w_s^{\mathsf{x}}$;
  $w_{s+1/2}^{\mathsf{f_i}} \leftarrow w_s^{\mathsf{f_i}}$ for all $i \neq j$;
  $\Delta_s \leftarrow \nabla f_j(w_{s+1/2}^{\mathsf{f_j}}) - \nabla f_j(w_s^{\mathsf{f_j}})$;
  $w_{s+1}^{\mathsf{x}} \leftarrow w_s^{\mathsf{x}} - \frac{1}{\lambda\mu}(\mu w_{s+1/2}^{\mathsf{x}} + \frac{1}{n}\sum_{i\in[n]} \nabla f_i(w_s^{\mathsf{f_i}}) + \frac{1}{np_j}\Delta_s)$;
  $w_{s+1}^{\mathsf{f_j}} \leftarrow w_s^{\mathsf{f_j}} + \frac{1}{\lambda n p_j}(w_{s+1/2}^{\mathsf{x}} - w_{s+1/2}^{\mathsf{f_j}})$;
  $w_{s+1}^{\mathsf{f_i}} \leftarrow w_s^{\mathsf{f_i}}$ for all $i \neq j$;
Return: $(w_{\sigma+1/2}^{\mathsf{x}}, \{\nabla f_i((1 - \frac{1}{\lambda n p_i}) w_\sigma^{\mathsf{f_i}} + \frac{1}{\lambda n p_i} w_\sigma^{\mathsf{x}})\}_{i\in[n]})$

where $\mathbf{1}_{i=j}$ is a zero-one indicator. In other words, $\Phi_j(w)$ and $\Phi_j(w_{\mathsf{aux}}(j))$ both only have two nonzero blocks, corresponding to the $\mathcal{X}$ and j^{th} $\mathcal{X}^*$ blocks. We record the following useful observation about our randomized operators (2.43), in accordance with the first condition in (2.41). To give a brief interpretation of our "aggregate point" defined in (2.44), the $\mathcal{X}$ coordinate is updated deterministically from w^{x} according to the corresponding block of Φ, and every dual block $j \in [n]$ of $\bar{w}$ is set to the corresponding dual block had j been sampled in that step.

Lemma 19 (Expected regret)**.** *Define* $\{\Phi_j\}_{j\in[n]} : \mathcal{Z} \to \mathcal{Z}^*$ *as in* (2.43)*, and the "aggregate point"*

$$\bar{w} := \left(w_{\mathsf{aux}}^{\mathsf{x}}, \left\{ w_{\mathsf{aux}}^{\mathsf{f_j^*}}(j) \right\}_{j\in[n]} \right). \tag{2.44}$$

Then, for all $u \in \mathcal{Z}$*, defining* Φ *as in* (2.40)*,*

$$\mathbb{E}_{j\sim p}\left[\langle \Phi_j(w_{\mathsf{aux}}(j)), w_{\mathsf{aux}}(j) - u\rangle\right] = \langle \Phi(\bar{w}), \bar{w} - u\rangle .$$

Proof. We expand the expectation, using (2.43) and taking advantage of the sparsity of Φ_j:

$$\begin{aligned}
&\mathbb{E}_{j\sim p}\left[\langle \Phi_j(w_{\mathsf{aux}}(j)), w_{\mathsf{aux}}(j) - u\rangle\right] \\
&\quad= \left\langle \sum_{j\in[n]} p_j \left(\frac{1}{n}\sum_{i\in[n]} w^{\mathsf{f}_i^*} + \frac{1}{np_j}\left(w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j) - w^{\mathsf{f}_j^*}\right) + \mu w^{\mathsf{x}}_{\mathsf{aux}}\right), w^{\mathsf{x}}_{\mathsf{aux}} - u^{\mathsf{x}}\right\rangle \\
&\qquad + \sum_{j\in[n]} p_j \left\langle \frac{1}{np_j}\left(\nabla f_j^*\left(w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j)\right) - w^{\mathsf{x}}_{\mathsf{aux}}\right), w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j) - u^{\mathsf{f}_j^*}\right\rangle \\
&\quad= \left\langle \frac{1}{n}\sum_{j\in[n]} w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j) + \mu w^{\mathsf{x}}_{\mathsf{aux}}, w^{\mathsf{x}}_{\mathsf{aux}} - u^{\mathsf{x}}\right\rangle \\
&\qquad + \sum_{j\in[n]} \left\langle \frac{1}{n}\left(\nabla f_j^*\left(w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j)\right) - w^{\mathsf{x}}_{\mathsf{aux}}\right), w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j) - u^{\mathsf{y}_j}\right\rangle = \langle \Phi(\bar{w}), \bar{w} - u\rangle .
\end{aligned}$$

□

We conclude this section by demonstrating that Algorithm 7 is an appropriate implementation of Algorithm 5.

Lemma 20 (Implementation)**.** *Algorithm 7 implements Algorithm 5 on* $(\{\Phi_i\}_{i\in[n]}, r)$ *defined in* (2.43), (2.39)*, for* σ *iterations, and returns* $\bar{w}_\sigma$*, following the definition* (2.44)*. Each iteration* $s > 0$ *is implementable in* $O(1)$ *gradient calls to some* f_i*, and* $O(1)$ *vector operations on* $\mathcal{X}$*.*

Proof. Let $\{w_s, w_{s+1/2}\}_{0\le s\le\sigma}$ be the iterates of Algorithm 5. We will inductively show that Algorithm 7 preserves the invariants

$$w_s = \left(w_s^{\mathsf{x}}, \left\{\nabla f_i(w_s^{\mathsf{f}_i})\right\}_{i\in[n]}\right),\ w_{s+1/2} = \left(w_s^{\mathsf{x}}, \left\{\nabla f_i(w_{s+1/2}^{\mathsf{f}_i})\right\}_{i\in[n]}\right)$$

for all $0 \le s \le \sigma$. Once we prove this claim, it is clear from inspection that Algorithm 7 implements Algorithm 5 and returns $\bar{w}_\sigma$, upon recalling the definitions (2.43), (2.39), and (2.44).

The base case of our induction follows from the initialization guarantee of Line 3 in Algorithm 7. Next, suppose for some $0 \le s \le \sigma$, we have $w_s^{\mathsf{f}_i^*} = \nabla f(w_s^{\mathsf{f}_i})$ for all $i \in [n]$. By the updates in Algorithm 5, if $j \in [n]$ was sampled on iteration s,

$$\begin{aligned}
w^{\mathsf{f}_j^*}_{s+1/2} &\leftarrow \operatorname{argmin}_{w^{\mathsf{f}_j^*}\in\mathcal{X}^*}\left\{\frac{1}{\lambda np_j}\left\langle w_s^{\mathsf{f}_j^*} - w_s^{\mathsf{x}}, w^{\mathsf{f}_j^*}\right\rangle - \left\langle w_s^{\mathsf{f}_j^*}, w^{\mathsf{f}_j^*}\right\rangle + f_j^*\left(w^{\mathsf{f}_j^*}\right)\right\} \\
&= \operatorname{argmax}_{w^{\mathsf{f}_j^*}\in\mathcal{X}^*}\left\{\left\langle \left(1 - \frac{1}{\lambda np_j}\right) w_s^{\mathsf{f}_j^*} + \frac{1}{\lambda np_j} w_s^{\mathsf{x}}, w^{\mathsf{f}_j^*}\right\rangle - f_j^*\left(w^{\mathsf{f}_j^*}\right)\right\} \\
&= \nabla f_j\left(\left(1 - \frac{1}{\lambda np_j}\right) w_s^{\mathsf{f}_j^*} + \frac{1}{\lambda np_j} w_s^{\mathsf{x}}\right).
\end{aligned}$$

Here, we used the first item in Fact 1 in the last line. Hence, the update to $w^{\mathsf{f}_j^*}_{s+1/2}$ in Algorithm 7

preserves our invariant, and all other $w_{s+1/2}^{\mathsf{f}_i^*}$, $i \neq j$ do not change by sparsity of Φ_j. An analogous argument shows the update to each $w_{s+1}^{\mathsf{f}_i^*}$ preserves our invariant. Finally, in every iteration $s > 0$, the updates to $w_{s+1/2}^{\mathsf{x}}$ and w_{s+1}^{x} only require evaluating one new gradient each, by 1-sparsity of the dual block updates in the prior iteration. □

2.8.3 Convergence analysis

In this section, we prove a convergence result on Algorithm 7 via an application of Proposition 4. To begin, we require a bound on the quantity λ in (2.41). We remark that our derivation of this bound follows the derivation of Lemma 8 closely.

Lemma 21 (Expected relative Lipschitzness)**.** *Define $\{\Phi_j\}_{j\in[n]} : \mathcal{Z} \to \mathcal{Z}^*$ as in* (2.43)*, and define $r : \mathcal{Z} \to \mathbb{R}$ as in* (2.39)*. Letting $w_+(j)$ be w_{s+1} in Algorithm 5 if $j \in [n]$ was sampled in iteration s,*

$$\mathbb{E}_{j\sim p}\left[\langle \Phi_j(w_{\mathsf{aux}}(j)) - \Phi_j(w), w_{\mathsf{aux}}(j) - w_+(j)\rangle\right] \leq \mathbb{E}_{j\sim p}\left[V_w^r\left(w_{\mathsf{aux}}(j)\right) + V_{w_{\mathsf{aux}}(j)}^r\left(w_+(j)\right)\right]$$

for

$$\lambda = 2n + \frac{2\sum_{j\in[n]}\sqrt{L_j}}{\sqrt{n\mu}}. \tag{2.45}$$

Proof. We begin by expanding the expectation of the left-hand side:

$$\begin{aligned}
\mathbb{E}_{j\sim p}\left[\langle \Phi_j(w_{\mathsf{aux}}(j)) - \Phi_j(w), w_{\mathsf{aux}}(j) - w_+(j)\rangle\right] &= \mathbb{E}_{j\sim p}\left[\left\langle \mu w_{\mathsf{aux}}^{\mathsf{x}} - \mu w^{\mathsf{x}}, w_{\mathsf{aux}}^{\mathsf{x}} - w_+^{\mathsf{x}}(j)\right\rangle\right] \\
&+ \mathbb{E}_{j\sim p}\left[\frac{1}{np_j}\left\langle \nabla f_j^*\left(w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j)\right) - \nabla f_j^*\left(w^{\mathsf{f}_j^*}\right), w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w_+^{\mathsf{f}_j^*}(j)\right\rangle\right] \\
&+ \mathbb{E}_{j\sim p}\left[\frac{1}{np_j}\left\langle w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w^{\mathsf{f}_j^*}, w_{\mathsf{aux}}^{\mathsf{x}} - w_+^{\mathsf{x}}(j)\right\rangle\right] \\
&+ \mathbb{E}_{j\sim p}\left[\frac{1}{np_j}\left\langle w^{\mathsf{x}} - w_{\mathsf{aux}}^{\mathsf{x}}, w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w_+^{\mathsf{f}_j^*}(j)\right\rangle\right].
\end{aligned} \tag{2.46}$$

To bound the first two lines of (2.46), fix some $j \in [n]$. We apply Lemma 1 to the functions $\frac{\mu}{2}\|\cdot\|^2$ and $\frac{1}{n}\nabla f_j^*$, and use nonnegativity of Bregman divergences, to conclude

$$\begin{aligned}
&\left\langle \mu w_{\mathsf{aux}}^{\mathsf{x}} - \mu w^{\mathsf{x}}, w_{\mathsf{aux}}^{\mathsf{x}} - w_+^{\mathsf{x}}(j)\right\rangle + \frac{1}{np_j}\left\langle \nabla f_j^*\left(w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j)\right) - \nabla f_j^*\left(w^{\mathsf{f}_j^*}\right), w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w_+^{\mathsf{f}_j^*}(j)\right\rangle \\
&\qquad \leq 2n\left(V_w^r\left(w_{\mathsf{aux}}(j)\right) + V_{w_{\mathsf{aux}}(j)}^r\left(w_+(j)\right)\right).
\end{aligned}$$

In particular, we used $\frac{1}{p_j} \leq 2n$ by assumption, and noted we only need to handle the case where the second inner product term above is positive (in the other case, the above inequality is clearly true). Hence, taking expectations the first two lines in (2.46) contribute $2n$ to λ in the final bound.

To bound the last two lines of (2.46), fix $j \in [n]$. By applying Item 1 in Lemma 13 to the pair

$(\frac{\mu}{2}\|\cdot\|^2, nf_i)$, we have

$$\begin{aligned}
&\frac{1}{n}\left\langle w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w^{\mathsf{f}_j^*}, w_{\mathsf{aux}}^{\mathsf{x}} - w_+^{\mathsf{x}}(j)\right\rangle + \frac{1}{n}\left\langle w^{\mathsf{x}} - w_{\mathsf{aux}}^{\mathsf{x}}, w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w_+^{\mathsf{f}_j^*}(j)\right\rangle \\
&\leq \frac{1}{n}\sqrt{\frac{nL_j}{\mu}}\left(\mu V_{w_{\mathsf{aux}}^{\mathsf{x}}}\left(w_+^{\mathsf{x}}(j)\right) + V_{w^{\mathsf{f}_j^*}}^{f_j^*}\left(w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j)\right)\right) + \frac{1}{n}\sqrt{\frac{nL_j}{\mu}}\left(\mu V_{w^{\mathsf{x}}}\left(w_{\mathsf{aux}}^{\mathsf{x}}\right) + V_{w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j)}^{f_j^*}\left(w_+^{\mathsf{f}_j^*}(j)\right)\right) \\
&= \sqrt{\frac{L_j}{n\mu}}\left(V_w^r\left(w_{\mathsf{aux}}(j)\right) + V_{w_{\mathsf{aux}}(j)}^r\left(w_+(j)\right)\right).
\end{aligned}$$

Using $\frac{\sqrt{L_i}}{p_i} \leq 2\sum_{j\in[n]}\sqrt{L_j}$ and taking expectations over the above display,

$$\begin{aligned}
&\mathbb{E}_{j\sim p}\left[\frac{1}{np_j}\left\langle w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w^{\mathsf{f}_j^*}, w_{\mathsf{aux}}^{\mathsf{x}} - w_+^{\mathsf{x}}(j)\right\rangle + \frac{1}{np_j}\left\langle w^{\mathsf{x}} - w_{\mathsf{aux}}^{\mathsf{x}}, w_{\mathsf{aux}}^{\mathsf{f}_j^*}(j) - w_+^{\mathsf{f}_j^*}(j)\right\rangle\right] \\
&\leq \frac{2\sum_{j\in[n]}\sqrt{L_j}}{\sqrt{n\mu}}\mathbb{E}_{j\sim p}\left[V_w^r\left(w_{\mathsf{aux}}(j)\right) + V_{w_{\mathsf{aux}}(j)}^r\left(w_+(j)\right)\right].
\end{aligned}$$

Hence, the last two lines in (2.46) contribute $\frac{2\sum_{j\in[n]}\sqrt{L_j}}{\sqrt{n\mu}}$ to λ in the final bound. □

We next apply Proposition 4 to analyze the convergence of Algorithm 7.

Lemma 22. *Let $w_0 := (w_0^{\mathsf{x}}, \{\nabla f_i(w_0^{\mathsf{f}_i})\}_{i\in[n]})$, which is the input z_t to Algorithm 7 at iteration t. If $S \geq 2\lambda$ in Algorithm 7 with λ as in* (2.45)*, then Algorithm 7 returns $\widetilde{w} \leftarrow \bar{w}_\sigma$ as defined in* (2.44) *such that for $z_\star$ as the saddle point to* (2.38)*,*

$$\mathbb{E}V_{\widetilde{w}}^r(z_\star) \leq \frac{1}{2}V_{w_0}^r(z_\star).$$

Proof. We apply Proposition 4, where (2.41) is satisfied via Lemmas 19 and 21. By Proposition 4 with $u = z_\star$ and $S \geq 2\lambda$,

$$\mathbb{E}\left[\frac{1}{S}\sum_{0\leq s<S}\langle\Phi(\bar{w}_s), \bar{w}_s - z_\star\rangle\right] \leq \frac{1}{2}V_{w_0}^r(z_\star).$$

Moreover, since σ is uniformly chosen in $[0, S-1]$, we have

$$\mathbb{E}\left[\langle\Phi(\bar{w}_\sigma), \bar{w}_\sigma - z_\star\rangle\right] \leq \frac{1}{2}V_{w_0}^r(z_\star).$$

Finally, Lemma 20 shows that (an implicit representation of) $\bar{w}_\sigma$ is indeed returned. We conclude by applying Lemma 18 and using that $z_\star$ solves the VI in Φ, yielding

$$\mathbb{E}\left[\langle\Phi(\bar{w}_\sigma), \bar{w}_\sigma - z_\star\rangle\right] \geq \mathbb{E}\left[\langle\Phi(\bar{w}_\sigma) - \Phi(z_\star), \bar{w}_\sigma - z_\star\rangle\right] \geq V_{\bar{w}_\sigma}^r(z_\star).$$

□

Finally, we provide a simple bound regarding initialization of Algorithm 6.

Lemma 23. *Let $x_0 \in \mathcal{X}$, and define*

$$z_0 := \left(x_0, \{\nabla f_i(x_0)\}_{i\in[n]}\right). \tag{2.47}$$

Moreover, suppose that for $x_\star$ the solution to (2.37), $F_{\text{fs-reg}}(x_0) - F_{\text{fs-reg}}(x_\star) \le \epsilon_0$. Then, letting $z_\star$ be the solution to (2.38), we have

$$V_{z_0}^r(z_\star) \le \left(1 + \frac{\sum_{i\in[n]} L_i}{n\mu}\right)\epsilon_0.$$

Proof. By the characterization in Lemma 17, we have by Item 1 in Fact 1:

$$z_\star = \left(x_\star, \{\nabla f_i(x_\star)\}_{i\in[n]}\right).$$

Hence, we bound analogously to Lemma 15:

$$\begin{aligned} V_{z_0}^r(z_\star) &\le \mu V_{x_0}(x_\star) + V_{x_\star}^{\frac{1}{n}\sum_{i\in[n]} f_i}(x_0) \\ &\le \mu V_{x_0}(x_\star) + \frac{\sum_{i\in[n]} L_i}{2n}\|x_0 - x_\star\|^2 \\ &\le \left(1 + \frac{\sum_{i\in[n]} L_i}{n\mu}\right)\mu V_{x_0}(x_\star) \le \left(1 + \frac{\sum_{i\in[n]} L_i}{n\mu}\right)\epsilon_0. \end{aligned}$$

The last line applied strong convexity of $F_{\text{fs-reg}}$. □

2.8.4 Main result

We now state and prove our main claim.

Theorem 8. *Suppose $F_{\text{fs-reg}}$ satisfies Assumption 2 and has minimizer $x_\star$, and suppose we have $x_0 \in \mathcal{X}$ such that $F_{\text{fs-reg}}(x_0) - F_{\text{fs-reg}}(x_\star) \le \epsilon_0$. Algorithm 6 using Algorithm 7 with λ as in (2.45) returns $x \in \mathcal{X}$ with $\mathbb{E}F_{\text{fs-reg}}(x) - F_{\text{fs-reg}}(x_\star) \le \epsilon$ in N_{tot} iterations, using a total of $O(N_{\text{tot}})$ gradient calls each to some f_i for $i \in [n]$, where*

$$N_{\text{tot}} = O\left(\kappa_{\text{fs}} \log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\text{fs}} := n + \frac{\sum_{i\in[n]}\sqrt{L_i}}{\sqrt{n\mu}}. \tag{2.48}$$

Proof. By Lemma 17, the point $x_\star$ is consistent between (2.36) and (2.38). We run Algorithm 6 with

$$T = O\left(\log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right).$$

By recursively applying Lemma 22 for T times, we obtain a point z such that

$$\mathbb{E}V_z^r(z_\star) \le \frac{\epsilon\mu}{\mathcal{L}} \text{ for } \mathcal{L} = \mu + \frac{1}{n}\sum_{i\in[n]} L_i,$$

and hence applying $\mathcal{L}$-smoothness of $F_{\text{fs-reg}}$ and optimality of $z_\star^{\mathsf{x}}$ yields the claim. The complexity follows from Lemma 12, and spending $O(n)$ gradient evaluations on the first and last iterates of each call to Algorithm 7 (which is subsumed by the fact that $S = \Omega(n)$). □

By applying a generic reduction we derive in Appendix A.8, we then obtain the following corollary.

Corollary 4. *Suppose the summands $\{f_i\}_{i\in[n]}$ in (2.36) satisfy Assumption 2, and F_{fs} is μ-strongly convex with minimizer $x_\star$. Further, suppose we have $x_0 \in \mathcal{X}$ such that $F_{\text{fs}}(x_0) - F_{\text{fs}}(x_\star) \le \epsilon_0$. Algorithm 66 using Algorithm 6 to implement steps returns $x \in \mathcal{X}$ with $\mathbb{E}F_{\text{fs}}(x) - F_{\text{fs}}(x_\star) \le \epsilon$ in N_{tot} iterations, using a total of $O(N_{\text{tot}})$ gradient calls each to some f_i for $i \in [n]$, where*

$$N_{\text{tot}} = O\left(\kappa_{\text{fs}} \log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\text{fs}} := n + \sum_{i\in[n]} \frac{\sqrt{L_i}}{\sqrt{n\mu}}.$$

2.9 Minimax finite sum optimization

In this section, we provide efficient algorithms for computing an approximate saddle point of the following minimax finite sum optimization problem:

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} F_{\text{mmfs}}(x,y) := \frac{1}{n}\sum_{i\in[n]} \left(f_i(x) + h_i(x,y) - g_i(y)\right). \tag{2.49}$$

Here and throughout this section $\{f_i : \mathcal{X} \to \mathbb{R}\}_{i\in[n]}$, $\{g_i : \mathcal{Y} \to \mathbb{R}\}_{i\in[n]}$ are differentiable convex functions, and $\{h_i : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}\}_{i\in[n]}$ are differentiable convex-concave functions. For the remainder, we focus on algorithms for solving the following regularized formulation of (2.49):

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} F_{\text{mmfs-reg}}(x,y) := \frac{1}{n}\sum_{i\in[n]} \left(f_i(x) + h_i(x,y) - g_i(y)\right) + \frac{\mu^{\mathsf{x}}}{2}\|x\|^2 - \frac{\mu^{\mathsf{y}}}{2}\|y\|^2. \tag{2.50}$$

As in Section 2.7 and Section 2.8, to instead solve an instance of (2.49) where each f_i is $2\mu^{\mathsf{x}}$-strongly convex and each g_i is $2\mu^{\mathsf{y}}$-strongly convex, we may instead equivalently solve (2.50) by reparameterizing $f_i \leftarrow f_i - \mu^{\mathsf{x}}\|\cdot\|^2$, $g_i \leftarrow g_i - \mu^{\mathsf{y}}\|\cdot\|^2$ for each $i \in [n]$. The extra factor of 2 is so we can make a strong convexity assumption in Assumption 3 about separable summands, which only affects our final bounds by constants. We further remark that our algorithms extend to solve instances of (2.49) where f, g is μ^{x} and μ^{y}-strongly convex in $\|\cdot\|$, but individual summands are not. We provide this result at the end of the section in Corollary 5.

In designing methods for solving (2.50) we make the following additional regularity assumptions.

Assumption 3. *We assume the following about* (2.50) *for all* $i \in [n]$.

1. f_i *is* L_i^{x}*-smooth and* μ_i^{x}*-strongly convex and* g_i *is* L_i^{y}*-smooth and* μ_i^{y}*-strongly convex.*
2. h_i *has the following blockwise-smoothness properties: for all* $u, v \in \mathcal{X} \times \mathcal{Y}$,

$$\begin{aligned} \|\nabla_x h_i(u) - \nabla_x h_i(v)\| &\leq \Lambda_i^{\mathsf{xx}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda_i^{\mathsf{xy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\| \quad \textit{and} \\ \|\nabla_y h_i(u) - \nabla_y h_i(v)\| &\leq \Lambda_i^{\mathsf{xy}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda_i^{\mathsf{yy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\|. \end{aligned} \tag{2.51}$$

The remainder of this section is organized as follows.

1. In Section 2.9.1, we state a primal-dual formulation of (2.50) which we will apply our methods to, and prove that its solution also yields a solution to (2.50).
2. In Section 2.9.2, we give our algorithm, which is composed of an outer loop and an inner loop, and prove it is efficiently implementable.
3. In Section 2.9.3, we prove the convergence rate of our inner loop.
4. In Section 2.9.4, we prove the convergence rate of our outer loop.
5. In Section 2.9.5, we state and prove our main result, Theorem 9.

2.9.1 Setup

To solve (2.50), we will instead find a saddle point to the primal-dual function

$$\begin{aligned} F_{\text{mmfs-pd}}(z) := \frac{\mu^{\mathsf{x}}}{2} \|z^{\mathsf{x}}\|^2 - \frac{\mu^{\mathsf{y}}}{2} \|z^{\mathsf{y}}\|^2 \\ + \frac{1}{n} \sum_{i \in [n]} \left(h_i(z^{\mathsf{x}}, z^{\mathsf{y}}) + \left\langle z^{\mathsf{f_i^*}}, z^{\mathsf{x}} \right\rangle - \left\langle z^{\mathsf{g_i^*}}, z^{\mathsf{y}} \right\rangle - f_i^* \left(z^{\mathsf{f_i^*}} \right) + g_i^*(z^{\mathsf{g_i^*}}) \right). \end{aligned} \tag{2.52}$$

We denote the domain of $F_{\text{mmfs-pd}}$ by $\mathcal{Z} := \mathcal{X} \times \mathcal{Y} \times (\mathcal{X}^*)^n \times (\mathcal{Y}^*)^n$. For $z \in \mathcal{Z}$, we refer to its blocks by $(z^{\mathsf{x}}, z^{\mathsf{y}}, \{z^{\mathsf{f_i^*}}\}_{i \in [n]}, \{z^{\mathsf{g_i^*}}\}_{i \in [n]})$. The primal-dual function $F_{\text{mmfs-pd}}$ is related to the original function F_{mmfs} in the following way; we omit the proof, as it follows analogously to the proofs of Lemmas 10 and 17.

Lemma 24. *Let* $z_\star = (z_\star^{\mathsf{x}}, z_\star^{\mathsf{y}}, \{z_\star^{\mathsf{f_i^*}}\}_{i \in [n]}, \{z_\star^{\mathsf{g_i^*}}\}_{i \in [n]})$ *be the saddle point to* (2.52)*. Then,* $(z_\star^{\mathsf{x}}, z_\star^{\mathsf{y}})$ *is a saddle point to* (2.50)*.*

As in Section 2.7.1, it will be convenient to define the convex function $r : \mathcal{Z} \to \mathbb{R}$, which combines the (unsigned) separable components of $F_{\text{mmfs-pd}}$:

$$r\left(z^{\mathsf{x}}, z^{\mathsf{y}}, \left\{z^{\mathsf{f_i^*}}\right\}_{i \in [n]}, \left\{z^{\mathsf{g_i^*}}\right\}_{i \in [n]}\right) := \frac{\mu^{\mathsf{x}}}{2} \|z^{\mathsf{x}}\|^2 + \frac{\mu^{\mathsf{y}}}{2} \|z^{\mathsf{y}}\|^2 + \frac{1}{n} \sum_{i \in [n]} f_i^* \left(z^{\mathsf{f_i^*}}\right) + \frac{1}{n} \sum_{i \in [n]} g_i^* \left(z^{\mathsf{g_i^*}}\right). \tag{2.53}$$

Again, r serves as a regularizer in our algorithm. We next define $\Phi^{\text{mmfs-pd}}$, the gradient operator of $F_{\text{mmfs-pd}}$. We decompose $\Phi^{\text{mmfs-pd}}$ into three parts, roughly corresponding to the contribution from r, the contributions from the primal-dual representations of $\{f_i\}_{i\in[n]}$ and $\{g_i\}_{i\in[n]}$, and the contribution from $\{h_i\}_{i\in[n]}$. In particular, we define

$$\begin{aligned}
\Phi^{\text{mmfs-pd}}(z) &:= \nabla r(z) + \Phi^h(z) + \Phi^{\text{bilin}}(z), \\
\nabla r\left(z^{\mathsf{x}}, z^{\mathsf{y}}, \left\{z^{\mathsf{f}_\mathsf{i}^*}\right\}_{i\in[n]}, \left\{z^{\mathsf{g}_\mathsf{i}^*}\right\}_{i\in[n]}\right) &:= \left(\mu^{\mathsf{x}} z^{\mathsf{x}}, \mu^{\mathsf{y}} z^{\mathsf{y}}, \left\{\frac{1}{n}\nabla f_i^*\left(z^{\mathsf{f}_\mathsf{i}^*}\right)\right\}_{i\in[n]}, \left\{\frac{1}{n}\nabla g_i^*\left(z^{\mathsf{g}_\mathsf{i}^*}\right)\right\}_{i\in[n]}\right), \\
\Phi^h\left(z^{\mathsf{x}}, z^{\mathsf{y}}, \left\{z^{\mathsf{f}_\mathsf{i}^*}\right\}_{i\in[n]}, \left\{z^{\mathsf{g}_\mathsf{i}^*}\right\}_{i\in[n]}\right) &:= \left(\frac{1}{n}\sum_{i\in[n]} \nabla_x h_i(z^{\mathsf{x}}, z^{\mathsf{y}}), -\frac{1}{n}\sum_{i\in[n]} \nabla_y h_i(z^{\mathsf{x}}, z^{\mathsf{y}}), \{0\}_{i\in[n]}, \{0\}_{i\in[n]}\right), \\
\Phi^{\text{bilin}}\left(z^{\mathsf{x}}, z^{\mathsf{y}}, \left\{z^{\mathsf{f}_\mathsf{i}^*}\right\}_{i\in[n]}, \left\{z^{\mathsf{g}_\mathsf{i}^*}\right\}_{i\in[n]}\right) &:= \left(\frac{1}{n}\sum_{i\in[n]} z^{\mathsf{f}_\mathsf{i}^*}, \frac{1}{n}\sum_{i\in[n]} z^{\mathsf{g}_\mathsf{i}^*}, \left\{-\frac{1}{n}z^{\mathsf{x}}\right\}_{i\in[n]}, \left\{-\frac{1}{n}z^{\mathsf{y}}\right\}_{i\in[n]}\right).
\end{aligned} \tag{2.54}$$

2.9.2 Algorithm

In this section we present our algorithm which consists of the following two parts; its design is inspired by a similar strategy used in prior work [111, 112].

1. Our "outer loop" is based on a proximal point method (Algorithm 8, adapted from [415]).
2. Our "inner loop" solves each proximal subproblem to high accuracy via a careful analysis of randomized mirror prox (Algorithm 9, adapted from Algorithm 5).

At each iteration t of the outer loop (Algorithm 8), we require an accurate approximation

$$z_{t+1} \approx z_{t+1}^\star \quad \text{which solves the VI in} \quad \Phi := \Phi^{\text{mmfs-pd}}(z) + \gamma\left(\nabla r(z) - \nabla r(z_t)\right), \tag{2.55}$$

where we recall the definitions of g_{tot} and r from (2.54) and (2.53), and when z_t is clear from context (i.e. we are analyzing a single implementation of the inner loop).

To implement our inner loop (i.e. solve the VI in Φ), we apply randomized mirror prox (Algorithm 5) with a new analysis. In particular, we will not be able to obtain the expected relative Lipschitzness bound required by Proposition 4 for our randomized gradient estimators, so we develop a new "partial variance" analysis of Algorithm 5 to obtain our rate. We use this terminology because we use variance bounds on a component of Φ for which we cannot directly obtain expected relative Lipschitzness bounds. We prove Proposition 5 in Appendix A.10.1.

Proposition 5 (Partial variance analysis of randomized mirror prox)**.** *Suppose (possibly random) $\widetilde{\Phi}$ is defined so that in each iteration s, for all $u \in \mathcal{Z}$ and all $\rho > 0$, there exists a (possibly random)*

point $\bar{w}_s \in \mathcal{Z}$ and a γ-strongly monotone operator $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ (with respect to r) such that

$$\begin{aligned}\mathbb{E}\left[\left\langle \widetilde{\Phi}(w_{s+1/2}), w_{s+1/2} - w_\star \right\rangle\right] &= \mathbb{E}\left[\langle \Phi(\bar{w}_s), \bar{w}_s - w_\star \rangle\right], \\ \mathbb{E}\left[\left\langle \widetilde{\Phi}(w_{s+1/2}) - \widetilde{\Phi}(w_s), w_{s+1/2} - w_{s+1} \right\rangle\right] &\le \left(\lambda_0 + \frac{1}{\rho}\right) \mathbb{E}\left[V^r_{w_s}(w_{s+1/2}) + V^r_{w_{s+1/2}}(w_{s+1})\right] \\ &\quad + \rho\lambda_1 \mathbb{E}\left[V^r_{w_0}(w_\star) + V^r_{\bar{w}_s}(w_\star)\right],\end{aligned} \tag{2.56}$$

where $w_\star$ solves the VI in Φ. Then by setting

$$\rho \leftarrow \frac{\gamma}{5\lambda_1},\ \lambda \leftarrow \lambda_0 + \frac{1}{\rho},\ T \leftarrow \frac{5\lambda}{\gamma} = \frac{5\lambda_0}{\gamma} + \frac{25\lambda_1}{\gamma^2},$$

in Algorithm 5, and returning $\bar{w}_\sigma$ for $0 \le \sigma < S$ sampled uniformly at random,

$$\mathbb{E}\left[V^r_{\bar{w}_\sigma}(w_\star)\right] \le \frac{1}{2} V^r_{w_0}(w_\star).$$

For simplicity in the following we denote $\bar{z} := z_t$ whenever we discuss a single proximal subproblem. We next introduce the gradient estimator $\widetilde{\Phi}$ we use in each inner loop, i.e. finding a solution to the VI in Φ defined in (2.55). We first define three sampling distributions p, q, r, via

$$\begin{aligned}&p_j := \frac{\sqrt{L^{\mathsf{x}}_j}}{2\sum_{i\in[n]} \sqrt{L^{\mathsf{x}}_i}} + \frac{1}{2n} \text{ for all } j \in [n], \quad q_k := \frac{\sqrt{L^{\mathsf{y}}_k}}{2\sum_{i\in[n]} \sqrt{L^{\mathsf{y}}_i}} + \frac{1}{2n} \text{ for all } k \in [n], \\ &\text{and } r_\ell := \frac{\Lambda^{\text{tot}}_\ell}{2\sum_{i\in[n]} \Lambda^{\text{tot}}_i} + \frac{1}{2n} \text{ for all } \ell \in [n], \text{ where } \Lambda^{\text{tot}}_i := \frac{\Lambda^{\mathsf{xx}}_i}{\mu^{\mathsf{x}}} + \frac{\Lambda^{\mathsf{xy}}_i}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda^{\mathsf{yy}}_i}{\mu^{\mathsf{y}}} \text{ for all } i \in [n].\end{aligned} \tag{2.57}$$

Algorithm 9 will run in logarithmically many phases, each initialized at an "anchor point" w_0 (cf. Line 18). We construct gradient estimators for Algorithm 5 of $\Phi(w) = \Phi^{\text{mmfs-pd}}(w) + \gamma(\nabla r(w) - \nabla r(\bar{z}))$ as defined in (2.55) as follows. In each iteration, for a current anchor point w_0, we sample four coordinates $j \sim p$, $k \sim q$, and $\ell, \ell' \sim r$, all independently. We believe that it is likely that other sampling schemes, e.g. sampling j and k non-independently, will also suffice for our method but focus on the independent scheme for simplicity. We use g^{xy} to refer to the $\mathcal{X} \times \mathcal{Y}$ blocks of a vector g in $\mathcal{Z}^*$, and $^{\mathsf{f}^*\mathsf{g}^*}$ to refer to all other blocks corresponding to $(\mathcal{X}^*)^n \times (\mathcal{Y}^*)^n$. Then we define

for an iterate $w = w_s$ of Algorithm 9 (where Φ^h is as in (2.54)):

$$\begin{aligned}
\widetilde{\Phi}(w) &:= \Phi_{jk\ell}(w) := \Phi^h_{jk\ell}(w) + \Phi^{\text{sep}}_{jk\ell}(w) + \Phi^{\text{bilin}}_{jk\ell}(w), \\
\left[\Phi^h_{jk\ell}(w)\right]^{\mathsf{x}} &:= \left[\Phi^h(w_0)\right]^{\mathsf{x}} + \frac{1}{nr_\ell}\left(\nabla_x h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_x h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})\right), \\
\left[\Phi^h_{jk\ell}(w)\right]^{\mathsf{y}} &:= \left[\Phi^h(w_0)\right]^{\mathsf{y}} - \frac{1}{nr_\ell}\left(\nabla_y h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_y h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})\right), \\
\left[\Phi^h_{jk\ell}(w)\right]^{\mathsf{f}^*\mathsf{g}^*} &:= \left(\{0\}_{i\in[n]}, \{0\}_{i\in[n]}\right), \\
\left[\Phi^{\text{sep}}_{jk\ell}(w)\right]^{\mathsf{xy}} &:= (1+\gamma)\left(\mu^{\mathsf{x}} w^{\mathsf{x}}, \mu^{\mathsf{y}} w^{\mathsf{y}}\right) - \gamma\left(\mu^{\mathsf{x}} \bar{z}^{\mathsf{x}}, \mu^{\mathsf{y}} \bar{z}^{\mathsf{y}}\right), \\
\left[\Phi^{\text{sep}}_{jk\ell}(w)\right]^{\mathsf{f}^*\mathsf{g}^*} &:= (1+\gamma)\left(\left\{\frac{1}{np_j}\nabla f_j^*\left(w^{\mathsf{f}_j^*}\right)\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{\frac{1}{nq_k}\nabla g_k^*\left(w^{\mathsf{f}_k^*}\right)\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right) \\
&\quad - \gamma\left(\left\{\frac{1}{np_j}\nabla f_j^*\left(\bar{z}^{\mathsf{f}_j^*}\right)\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{\frac{1}{nq_k}\nabla g_k^*\left(\bar{z}^{\mathsf{g}_k^*}\right)\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right), \\
\left[\Phi^{\text{bilin}}_{jk\ell}(w)\right]^{\mathsf{xy}} &:= \left(\frac{1}{n}\sum_{i\in[n]} w^{\mathsf{f}_i^*}, \frac{1}{n}\sum_{i\in[n]} w^{\mathsf{g}_i^*}\right), \\
\left[\Phi^{\text{bilin}}_{jk\ell}(w)\right]^{\mathsf{f}^*\mathsf{g}^*} &:= \left(\left\{-\frac{1}{np_j} w^{\mathsf{x}}\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{-\frac{1}{nq_k} w^{\mathsf{y}}\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right).
\end{aligned} \tag{2.58}$$

In particular, the estimator $\Phi_{jk\ell}(w)$ only depends on the sampled indices j, k, ℓ, and not ℓ'. Next, consider taking the step $w_{\mathsf{aux}}(jk\ell) \leftarrow \text{Prox}^r_w(\frac{1}{\lambda} g_{jk\ell}(w))$ as in Algorithm 5, where we use the shorthand $w_{\mathsf{aux}}(jk\ell) = w_{s+1/2}$ to indicate the iterate of Algorithm 5 taken from w_s assuming j, k, ℓ were sampled. Observing the form of $g_{jk\ell}$, we denote the blocks of $w_{\mathsf{aux}}(jk\ell)$ by

$$w_{\mathsf{aux}}(jk\ell) := \left(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell), \left\{w^{\mathsf{f}_i^*}_{\mathsf{aux}}(j)\right\}_{i\in[n]}, \left\{w^{\mathsf{g}_i^*}_{\mathsf{aux}}(k)\right\}_{i\in[n]}\right),$$

where we write $w^{\mathsf{x}}_{\mathsf{aux}}(\ell)$ to indicate that it only depends on the random choice of ℓ (and not j or k); we use similar notation for the other blocks. We also define

$$\Delta^{\mathsf{x}}(j) := w^{\mathsf{f}_j^*}_{\mathsf{aux}}(j) - w^{\mathsf{f}_j^*}(j),\ \Delta^{\mathsf{y}}(k) := w^{\mathsf{g}_k^*}_{\mathsf{aux}}(k) - w^{\mathsf{g}_k^*}(k),$$

and then set (where we use the notation $\Phi_{jk\ell'}$ to signify its dependence on j, k, ℓ', and not ℓ):

$$\begin{aligned}
\widetilde{\Phi}(w_{\mathsf{aux}}(jk\ell)) &:= \Phi_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) := \Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) + \Phi^{\text{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) + \Phi^{\text{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)),\\
\left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{x}} &:= \left[\Phi^h(w_0)\right]^{\mathsf{x}} + \frac{1}{nr_{\ell'}}\left(\nabla_x h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_x h_{\ell'}(w^{\mathsf{x}}_0, w^{\mathsf{y}}_0)\right),\\
\left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{y}} &:= \left[\Phi^h(w_0)\right]^{\mathsf{y}} - \frac{1}{nr_{\ell'}}\left(\nabla_y h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_y h_{\ell'}(w^{\mathsf{x}}_0, w^{\mathsf{y}}_0)\right),\\
\left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{f}^*\mathsf{g}^*} &:= \left(\{0\}_{i\in[n]}, \{0\}_{i\in[n]}\right),\\
\left[\Phi^{\text{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{xy}} &:= (1+\gamma)\left(\mu^{\mathsf{x}} w^{\mathsf{x}}_{\mathsf{aux}}(\ell), \mu^{\mathsf{y}} w^{\mathsf{y}}_{\mathsf{aux}}(\ell)\right) - \gamma\left(\mu^{\mathsf{x}}\bar{z}^{\mathsf{x}}, \mu^{\mathsf{y}}\bar{z}^{\mathsf{y}}\right),\\
\left[\Phi^{\text{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{f}^*\mathsf{g}^*} &:= (1+\gamma)\left(\left\{\frac{1}{np_j}\nabla f^*_j\left(w^{\mathsf{f}^*_j}_{\mathsf{aux}}\right)\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{\frac{1}{nq_k}\nabla g^*_k\left(w^{\mathsf{f}^*_k}_{\mathsf{aux}}\right)\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right)\\
&\quad - \gamma\left(\left\{\frac{1}{np_j}\nabla f^*_j\left(\bar{z}^{\mathsf{f}^*_j}\right)\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{\frac{1}{nq_k}\nabla g^*_k\left(\bar{z}^{\mathsf{g}^*_k}\right)\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right),\\
\left[\Phi^{\text{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{xy}} &:= \left(\frac{1}{n}\sum_{i\in[n]} w^{\mathsf{f}^*_i} + \frac{1}{np_j}\Delta^{\mathsf{x}}(j), \frac{1}{n}\sum_{i\in[n]} w^{\mathsf{g}^*_i} + \frac{1}{nq_k}\Delta^{\mathsf{y}}(k)\right),\\
\left[\Phi^{\text{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{f}^*\mathsf{g}^*} &:= \left(\left\{-\frac{1}{np_j}w^{\mathsf{x}}_{\mathsf{aux}}(\ell)\cdot \mathbf{1}_{i=j}\right\}_{i\in[n]}, \left\{-\frac{1}{nq_k}w^{\mathsf{y}}_{\mathsf{aux}}(\ell)\cdot \mathbf{1}_{i=k}\right\}_{i\in[n]}\right).
\end{aligned} \tag{2.59}$$

We also define the random "aggregate point" we will use in Proposition 5:

$$\bar{w}(\ell) := w + \left(w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}, w^{\mathsf{y}}_{\mathsf{aux}}(\ell) - w^{\mathsf{y}}, \{\Delta^{\mathsf{x}}(j)\}_{j\in[n]}, \{\Delta^{\mathsf{y}}(k)\}_{k\in[n]}\right). \tag{2.60}$$

Notably, $\bar{w}(\ell)$ depends only on the randomly sampled ℓ. We record the following useful observation about our randomized operators (2.58), (2.59), in accordance with the first condition in (2.56).

Lemma 25. *Define $\{\Phi_{jk\ell}, \Phi_{jk\ell'}\} : \mathcal{Z} \to \mathcal{Z}^*$ as in (2.58), (2.59), and the random "aggregate point" $\bar{w}(\ell)$ as in (2.60). Then, for all $u \in \mathcal{Z}$, recalling the definition of $\Phi = \Phi^{\text{mmfs-pd}} + \gamma(\nabla r - \nabla r(\bar{z}))$ from (2.55),*

$$\mathbb{E}\left[\langle \Phi_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)), w_{\mathsf{aux}}(jk\ell) - u\rangle\right] = \mathbb{E}_{\ell\sim r}\left[\langle \Phi(\bar{w}(\ell)), \bar{w}(\ell) - u\rangle\right].$$

We prove Lemma 25 in Appendix A.10.1. Finally, we give a complete implementation of our method as pseudocode below in Algorithms 8 (the outer loop) and 9 (the inner loop). We also show that it is a correct implementation in the following Lemma 26, which we prove in Appendix A.10.1.

Lemma 26. *Lines 6 to 18 of Algorithm 9 implement Algorithm 5 on $(\{\widetilde{\Phi}\}, r)$ defined in (2.58), (2.59), (2.53), for σ iterations, and returns $\bar{w}_\sigma$, following the definition (2.60). Each iteration $s > 0$ is implementable in $O(1)$ gradient calls to some $\{f_j, g_k, h_l\}$, and $O(1)$ vector operations on $\mathcal{X}$ and $\mathcal{Y}$.*

Algorithm 8: MINIMAX-FINITESUM-SOLVE($F_{\text{mmfs-reg}}, x_0, y_0$): Minimax finite sum optimization

Input: (2.50) satisfying Assumption 3, $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$
Parameter(s): $T \in \mathbb{N}$
$z_0^{\mathsf{x}} \leftarrow x_0$, $z_0^{\mathsf{y}} \leftarrow y_0$, $z_0^{\mathsf{f_i}} \leftarrow x_0, z_0^{\mathsf{f_i}^*} \leftarrow \nabla f_i(x_0), z_0^{\mathsf{g_i}} \leftarrow y_0, z_0^{\mathsf{g_i}^*} \leftarrow \nabla g_i(y_0)$ for all $i \in [n]$
for $0 \le t < T$ **do**
  $z_{t+1} \leftarrow$ MINIMAX-FINITESUM-INNER($F_{\text{mmfs-reg}}, \{z_t^{\mathsf{x}}, z_t^{\mathsf{y}}, \{z_t^{\mathsf{f_i}}\}_{i\in[n]}, \{z_t^{\mathsf{g_i}}\}_{i\in[n]}\}$)
Return: $(z_T^{\mathsf{x}}, z_T^{\mathsf{y}})$

2.9.3 Inner loop convergence analysis

We give a convergence guarantee on Algorithm 9 for solving the VI in $\Phi := g_{\text{tot}} + \gamma(\nabla r - \nabla r(\bar{z}))$. In order to use Proposition 5 to solve our problem, we must prove strong monotonicity of Φ and specify the parameters λ_0, λ_1 and ρ in (2.56); note that Lemma 25 handles the first condition in (2.56). To this end we give the following properties on Φ, $\widetilde{\Phi}$ as defined in (2.58) and (2.59); proofs of Lemmas 28 and 29 are deferred to Appendix A.10.2.

Strong monotonicity. We begin by proving strong monotonicity of Φ.

Lemma 27 (Strong monotonicity)**.** *Define $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ as in* (2.55)*, and define $r : \mathcal{Z} \to \mathbb{R}$ as in* (2.53)*. Then Φ is $(1+\gamma)$-strongly monotone with respect to r.*

Proof. We decompose $\Phi(z) = (1+\gamma)\nabla r(z) + \Phi^{\text{bilin}}(z) + \Phi^h(z) - \gamma\nabla r(\bar{z})$, using the definitions in (2.54). By a similar argument as Lemma 11, we obtain the claim. □

Expected relative Lipschitzness. We next provide bounds on the components of (2.56) corresponding to Φ^{sep} and Φ^{bilin}, where we use the shorthand $\Phi^{\text{sep}} := (1+\gamma)\nabla r - \gamma\nabla r(\bar{z})$ in the remainder of this section. In particular, we provide a partial bound on the quantity λ_0.

Lemma 28. *Define $\{\Phi_{jk\ell}, \Phi_{jk\ell'}\} : \mathcal{Z} \to \mathcal{Z}^*$ as in* (2.58)*,* (2.59)*, and define $r : \mathcal{Z} \to \mathbb{R}$ as in* (2.53)*. Letting $w_+(jk\ell\ell')$ be w_{s+1} in Algorithm 9 if j, k, ℓ, ℓ' were sampled in iteration s, defining*

$$\Phi^{fg}_{jk\ell}(w) := \Phi^{\text{sep}}_{jk\ell}(w) + \Phi^{\text{bilin}}_{jk\ell}(w),$$
$$\Phi^{fg}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) := \Phi^{\text{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) + \Phi^{\text{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)),$$

we have

$$\mathbb{E}\left[\left\langle \Phi^{fg}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^{fg}_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle\right] \le \lambda^{fg}\mathbb{E}\left[V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right],$$

Algorithm 9: MINIMAX-FINITESUM-INNER($F_{\text{mmfs-reg}}, \bar{z}^{\mathsf{x}}, \bar{z}^{\mathsf{y}}, \{\bar{z}^{\mathsf{f_i}}\}_{i\in[n]}, \{\bar{z}^{\mathsf{g_i}}\}_{i\in[n]}$): Minimax finite sum optimization subroutine

Input: (2.50) satisfying Assumption 3, $\bar{z}^{\mathsf{x}}, \{\bar{z}^{\mathsf{f_i}}\}_{i\in[n]} \in \mathcal{X}$, $\bar{z}^{\mathsf{y}}, \{\bar{z}^{\mathsf{g_i}}\}_{i\in[n]} \in \mathcal{Y}$

Parameter(s): $\gamma \geq 1$, $\lambda > 0$, $N, S \in \mathbb{N}$

$w_0 \leftarrow \bar{z}$

for $0 \leq \tau < N$ **do**

  Sample $0 \leq \sigma < S$ uniformly at random

  for $0 \leq s \leq \sigma$ **do**

    Sample $j, k, \ell, \ell' \in [n]$ independently according to p, q, r, r respectively defined in (2.57), and define

$$[\Phi^{\text{sep}}]^{\mathsf{x}} := (1+\gamma)\,\mu^{\mathsf{x}} w_s^{\mathsf{x}} - \gamma\mu^{\mathsf{x}}\bar{z}^{\mathsf{x}}, \quad [\Phi^{\text{sep}}]^{\mathsf{y}} = (1+\gamma)\,\mu^{\mathsf{y}} w_s^{\mathsf{y}} - \gamma\mu^{\mathsf{y}}\bar{z}^{\mathsf{y}},$$

$$\left[\Phi^{\text{bilin}}\right]^{\mathsf{x}} := \frac{\sum_{i\in[n]} \nabla f_i\left(w_s^{\mathsf{f_i}}\right)}{n}, \quad \left[\Phi^{\text{bilin}}\right]^{\mathsf{y}} := \frac{\sum_{i\in[n]} \nabla g_i\left(w_s^{\mathsf{g_i}}\right)}{n},$$

$$\Phi^{\mathsf{x}} := \left[\Phi^{h}(w_0)\right]^{\mathsf{x}} + \frac{\nabla_x h_\ell(w_s^{\mathsf{x}}, w_s^{\mathsf{y}}) - \nabla_x h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})}{n r_\ell} + [\Phi^{\text{sep}}]^{\mathsf{x}} + \left[\Phi^{\text{bilin}}\right]^{\mathsf{x}},$$

$$\Phi^{\mathsf{y}} := \left[\Phi^{h}(w_0)\right]^{\mathsf{y}} - \frac{\nabla_y h_\ell(w_s^{\mathsf{x}}, w_s^{\mathsf{y}}) - \nabla_y h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})}{n r_\ell} + [\Phi^{\text{sep}}]^{\mathsf{y}} + \left[\Phi^{\text{bilin}}\right]^{\mathsf{y}}$$

    $w_{s+1/2}^{\mathsf{x}} \leftarrow w_s^{\mathsf{x}} - \frac{1}{\lambda\mu^{\mathsf{x}}}\Phi^{\mathsf{x}}$, $w_{s+1/2}^{\mathsf{y}} \leftarrow w_s^{\mathsf{y}} - \frac{1}{\lambda\mu^{\mathsf{y}}}\Phi^{\mathsf{y}}$

    $w_{s+1/2}^{\mathsf{f_j}} \leftarrow w_s^{\mathsf{f_j}} - \frac{1}{n\lambda p_j}\left((1+\gamma)\,w_s^{\mathsf{f_j}} - \gamma\bar{z}^{\mathsf{f_j}} - w_s^{\mathsf{x}}\right)$

    $w_{s+1/2}^{\mathsf{g_k}} \leftarrow w_s^{\mathsf{f_k}} - \frac{1}{n\lambda p_k}\left((1+\gamma)\,w_s^{\mathsf{g_k}} - \gamma\bar{z}^{\mathsf{g_k}} - w_s^{\mathsf{y}}\right)$

    Define

$$[\Phi^{\text{sep}}]^{\mathsf{x}} := (1+\gamma)\,\mu^{\mathsf{x}} w_{s+1/2}^{\mathsf{x}} - \gamma\mu^{\mathsf{x}}\bar{z}^{\mathsf{x}}, \quad [\Phi^{\text{sep}}]^{\mathsf{y}} := (1+\gamma)\,\mu^{\mathsf{y}} w_{s+1/2}^{\mathsf{y}} - \gamma\mu^{\mathsf{y}}\bar{z}^{\mathsf{y}},$$

$$\left[\Phi^{\text{bilin}}\right]^{\mathsf{x}} := \frac{\sum_{i\in[n]} \nabla f_i\left(w_s^{\mathsf{f_i}}\right)}{n} + \frac{\nabla f_j\left(w_{s+1/2}^{\mathsf{f_j}}\right) - \nabla f_j\left(w_s^{\mathsf{f_j}}\right)}{n p_j},$$

$$\left[\Phi^{\text{bilin}}\right]^{\mathsf{y}} := \frac{\sum_{i\in[n]} \nabla g_i\left(w_s^{\mathsf{g_i}}\right)}{n} + \frac{\nabla g_k\left(w_{s+1/2}^{\mathsf{g_k}}\right) - \nabla g_k\left(w_s^{\mathsf{g_k}}\right)}{n q_k},$$

$$\Phi^{\mathsf{x}} := \left[\Phi^{h}(w_0)\right]^{\mathsf{x}} + \frac{\nabla_x h_{\ell'}(w_{s+1/2}^{\mathsf{x}}, w_{s+1/2}^{\mathsf{y}}) - \nabla_x h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})}{n r_{\ell'}} + [\Phi^{\text{sep}}]^{\mathsf{x}} + \left[\Phi^{\text{bilin}}\right]^{\mathsf{x}},$$

$$\Phi^{\mathsf{y}} := \left[\Phi^{h}(w_0)\right]^{\mathsf{y}} - \frac{\nabla_y h_{\ell'}(w_{s+1/2}^{\mathsf{x}}, w_{s+1/2}^{\mathsf{y}}) - \nabla_y h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})}{n r_{\ell'}} + [\Phi^{\text{sep}}]^{\mathsf{y}} + \left[\Phi^{\text{bilin}}\right]^{\mathsf{y}}$$

    $w_{s+1}^{\mathsf{x}} \leftarrow w_s^{\mathsf{x}} - \frac{1}{\lambda\mu^{\mathsf{x}}}\Phi^{\mathsf{x}}$, $w_{s+1}^{\mathsf{y}} \leftarrow w_s^{\mathsf{y}} - \frac{1}{\lambda\mu^{\mathsf{y}}}\Phi^{\mathsf{y}}$

    $w_{s+1}^{\mathsf{f_j}} \leftarrow w_s^{\mathsf{f_j}} - \frac{1}{n\lambda p_j}\left((1+\gamma)\,w_{s+1/2}^{\mathsf{f_j}} - \gamma\bar{z}^{\mathsf{f_j}} - w_{s+1/2}^{\mathsf{x}}\right)$

    $w_{s+1}^{\mathsf{g_k}} \leftarrow w_s^{\mathsf{f_k}} - \frac{1}{n\lambda p_k}\left((1+\gamma)\,w_{s+1/2}^{\mathsf{g_k}} - \gamma\bar{z}^{\mathsf{g_k}} - w_{s+1/2}^{\mathsf{y}}\right)$

  $\bar{w}^{\mathsf{f_i}} \leftarrow w_\sigma^{\mathsf{f_i}} - \frac{1}{n\lambda p_i}\left((1+\gamma)\,w_\sigma^{\mathsf{f_i}} - \gamma\bar{z}^{\mathsf{f_i}} - w_\sigma^{\mathsf{x}}\right)$ for each $i \in [n]$

  $\bar{w}^{\mathsf{g_i}} \leftarrow w_\sigma^{\mathsf{g_i}} - \frac{1}{n\lambda q_i}\left((1+\gamma)\,w_\sigma^{\mathsf{g_i}} - \gamma\bar{z}^{\mathsf{g_i}} - w_\sigma^{\mathsf{y}}\right)$ for each $i \in [n]$

  $w_0^{\mathsf{xy}} \leftarrow w_{\sigma+1/2}^{\mathsf{xy}}$, $w_0^{\mathsf{f_i}} \leftarrow \bar{w}^{\mathsf{f_i}}$, $w_0^{\mathsf{g_i}} \leftarrow \bar{w}^{\mathsf{g_i}}$ for all $i \in [n]$

Return: $(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}, \{\nabla f_i(w_0^{\mathsf{f_i}})\}_{i\in[n]}, \{\nabla g_i(w_0^{\mathsf{g_i}})\}_{i\in[n]})$

for

$$\lambda^{fg} = 2n(1+\gamma) + \frac{\sum_{i\in[n]}\sqrt{L_i^{\mathsf{x}}}}{\sqrt{n\mu^{\mathsf{x}}}} + \frac{\sum_{i\in[n]}\sqrt{L_i^{\mathsf{y}}}}{\sqrt{n\mu^{\mathsf{y}}}}.$$

Partial variance bound. Finally, we provide bounds on the components of (2.56) corresponding to Φ^h. Namely, we bound the quantity λ_1, and complete the bound on λ_0 within Proposition 5.

Lemma 29. *Following notation of Lemma 28, and recalling the definition* (2.61)*, for*

$$\lambda_1 := 32(\lambda^h)^2,$$

where we define

$$\lambda^h := \frac{1}{n}\sum_{i\in[n]}\left(\frac{\Lambda_i^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda_i^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda_i^{\mathsf{yy}}}{\mu^{\mathsf{y}}}\right). \tag{2.61}$$

we have for any $\rho > 0$,

$$\begin{aligned}&\mathbb{E}\left[\left\langle \Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^h_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle\right]\\ &\quad\le \left(2\lambda^h + \frac{1}{\rho}\right)\mathbb{E}\left[V_w^r\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right] + \rho\lambda_1\mathbb{E}\left[V^r_{w_0}(w^\star) + V^r_{\widetilde{w}(\ell)}(w_\star)\right].\end{aligned} \tag{2.62}$$

Combining the properties we prove above with Proposition 5, we obtain the following convergence guarantee for each loop $0 \le \tau < N$ of Lines 6 to 18 in Algorithm 9.

Proposition 6. *Consider a run of Lines 6 to 18 in Algorithm 9 initialized at $w_0 \in \mathcal{Z}$, with*

$$\lambda \leftarrow \left(2n(1+\gamma) + \frac{2\sum_{i\in[n]}\sqrt{L_i^{\mathsf{x}}}}{\sqrt{n\mu^{\mathsf{x}}}} + \frac{2\sum_{i\in[n]}\sqrt{L_i^{\mathsf{y}}}}{\sqrt{n\mu^{\mathsf{y}}}} + 2\lambda^h\right) + \frac{160(\lambda^h)^2}{\gamma},\ S \leftarrow \frac{5\lambda}{\gamma}, \tag{2.63}$$

where λ^h is defined in (2.61)*. Letting $\widetilde{w}$ be the new setting of w_0 in Line 18 at the end of the run,*

$$\mathbb{E}\left[V^r_{\widetilde{w}}(w^\star)\right] \le \frac{1}{2}V^r_{w_0}(w^\star),$$

where $w^\star$ solves the VI in Φ (defined in (2.55)*).*

2.9.4 Outer loop convergence analysis

We state the following convergence guarantee on our outer loop, Algorithm 8. The analysis is a somewhat technical modification of the standard proximal point analysis for solving VIs [415], to handle approximation error. As a result, we state the claim here and defer a proof to Appendix A.10.3.

Proposition 7. *Consider a single iteration $0 \le t < T$ of Algorithm 8, and let $z_\star$ is the saddle point to $F_{\text{mmfs-pd}}$ (defined in (2.52)). Setting S as in (2.63) and*

$$N := O\left(\log\left(\gamma\lambda\right)\right), \tag{2.64}$$

for an appropriately large constant in our implementation of Algorithm 9 and λ as in (2.63), we have

$$\mathbb{E}V^r_{z_{t+1}}(z_\star) \le \frac{4\gamma}{1+4\gamma}V^r_{z_t}(z_\star).$$

2.9.5 Main result

We now state and prove our main claim.

Theorem 9. *Suppose F_{mmfs} in (2.50) satisfies Assumption 3, and has saddle point $(x_\star, y_\star)$. Further, suppose we have $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ such that $\text{Gap}_{F_{\text{mmfs-reg}}}(x_0, y_0) \le \epsilon_0$. Algorithm 8 using Algorithm 9 with λ as in (2.63) returns $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathbb{E}\text{Gap}_{F_{\text{mmfs-reg}}}(x, y) \le \epsilon$ in N_{tot} iterations, using a total of $O(N_{\text{tot}})$ gradient calls each to some f_i, g_i, or h_i for $i \in [n]$, where*

$$\begin{aligned} N_{\text{tot}} &= O\left(\kappa_{\text{mmfs}}\log\left(\kappa_{\text{mmfs}}\right)\log\left(\frac{\kappa_{\text{mmfs}}\epsilon_0}{\epsilon}\right)\right), \\ \text{for } \kappa_{\text{mmfs}} &:= n + \frac{1}{\sqrt{n}}\sum_{i\in[n]}\left(\sqrt{\frac{L_i^{\text{x}}}{\mu^{\text{x}}}} + \sqrt{\frac{L_i^{\text{y}}}{\mu^{\text{y}}}} + \frac{\Lambda_i^{\text{xx}}}{\mu^{\text{x}}} + \frac{\Lambda_i^{\text{xy}}}{\sqrt{\mu^{\text{x}}\mu^{\text{y}}}} + \frac{\Lambda_i^{\text{yy}}}{\mu^{\text{y}}}\right). \end{aligned} \tag{2.65}$$

In particular, we use $N_{\text{tot}} = NTS$ for

$$T = O\left(\gamma\log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right),\ N = O\left(\log\left(\kappa_{\text{mmfs}}\right)\right),\ S = O\left(n + \frac{\kappa_{\text{mmfs}}}{\gamma} + \frac{(\lambda^h)^2}{\gamma^2}\right),\ \gamma = \frac{\lambda^h}{\sqrt{n}}.$$

Proof. By Lemma 24, the point $(x_\star, y_\star)$ is consistent between (2.50) and (2.52). The complexity of each iteration follows from observation of Algorithm 8 and 9.

Next, by Proposition 6 and Proposition 7, and our choices of T, N, and S for appropriately large constants, we obtain a point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that

$$\mathbb{E}V^r_{(x,y)}(z_\star) \le \frac{\epsilon}{4}\left(\frac{1}{\kappa_{\text{mmfs}}}\right)^2.$$

Here we used an analogous argument to Lemma 15 to bound the initial divergence. We then use a similar bound as in Lemma 16 to obtain the desired duality gap bound. □

We now revisit the problem (2.49). We apply a generic reduction framework we develop in Appendix A.8 to develop a solver for this problem under a relaxed version of Assumption 3, without requiring strong convexity of individual summands.

Assumption 4. *We assume the following about* (2.49) *for all* $i \in [n]$.

1. f_i *is* L_i^{x}*-smooth, and* g_i *is* L_i^{y}*-smooth.*

2. h *has the following blockwise-smoothness properties: for all* $u, v \in \mathcal{X} \times \mathcal{Y}$,

$$\begin{aligned} \|\nabla_x h_i(u) - \nabla_x h_i(v)\| &\leq \Lambda_i^{\mathsf{xx}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda_i^{\mathsf{xy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\|, \\ \|\nabla_y h_i(u) - \nabla_y h_i(v)\| &\leq \Lambda_i^{\mathsf{xy}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda_i^{\mathsf{yy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\|. \end{aligned} \tag{2.66}$$

For minimax finite sum optimization problems with this set of relaxed conditions, we conclude with the following corollary of Theorem 9.

Corollary 5. *Suppose the summands* $\{f_i, g_i, h_i\}_{i \in [n]}$ *in* (2.49) *satisfy Assumption 4, and* F_{mmfs} *is* μ^{x}*-strongly convex in* x, μ^{y}*-strongly convex in* y, *with saddle point* $(x_\star, y_\star)$. *Further, suppose we have* $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ *such that* $\text{Gap}_{F_{\text{mmfs}}}(x_0, y_0) \leq \epsilon_0$. *Algorithm 66 using Algorithm 8 and 9 to implement steps returns* $(x, y) \in \mathcal{X} \times \mathcal{Y}$ *with* $\mathbb{E}\text{Gap}(x, y) \leq \epsilon$ *in* N_{tot} *iterations, using a total of* $O(N_{\text{tot}})$ *gradient calls each to some* f_i, g_i, *or* h_i *for* $i \in [n]$, *where*

$$N_{\text{tot}} = O\left(\kappa_{\text{mmfs}} \log(\kappa_{\text{mmfs}}) \log\left(\frac{\kappa_{\text{mmfs}} \epsilon_0}{\epsilon}\right)\right),$$

$$\text{for } \kappa_{\text{mmfs}} := n + \frac{1}{\sqrt{n}} \sum_{i \in [n]} \left(\sqrt{\frac{L_i^{\mathsf{x}}}{\mu^{\mathsf{x}}}} + \sqrt{\frac{L_i^{\mathsf{y}}}{\mu^{\mathsf{y}}}} + \frac{\Lambda_i^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda_i^{\mathsf{xy}}}{\sqrt{\mu^{\mathsf{x}} \mu^{\mathsf{y}}}} + \frac{\Lambda_i^{\mathsf{yy}}}{\mu^{\mathsf{y}}} \right).$$

Chapter 3

Stochastic Methods for Matrix Games

This chapter is based on [111, 112], with Yair Carmon, Yujia Jin, and Aaron Sidford.

3.1 Introduction

Bilinear minimax problems of the form

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} y^\top A x \text{ where } A \in \mathbb{R}^{m\times n}, \tag{3.1}$$

are fundamental to machine learning, economics and theoretical computer science [393, 525, 157]. We focus on three important settings characterized by different domain geometries. When $\mathcal{X}$ and $\mathcal{Y}$ are probability simplices—which we call the ℓ_1-ℓ_1 setting—the problem (3.1) corresponds to a zero-sum matrix game and also to a linear program in canonical feasibility form. The ℓ_2-ℓ_1 setting, where $\mathcal{X}$ is a Euclidean ball and $\mathcal{Y}$ is a simplex, is useful for linear classification (hard-margin support vector machines) as well as problems in computational geometry [140]. Further, the ℓ_2-ℓ_2 setting, where both $\mathcal{X}$ and $\mathcal{Y}$ are Euclidean balls (with general center), includes linear regression.

Many problems of practical interest are *sparse*, i.e., the number of nonzero elements in A, which we denote by nnz, satisfies $\mathsf{nnz} \ll mn$. Examples include: linear programs with constraints involving few variables, linear classification with 1-hot-encoded features, and linear systems that arise from physical models with local interactions. The problem description size nnz plays a central role in several runtime analyses of algorithms for solving the problem (3.1).

However, sparsity is not an entirely satisfactory measure of instance complexity: it is not continuous in the elements of A and consequently it cannot accurately reflect the simplicity of "nearly

sparse" instances with many small (but nonzero) elements. Measures of *numerical sparsity*, such as the ℓ_1 to ℓ_2 norm ratio, can fill this gap [260]. Indeed, many problems encountered in practice are numerically sparse. Examples include: linear programming constraints of the form $x_1 \geq \frac{1}{n}\sum_i x_i$, linear classification with neural network activations as features, and linear systems arising from physical models with interaction whose strength decays with distance.

Existing bilinear minimax solvers do not exploit the numerical sparsity of A and their runtime guarantees do not depend on it—the basic limitation of these methods is that they do not directly access the large matrix entries, and instead sample the full columns and rows in which they occur. To overcome this limitation, we propose methods that access A a single entry at a time, leverage numerical sparsity by accessing larger coordinates more frequently, and enjoy runtime guarantees that depend explicitly on numerical sparsity measures. For numerically sparse large-scale instances our runtimes are substantially better than the previous state-of-the-art. Moreover, our runtimes subsume the previous state-of-the-art dependence on nnz and rcs, the maximum number of nonzeros in any row or column.

In addition to proposing algorithms with improved runtimes, we develop three techniques that may be of broader interest. First, we design non-uniform sampling schemes that minimize regret bounds; we use a general framework that unifies the Euclidean and (local norms) simplex geometries, possibly facilitating future extension. Next, we propose novel stochastic variance reduced methods for saddle point problems, which are capable of interpolating between the runtimes achieved by accelerated gradient methods and those achieved by sublinear (fully-)stochastic gradient methods using a proximal optimization framework of [415]. Lastly, we build a data structure capable of efficiently maintaining and sampling from multiplicative weights iterations (i.e. entropic projection) *with a fixed dense component*. This data structure overcomes limitations of existing techniques for maintaining entropic projections and we believe it may prove effective in other settings where such projections appear.

3.1.1 Our results

Table 3.2 summarizes our runtime guarantees and puts them in the context of the best existing results. We consider methods that output (expected) ϵ-accurate solutions of the saddle-point problem (3.1), namely a pair x, y satisfying

$$\mathbb{E}\left[\max_{v\in\mathcal{Y}} v^\top Ax - \min_{u\in\mathcal{X}} y^\top Au\right] \leq \epsilon.$$

The algorithms in Table 3.2 are all iterative solvers for the general problem $\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x,y)$, specialized to $f(x,y) = y^\top Ax$. Each algorithm presents a different tradeoff between per-iteration complexity and the required iteration count, corresponding to the matrix access modality: exact gradient methods compute matrix-vector products in each iteration, row-column stochastic gradient

methods sample a row and a column in each iteration, and our proposed coordinate methods take this tradeoff to an extreme by sampling a single coordinate of the matrix per iteration.[1] In addition, variance reduction (VR) schemes combine both fast stochastic gradient computations and infrequent exact gradient computations, maintaining the amortized per-iteration cost of the stochastic scheme and reducing the total iteration count for sufficiently small ϵ.

The runtimes in Table 3.2 depend on the numerical range of A through a matrix norm L that changes with both the problem geometry and the type of matrix access; we use L_{mv}, L_{rc} and L_{co} to denote the constants corresponding to matrix-vector products, row-column queries and coordinated queries, respectively. Below, we describe these runtimes in detail. In the settings we study, our results are the first theoretical demonstration of runtime gains arising from sampling a single coordinate of A at a time, as opposed to entire rows and columns.

Variance reduction for matrix games. In recent years, minimax formulations for neural network training rose to prominence [244, 378], leading to intense interest in algorithms for solving large scale minimax games [198, 237, 323, 161, 297, 391]. However, the algorithmic toolbox for minimax optimization is not as complete as the one for minimization. Variance reduction, a technique for improving stochastic gradient estimators by introducing control variates, stands as a case in point. A multitude of variance reduction schemes exist for finite-sum minimization [300, 472, 18, 92, 217], and their impact on complexity is well-understood [539]. In contrast, only a few works apply variance reduction to finite-sum minimax problems [55, 484, 121, 394], and the potential gains from variance reduction are not well-understood.

We take a step towards closing this gap by designing variance-reduced minimax game solvers that offer strict runtime improvements over non-stochastic gradient methods, similar to that of optimal variance reduction methods for finite-sum minimization. Our starting point is Nemirovski's "conceptual prox-method" [415] for solving $\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x,y)$, where $f:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}$ is convex in x and concave in y. The method solves a sequence of subproblems parameterized by $\alpha>0$, each of the form

$$\text{find } x,y \text{ s.t. } \forall x',y' \;\; \langle\nabla_x f(x,y), x-x'\rangle - \langle\nabla_y f(x,y), y-y'\rangle \le \alpha V_{x_0}(x') + \alpha V_{y_0}(y') \tag{3.2}$$

for some $(x_0,y_0)\in\mathcal{X}\times\mathcal{Y}$, where $V_a(b)$ is a norm-suitable Bregman divergence from a to b: squared Euclidean distance for ℓ_2 and KL divergence for ℓ_1. Combining each subproblem solution with an extragradient step, the prox-method solves the original problem to ϵ accuracy by solving $\widetilde{O}\left(\alpha/\epsilon\right)$

[1] Interior point methods offer an alternative tradeoff between iteration cost and iteration count: the number of required iterations depends on $1/\epsilon$ only logarithmically, but every iteration is costly, requiring a linear system solution which at present takes time $\Omega(\min\{m,n\}^2)$. In the ℓ_1-ℓ_1 geometry, the best known runtimes for interior point methods are $\widetilde{O}((\mathsf{nnz}+\min\{m,n\}^2)\sqrt{\min\{m,n\}})$ [349], $\widetilde{O}(\max\{m,n\}^\omega)$ [144], and $\widetilde{O}(mn+\min\{m,n\}^3)$ [517]. In this chapter we are mainly interested in the large-scale low-accuracy regime with $L/\epsilon<\min(m,n)$ where the runtimes described in Table 3.2 are favorable (with the exception of [517] in certain cases). Our methods take only few passes over the data, which are not the case for many interior-point methods [349, 144]. Also, our methods do not rely on a general (ill-conditioned) linear system solver, which is a key ingredient in interior point methods.

	L_{mv} (matrix-vector)	L_{rc} (row-column)	L_{co} (coordinate)
ℓ_1-ℓ_1	$\max_{i,j} \lvert A_{ij} \rvert$	$\max_{i,j} \lvert A_{ij} \rvert$	$\max\left\{ \max_i \lVert A_{i:} \rVert_2 , \max_j \lVert A_{:j} \rVert_2 \right\}$
ℓ_2-ℓ_1	$\max_i \lVert A_{i:} \rVert_2$	$\max_i \lVert A_{i:} \rVert_2$	$\max\left\{ \max_i \lVert A_{i:} \rVert_1 , \lVert A \rVert_{\mathrm{F}} \right\}^{\dagger}$
ℓ_2-ℓ_2	$\lVert A \rVert_{\mathrm{op}}$	$\lVert A \rVert_{\mathrm{F}}$	$\max\left\{ \sqrt{\sum_i \lVert A_{i:} \rVert_1^2}, \sqrt{\sum_j \lVert A_{:j} \rVert_1^2} \right\}$

Table 3.1: Dependence on A for different methods in different geometries. Comments: $A_{i:}$ and $A_{:j}$ denote the ith row and jth column of A, respectively. Numerically sparse instances satisfy $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$. † In the ℓ_2-ℓ_1 setting we can also achieve $L_{\mathsf{co}} = L_{\mathsf{rc}}\sqrt{\mathsf{rcs}}$ and $L_{\mathsf{co}} = \max\{\max_i \lVert A_{i:} \rVert_1 , \sqrt{\max_i \lVert A_{i:} \rVert_1 \max_j \lVert A_{:j} \rVert_1}\}$.

Method	Iteration cost	Total runtime
Exact gradient [415, 420]	$O(\mathsf{nnz})$	$\widetilde{O}\left(\mathsf{nnz} \cdot L_{\mathsf{mv}} \cdot \epsilon^{-1}\right)$
Row-column [253, 140, 55]	$O(n+m)$	$\widetilde{O}\left((m+n) \cdot L_{\mathsf{rc}}^2 \cdot \epsilon^{-2}\right)$
Row-column VR [55, 111]	$O(n+m)$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot (m+n)} \cdot L_{\mathsf{rc}} \cdot \epsilon^{-1}\right)$
Sparse row-col (folklore)	$\widetilde{O}\left(\mathsf{rcs}\right)$	$\widetilde{O}\left(\mathsf{rcs} \cdot L_{\mathsf{rc}}^2 \cdot \epsilon^{-2}\right)$
Sparse row-col VR (Appendix B.5)	$\widetilde{O}\left(\mathsf{rcs}\right)$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot L_{\mathsf{rc}} \cdot \epsilon^{-1}\right)$
Coordinate (Section 3.3.1)	$\widetilde{O}\left(1\right)$	$\widetilde{O}\left(\mathsf{nnz} + L_{\mathsf{co}}^2 \cdot \epsilon^{-2}\right)$
Coordinate VR (Section 3.3.2)	$\widetilde{O}\left(1\right)$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}} \cdot \epsilon^{-1}\right)$

Table 3.2: Comparison of iterative methods for bilinear problems. Comments: nnz denotes the number of nonzeros in $A \in \mathbb{R}^{m \times n}$ and $\mathsf{rcs} \leq \max\{m, n\}$ denotes the maximum number of nonzeros in any row and column of A. The quantities $L_{\mathsf{mv}}, L_{\mathsf{co}}$ and L_{rc} depend on problem geometry (see Table 3.1).

Task	Method	Runtime
MaxIB	[22]	$\widetilde{O}\left(mn + \rho m\sqrt{n} \cdot \epsilon^{-1}\right)$
	Our method (Theorem 12)	$\widetilde{O}\left(\mathsf{nnz} + \rho\sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot \epsilon^{-1}\right)^{\dagger}$
MinEB (when $m \geq n$)	[22]	$\widetilde{O}\left(mn + m\sqrt{n} \cdot \epsilon^{-1/2}\right)$
	Our method (Theorem 13)	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot \epsilon^{-1/2}\right)^{\dagger}$
Regression ($A^\top A \succeq \mu I$)	AGD [417]	$\widetilde{O}\left(\mathsf{nnz} \cdot \lVert A \rVert_{\mathrm{op}} \frac{1}{\sqrt{\mu}}\right)$
	[260]	$\widetilde{O}\left(\mathsf{nnz} + \mathsf{nnz}^{2/3} \cdot \left(\sum_{i \in [n]} \lVert A \rVert_{\mathrm{F}} \cdot \lVert A_{i:} \rVert_1 \cdot \lVert A_{i:} \rVert_2\right)^{1/3} \frac{1}{\sqrt{\mu}}\right)$
	Our method (Theorem 14)	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \max\left\{\sqrt{\sum_i \lVert A_{i:} \rVert_1^2}, \sqrt{\sum_j \lVert A_{:j} \rVert_1^2}\right\} \frac{1}{\sqrt{\mu}}\right)$

Table 3.3: Comparison of complexity for different applications. Comments: ρ denotes the radii ratio of the minimum ball enclosing the rows of A and maximum ball inscribed in them. † For MaxIB and MinEB, we refer the reader to Section 3.6.2 for a more fine-grained runtime bound.

subproblems.[2] (Solving (3.2) with $\alpha = 0$ is equivalent to to solving $\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x,y)$.) A basic form of our first contribution is showing that if a stochastic unbiased gradient estimator $\tilde{g}$ satisfies the "variance" bound

$$\mathbb{E}\left\|\tilde{g}(x,y) - \nabla f(x_0,y_0)\right\|_*^2 \le L^2\left\|x - x_0\right\|^2 + L^2\left\|y - y_0\right\|^2 \tag{3.3}$$

for some $L > 0$, then $O(L^2/\alpha^2)$ regularized stochastic mirror descent steps using $\tilde{g}$ solve (3.2) in a suitable probabilistic sense. We call unbiased gradient estimators that satisfy (3.3) "centered." We will shortly discuss the design of such centered gradient estimators in various application, which constitutes another contribution of this chapter.

As per the discussion above, to achieve accuracy ϵ our algorithm solves $\widetilde{O}(\alpha/\epsilon)$ subproblems. Each subproblem takes $O(\mathsf{nnz}(A))$ time for computing two exact gradients (one for variance reduction and one for an extragradient step), plus an additional $(m+n)L^2/\alpha^2$ time for the inner mirror descent iterations, with L an appropriate Lipschitz constant of the problem. The total runtime is therefore

$$\widetilde{O}\left(\left(\mathsf{nnz}(A) + \frac{(m+n)L^2}{\alpha^2}\right)\frac{\alpha}{\epsilon}\right).$$

By setting α optimally to be $\max\{\epsilon, L\sqrt{(m+n)/\mathsf{nnz}(A)}\}$, we obtain the runtime

$$\widetilde{O}\left(\mathsf{nnz}(A) + \sqrt{\mathsf{nnz}(A)\cdot(m+n)}\cdot L\cdot\epsilon^{-1}\right). \tag{3.4}$$

Some problem instances admit a structured form of sparsity where every row and column has at most rcs nonzero elements. In these settings, we develop sampling distributions and data structures to allow the above framework to leverage row-column sparsity, reducing the amortized per-iteration cost from $O(m+n)$ to $\widetilde{O}(\mathsf{rcs})$, and yielding improved tradeoffs of problem parameters.

Coordinate stochastic gradient methods. In conjunction with our novel variance reduction framework, we develop coordinate stochastic gradient estimators which allow per-iteration cost $\widetilde{O}(1)$ and iteration count $\widetilde{O}\left(n+m+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$. We define L_{co} in Table 3.1; for each domain geometry, the quantity $\frac{L_{\mathsf{co}}}{L_{\mathsf{rc}}}$ is a measure of the numerical sparsity of A, satisfying

$$1 \le \frac{L_{\mathsf{co}}^2}{L_{\mathsf{rc}}^2} \le \mathsf{rcs}.$$

Every iteration of our method requires sampling an element in a row or a column with probability proportional to its entries. Assuming a matrix access model that allows such sampling in time $\widetilde{O}(1)$

[2] More precisely, the required number of subproblem solutions is at most $\Theta\cdot\frac{\alpha}{\epsilon}$, where Θ is a "domain size" parameter that depends on $\mathcal{X}$, $\mathcal{Y}$, and the Bregman divergence V. In the ℓ_1 and ℓ_2 settings considered in this chapter, we have the bound $\Theta \le \log(nm)$ and we use the $\widetilde{O}$ notation to suppress terms logarithmic in n and m. However, in other settings—e.g., ℓ_∞-ℓ_1 games [483, 486]—making the parameter Θ scale logarithmically with the problem dimension is far more difficult.

(similarly to [47, 487, 238]), the total runtime of our method is $\widetilde{O}\left(n+m+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$. In this case, for numerically sparse problems such that $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$, the proposed coordinate methods outperform row-column sampling by a factor of $m+n$. Moreover, the bound $L_{\mathsf{co}}^2 \leq L_{\mathsf{rc}}^2(m+n)$ implies that our runtime is never worse than that of row-column methods. When only coordinate access to the matrix A is initially available, we may implement the required sampling access via preprocessing in time $O(\mathsf{nnz})$. This changes the runtime to $\widetilde{O}\left(\mathsf{nnz}+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$, so that the comparison above holds only when $(\frac{L_{\mathsf{co}}}{\epsilon})^2 = \tilde{\Omega}(\mathsf{nnz})$. In that regime, the variance reduction technique we describe below provides even stronger guarantees.

Coordinate methods with variance reduction. By combining the two aforementioned contributions, we design a variance reduction algorithm with amortized per-iteration cost $\widetilde{O}(1)$, required iteration count of $\widetilde{O}\left(\sqrt{\mathsf{nnz}}\cdot\frac{L_{\mathsf{co}}}{\epsilon}\right)$ and total running time $\widetilde{O}\left(\mathsf{nnz}+\sqrt{\mathsf{nnz}}\cdot\frac{L_{\mathsf{co}}}{\epsilon}\right)$. In the numerically sparse regime $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$, our runtime improves on row-column VR by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$, and in general the bound $L_{\mathsf{co}} \leq L_{\mathsf{rc}}\sqrt{m+n}$ guarantees it is never worse. Since variance reduction methods always require a single pass over the data to compute an exact gradient, this comparison holds regardless of the matrix access model. In the ℓ_2-ℓ_2 setting we note that for *elementwise non-negative* matrices, $L_{\mathsf{co}} = \max\{\|A\mathbf{1}\|_2, \|A^\top\mathbf{1}\|_2\} \leq L_{\mathsf{mv}}\sqrt{m+n}$, and consequently our method outperforms exact gradient methods by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$, even without any numerical or spectral sparsity in A. Notably, this is the same factor of improvement that row-column VR achieves over exact gradient methods in the ℓ_1-ℓ_1 and ℓ_2-ℓ_1 regimes.

Optimality of the constant L_{co}. For the ℓ_1-ℓ_1 and ℓ_2-ℓ_2 settings, we argue that the constant L_{co} in Table 3.1 is optimal in the restricted sense that no alternative sampling distribution for coordinate gradient estimation can have a better variance bound than L_{co} (a similar sense of optimality also holds for L_{rc} in each geometry). In the ℓ_2-ℓ_1 setting, a different sampling distribution produces an improved (and optimal) constant $\max\{\max_i \|A_{i:}\|_1, \||A|\|_{\mathrm{op}}\}$, where $A_{i:}$ is the ith row of A, and $|A|_{ij} = |A_{ij}|$ is the elementwise absolute value of A. However, it is unclear how to efficiently sample from this distribution.

Applications. We illustrate the implications of our results for two problems in computational geometry, minimum enclosing ball (Min-EB) and maximum inscribed ball (Max-IB), as well as linear regression. For Min-EB and Max-IB in the non-degenerate case $m \geq n$, we apply our ℓ_2-ℓ_1 results to obtain algorithms whose runtime bounds coincide with the state-of-the-art [22] for dense problems, but can be significantly better for sparse or row-column sparse instances. For linear regression we focus on accelerated linearly converging algorithms, i.e., those that find x such that $\|Ax-b\|_2 \leq \epsilon$ in time proportional to $\mu^{-\frac{1}{2}}\log\frac{1}{\epsilon}$ where μ is the smallest eigenvalue of $A^\top A$. Within this class and in a number of settings, our reduced variance coordinate method offers improvement over the state-of-the-art: for instances where $\|A_{i:}\|_1 = O(\|A_{i:}\|_2)$ and $\|A_{:j}\|_1 = O(\|A_{:j}\|_2)$ for all i,j it outperforms [260] by a factor of $\mathsf{nnz}^{1/6}$, and for elementwise nonnegative instances it outperforms accelerated gradient descent by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$. See Table 3.3 for a detailed runtime

comparison.

3.1.2 Our approach

We now provide a detailed overview of our algorithm design and analysis techniques, highlighting our main technical insights. We focus on the ℓ_1-ℓ_1 geometry, since it showcases all of our developments. Beyond our proposal of a new variance-reduced scheme for minimax optimization, our technical contributions have two central themes:

1. **Sampling schemes design.** The key to obtaining efficient coordinate methods is carefully choosing the sampling distribution. Here, local norms analysis of stochastic mirror descent [474] on the one hand enables tight regret bounds, and on the other hand imposes an additional design constraint since the stochastic estimators must be bounded for the analysis to apply. We achieve estimators with improved variance bounds meeting this boundedness constraint by leveraging a "clipping" operation introduced by [140]. Specifically, in the simplex geometry, we truncate large coordinates of our estimators, and show that our method is robust to the resulting distortion.

2. **Data structure design.** Our goal is to perform iterations in $\widetilde{O}(1)$ time, but our mirror descent procedures call for updates that change $m+n$ variables in each step. We resolve this tension via data structures that implicitly maintain the iterates. Variance reduction poses a considerable challenge here, because every reduced-variance stochastic gradient contains a dense component that changes all coordinates in a complicated way. In particular, existing data structures cannot efficiently compute the normalization factor necessary for projection to the simplex. We design a data structure that overcomes this hurdle via Taylor expansions, coordinate binning, and a binomial heap-like construction. The data structure computes approximate mirror projections, and we modify the standard mirror descent analysis to show it is stable under the particular structure of the resulting approximation errors.

At the intersection of these two themes is a novel sampling technique we call "sampling from the sum," which addresses the same variance challenges as the "sampling from the difference" technique of an earlier version of this work [111], but is more amenable to efficient implementation with a data structure.

Coordinate stochastic gradient method

Our algorithm is an instance of stochastic mirror descent [416], which in the ℓ_1-ℓ_1 setting produces a sequence of iterates $(x_1, y_1), (x_2, y_2), \ldots$ according to

$$x_{t+1} = \Pi_\Delta \left(x_t \circ \exp\{-\eta \tilde{g}^{\mathsf{x}}(x_t, y_t)\} \right) \quad \text{and} \quad y_{t+1} = \Pi_\Delta \left(y_t \circ \exp\{-\eta \tilde{g}^{\mathsf{y}}(x_t, y_t)\} \right), \tag{3.5}$$

where $\Pi_\Delta(v) = \frac{v}{\|v\|_1}$ is the projection onto the simplex (exp and log are applied to vectors elementwise, and elementwise multiplication is denoted by $\circ$), η is a step size, and $\tilde{g}^{\mathsf{x}}, \tilde{g}^{\mathsf{y}}$ are stochastic gradient estimators for $f(x,y) = y^\top Ax$ satisfying

$$\mathbb{E}\tilde{g}^{\mathsf{x}}(x,y) = \nabla_x f(x,y) = A^\top y \quad \text{and} \quad \mathbb{E}\tilde{g}^{\mathsf{y}}(x,y) = -\nabla_y f(x,y) = -Ax.$$

We describe the computation and analysis of $\tilde{g}^{\mathsf{x}}$; the treatment of $\tilde{g}^{\mathsf{y}}$ is analogous. To compute $\tilde{g}^{\mathsf{x}}(x,y)$, we sample i,j from a distribution $p(x,y)$ on $[m] \times [n]$ and let

$$\tilde{g}^{\mathsf{x}}(x,y) = \frac{y_i A_{ij}}{p_{ij}(x,y)} e_j, \tag{3.6}$$

where $p_{ij}(x,y)$ denotes the probability of drawing i,j from $p(x,y)$ and e_j is the jth standard basis vector—a simple calculation shows that $\mathbb{E}\tilde{g}^{\mathsf{x}} = A^\top y$ for any p. We first design $p(x,y)$ to guarantee an $\widetilde{O}\left((\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$ iteration complexity for finding an ϵ-accurate solution, and then briefly touch on how to compute the resulting iterations in $\widetilde{O}(1)$ time.

Local norms-informed distribution design. The standard stochastic mirror descent analysis [416] shows that if $\mathbb{E}\left\|\tilde{g}^{\mathsf{x}}(x,y)\right\|_\infty^2 \le L^2$ for all x,y (and similarly for $\tilde{g}^{\mathsf{y}}$), taking $\eta = \frac{\epsilon}{L^2}$ and a choice of $T = \widetilde{O}\left((\frac{L}{\epsilon})^2\right)$ suffices to ensure that the iterate average $\frac{1}{T}\sum_{t=1}^T (x_t, y_t)$ is an ϵ-accurate solution in expectation. Unfortunately, this analysis demonstrably fails to yield sufficiently tight bounds for our coordinate estimator: there exist instances for which any distribution p produces $L \ge nL_{\mathsf{rc}}$. We tighten the analysis using a *local norms* argument [**?**, cf.]Section 2.8]Shalev-Shwartz12, showing that $\widetilde{O}\left((\frac{L}{\epsilon})^2\right)$ iterations suffice whenever $\left\|\eta\tilde{g}^{\mathsf{x}}\right\|_\infty \le 1$ with probability 1 and for all x,y

$$\mathbb{E}\left\|\tilde{g}^{\mathsf{x}}(x,y)\right\|_x^2 \le L^2, \quad \text{where} \quad \|\gamma\|_x^2 = \sum_j x_j \gamma_j^2$$

is the local norm at $x \in \mathcal{X}$. We take

$$p_{ij} = y_i \frac{A_{ij}^2}{\|A_{i:}\|_2^2} \tag{3.7}$$

(recalling that x,y are both probability vectors). Substituting into (3.6) gives

$$\mathbb{E}\left\|\tilde{g}^{\mathsf{x}}(x,y)\right\|_x^2 = \sum_{i,j} \frac{y_i^2 A_{ij}^2 x_j}{p_{ij}} = \sum_{i,j} y_i \|A_{i:}\|_2^2 x_j = \sum_i y_i \|A_{i:}\|_2^2 \le \max_i \|A_{i:}\|_2^2 \le L_{\mathsf{co}}^2,$$

with $L_{\mathsf{co}} = \max\{\max_i \|A_{i:}\|_2, \max_j \|A_{:j}\|_2\}$ as in Table 3.1.

While this is the desired bound on $\mathbb{E}\left\|\tilde{g}^{\mathsf{x}}(x,y)\right\|_x^2$, the requirement $\left\|\eta\tilde{g}^{\mathsf{x}}\right\|_\infty \le 1$ does not hold when A has sufficiently small elements. We address this by clipping $\tilde{g}$: we replace $\eta\tilde{g}^{\mathsf{x}}$ with $\mathrm{clip}(\eta\tilde{x}^{\mathsf{x}})$, where

$$[\mathrm{clip}(v)]_i := \min\{|v_i|, 1\} \operatorname{sign}(v_i),$$

the Euclidean projection to the unit box. The clipped gradient estimator clearly satisfies the desired bounds on infinity norm and local norm second moment, but is biased for the true gradient. Following the analysis of [140], we account for the bias by relating it to the second moment via

$$|\langle \gamma - \mathrm{clip}(\gamma), x\rangle| \leq \|\gamma\|_x^2,$$

which allows to absorb the effect of the bias into existing terms in our error bounds. Putting together these pieces yields the desired bound on the iteration count.

Efficient implementation. Data structures for performing the update (3.5) and sampling from the resulting iterates in $\widetilde{O}(1)$ time are standard in the literature [**?**, e.g.,]]ShalevW16. We add to these the somewhat non-standard ability to also efficiently track the running sum of the iterates. To efficiently sample $i, j \sim p$ according to (3.7) we first use the data structure to sample $i \sim y$ in $\widetilde{O}(1)$ time and then draw $j \in [n]$ with probability proportional to A_{ij}^2 in time $O(1)$, via either $O(\mathsf{nnz})$ preprocessing or an appropriate assumption about the matrix access model. The "heavy lifting" of our data structure design is dedicated for supporting variance reduction, which we describe in the next section.

Sampling distributions beyond ℓ_1-ℓ_1. Table 3.4 lists the sampling distributions we develop for the various problem geometries. Note that for the ℓ_2-ℓ_1 setting we give three different distributions for sampling the simplex block of the gradient (i.e., $\tilde{g}^{\mathsf{y}}$); each distribution corresponds to a different parameter L_{co} (see comments following Table 3.1). The distribution $q_{ij} \propto \sqrt{y_i}\,|A_{ij}x_j|$ yields a stronger bound L in the ℓ_2-ℓ_1 setting, but we do not know how to efficiently sample from it.

Coordinate variance reduction

To accelerate the stochastic coordinate method we apply our variance reduction framework. This framework operates in $\frac{\alpha}{\epsilon}$ epochs, where α is a design parameter that trades between full and stochastic gradient computations. Each epoch consists of three parts: (i) computing the exact gradient at a reference point (x_0, y_0), (ii) performing T iterations of regularized stochastic mirror descent to produce the sequence $(x_1, y_1), \ldots, (x_T, y_T)$ and (iii) taking an extra-gradient step from the average of the iterates in (ii). Setting $\kappa = 1/(1 + \eta\alpha/2)$, the iterates x_t follow the recursion

$$x_{t+1} = \Pi_\Delta\big(x_t^\kappa \circ x_0^{1-\kappa} \circ \exp\{-\eta\kappa[g_0^{\mathsf{x}} + \tilde{\delta}^{\mathsf{x}}(x_t, y_t)]\}\big), \text{ where } \Pi_\Delta(v) = \frac{v}{\|v\|_1}, \tag{3.8}$$

and $g_0^{\mathsf{x}} = A^\top y_0$ is the exact gradient at the reference point, and $\tilde{\delta}^{\mathsf{x}}$ is a stochastic gradient *difference* estimator satisfying

$$\mathbb{E}\tilde{\delta}^{\mathsf{x}}(x, y) = \nabla_x f(x, y) - \nabla_x f(x_0, y_0) = A^\top(y - y_0).$$

The iteration for y_t is similar. As discussed earlier, we show that if $\tilde{\delta}^{\mathsf{x}}$ satisfies

$$\mathbb{E}\|\tilde{\delta}^{\mathsf{x}}(x,y)\|_\infty^2 \le L^2\left(\|x-x_0\|_1^2 + \|y-y_0\|_1^2\right) \quad \forall x, y \tag{3.9}$$

and a similar bound holds on $\mathbb{E}\|\tilde{\delta}^{\mathsf{y}}(x,y)\|_\infty^2$, then $T = O(\frac{L^2}{\alpha^2})$ iterations per epoch with step size $\eta = \frac{\alpha}{L^2}$ suffice for the overall algorithm to return a point with expected error below ϵ.

We would like to design a coordinate-based estimator $\tilde{\delta}$ such that the bound (3.9) holds for $L = L_{\mathsf{co}}$ as in Table 3.1 and each iteration (3.8) takes $\widetilde{O}(1)$ time. Since every epoch also requires $O(\mathsf{nnz})$ time for matrix-vector product (exact gradient) computations, the overall runtime would be $\widetilde{O}((\mathsf{nnz} + \frac{L_{\mathsf{co}}^2}{\alpha^2}) \cdot \frac{\alpha}{\epsilon})$. Choosing $\alpha = L_{\mathsf{co}}/\sqrt{\mathsf{nnz}}$ then gives the desired runtime $\widetilde{O}(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \frac{L_{\mathsf{co}}}{\epsilon})$.

Distribution design (sampling from the difference). We start with a straightforward adaptation of the general estimator form (3.6). To compute $\tilde{\delta}^{\mathsf{x}}(x,y)$, we sample $i, j \sim p$, where p may depend on x, x_0, y and y_0, and let

$$\tilde{\delta}^{\mathsf{x}}(x,y) = \frac{(y_i - [y_0]_i)A_{ij}}{p_{ij}} e_j, \tag{3.10}$$

where e_j is the jth standard basis vector. As in the previous section, we find that the requirement (3.9) is too stringent for coordinate-based estimators. Here too, we address this challenge with a local norms argument and clipping of the difference estimate. Using the "sampling from the difference" technique from an earlier version of this work [111], we arrive at

$$p_{ij} = \frac{|y_i - [y_0]_i|}{\|y - y_0\|_1} \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}. \tag{3.11}$$

This distribution satisfies the local norm relaxation of (3.9) with $L^2 = L_{\mathsf{co}}^2$.

Data structure design. Efficiently computing (3.8) is significantly more challenging than its counterpart (3.5). To clarify the difficulty and describe our solution, we write

$$x_t = \Pi_\Delta(\hat{x}_t) = \hat{x}_t / \|\hat{x}_t\|_1$$

and break the recursion for the unnormalized iterates $\hat{x}_t$ into two steps

$$\hat{x}_t' = \hat{x}_t^\kappa \circ \exp\{v\}, \text{ and} \tag{3.12}$$

$$\hat{x}_{t+1} = \hat{x}_t' \circ \exp\{s_t\}, \tag{3.13}$$

where $v = (1-\kappa)\log x_0 - \eta\kappa g_0^{\mathsf{x}}$ is a fixed dense vector, and $s_t = -\eta\tilde{\delta}^{\mathsf{x}}(x_t, y_t)$ is a varying 1-sparse vector. The key task of the data structures is maintaining the normalization factor $\|\hat{x}_t\|_1$ in near-constant time. Standard data structures do not suffice because they lack support for the dense step (3.12).

Our high-level strategy is to handle the two steps (3.12) and (3.13) separately. To handle the

dense step (3.12), we propose the data structure `ScaleMaintainer` that efficiently approximates $\|\hat{x}_t\|_1$ in the "homogeneous" case of no sparse updates (i.e. $s_t = 0$ for all t). We then add support for the sparse step (3.13) using a binomial heap-like construction involving $O(\log n)$ instances of `ScaleMaintainer`.

The `ScaleMaintainer` *data structure.* When $s_t = 0$ for all t the iterates $\hat{x}_t$ admit closed forms

$$\hat{x}_{t+\tau} = \hat{x}_t^{\kappa^\tau} \circ \exp\left\{ v \sum_{t'=0}^{\tau-1} \kappa^{t'} \right\} = \hat{x}_t^{\kappa^\tau} \circ \exp\left\{ \frac{1-\kappa^\tau}{1-\kappa} v \right\} = \hat{x}_t \circ \exp\left\{ [1-\kappa^\tau]\bar{v} \right\},$$

where $\bar{v} = \frac{v}{1-\kappa} - \log x_t$. Consequently, we design `ScaleMaintainer` to take as initialization $\bar{n}$-dimensional vectors $\bar{x} \in \mathbb{R}^{\bar{n}}_{\geq 0}$, and $\bar{v} \in \mathbb{R}^{\bar{n}}$ and provide approximations of the normalization factor

$$Z_\tau(\bar{x}, \bar{v}) = \|\bar{x} \circ \exp\{(1-\kappa^\tau)\bar{v}\}\|_1 \tag{3.14}$$

for arbitrary values of $\tau \geq 1$. We show how to implement each query of $Z_\tau(\bar{x}, \bar{v})$ in amortized time $\widetilde{O}(1)$. The data structure also supports initialization in time $\widetilde{O}(\bar{n})$ and deletions (i.e., setting elements of $\bar{x}$ to zero) in amortized time $\widetilde{O}(1)$.

To efficiently approximate the quantity $Z_\tau(\bar{x}, \bar{v})$ we replace the exponential with its order $p = O(\log n)$ Taylor series. That is, we would like to write

$$Z_\tau(\bar{x}, \bar{v}) = \sum_{i \in [\bar{n}]} [\bar{x}]_i e^{(1-\kappa^\tau)[\bar{v}]_i} \approx \sum_{i \in [\bar{n}]} [\bar{x}]_i \sum_{q=0}^{p} \frac{1}{q!} (1-\kappa^\tau)^q [\bar{v}]_i^q = \sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, \bar{v}^q \rangle .$$

The approximation $\sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, \bar{v}^q \rangle$ is cheap to compute, since for every τ it is a linear combination of the $p = \widetilde{O}(1)$ numbers $\{\langle \bar{x}, \bar{v}^q \rangle\}_{q \in [p]}$ which we can compute once at initialization. However, the Taylor series approximation has low multiplicative error only when $|(1-\kappa^\tau)[\bar{v}]_i| = O(p)$, which may fail to hold, as we may have $\|\bar{v}\|_\infty = \mathsf{poly}(n)$ in general. To handle this, suppose that for a fixed τ we have an offset $\mu \in \mathbb{R}$ and "active set" $A \subseteq [\bar{n}]$ such that the following conditions hold for a threshold $R = O(p)$: (a) the Taylor approximation is valid in A, e.g. we have $|(1-\kappa^\tau)(\bar{v}_i - \mu)| \leq 2R$ for all $i \in A$, (b) entries outside A are small; $(1-\kappa^\tau)[\bar{v}_i - \mu] \leq -R$ for all $i \notin A$, and (c) at least one entry in the active set is large; $(1-\kappa^\tau)[\bar{v}_i - \mu] \geq 0$ for some $i \in A$. Under these conditions, the entries in A^c are negligibly small and we can truncate them, resulting in the approximation

$$e^{(1-\kappa^\tau)\mu} \left[\sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, (\bar{v}-\mu)^q \rangle_A + e^{-R} \langle \bar{x}, \mathbf{1} \rangle_{A^c} \right],$$

which we show approximates $Z_\tau(\bar{x}, \bar{v})$ to within $e^{O(R+\log n)-\Omega(p)}$ multiplicative error, where we used $\langle a, b \rangle_S := \sum_{i \in S} a_i b_i$; here, we also require that $\log \frac{\max_i \bar{x}_i}{\min_i \bar{x}_i} = O(R)$, which we guarantee when choosing the initial $\bar{x}$.

The challenge then becomes efficiently mapping any τ to $\{\langle \bar{x}, (\bar{v}-\mu)^q \rangle_A\}_{q\in[p]}$ for suitable μ and A. We address this by jointly bucketing τ and $\bar{v}$. Specifically, we map τ into a bucket index $k = \lfloor \log_2 \frac{1-\kappa^\tau}{1-\kappa} \rfloor$, pick μ to be the largest integer multiple of $R/((1-\kappa)2^k)$ such that $\mu \le \max_i \bar{v}_i$, and set $A = \{i \mid |(1-\kappa)2^k(\bar{v}_i - \mu)| \le R\}$. Since $k \le k_{\max} = \lfloor \log_2 \frac{1}{1-\kappa} \rfloor = O(\log n)$, we argue that precomputing $\langle \bar{x}, (\bar{v}-\mu)^q \rangle_A$ for every possible resulting μ and A takes at most $O(\bar{n}p \log \frac{1}{1-\kappa}) = \widetilde{O}(\bar{n})$ time, which we can charge to initialization. We further show how to support deletions in $\widetilde{O}(1)$ time by carefully manipulating the precomputed quantities.

Supporting sparse updates. Building on `ScaleMaintainer`, we design the data structure `ApproxExpMaintainer` that (approximately) implements the entire mirror descent step (3.8) in time $\widetilde{O}(1)$.[3] The data structure maintains vectors $\bar{x} \in \Delta^n$ and $\bar{v} \in \mathbb{R}^n$ and $K = \lceil \log_2(n+1) \rceil$ instances of `ScaleMaintainer` denoted $\{\texttt{ScaleMaintainer}_k\}_{k\in[K]}$. The kth instance tracks a coordinate subset $S_k \subseteq [n]$ such that $\{S_k\}_{k\in[K]}$ partitions $[n]$, and has initial data $[\bar{x}]_{S_k}$ and $[\bar{v}]_{S_k}$. We let $\tau_k \ge 0$ denote the "time index" parameter of the kth instance. The data structure satisfies two invariants; first, the unnormalized iterate $\hat{x}$ satisfies

$$[\hat{x}]_{S_k} = [\bar{x} \circ \exp\{(1-\kappa^{\tau_k})\bar{v}\}]_{S_k}, \text{ for all } k \in [K]. \tag{3.15}$$

Second, the partition satisfies

$$|S_k| \le 2^k - 1 \text{ for all } k \in [K], \tag{3.16}$$

where at initialization we let $S_K = [n]$ and $S_k = \emptyset$ for $k < K$, $\bar{x} = x_0$, $\bar{v} = \frac{v}{1-\kappa} - \log x_0$ and $\tau_K = 0$.

The invariant (3.15) allows us to efficiently (in time $\widetilde{O}(K) = \widetilde{O}(1)$) query coordinates of $x_t = \hat{x}_t / \|\hat{x}_t\|_1$, since `ScaleMaintainer` allows us to approximate $\|\hat{x}_t\|_1 = \sum_{k\in[K]} Z_{\tau_k}([\bar{x}]_{S_k}, [\bar{v}]_{S_k})$ with Z as defined in (3.14). To implement the dense step (3.12), we simply increment $\tau_k \leftarrow \tau_k + 1$ for every k. Let j be the nonzero coordinate of s_t in the sparse step (3.13), and let $k \in [K]$ be such that $j \in S_k$. To implement (3.13), we delete coordinate j from $\texttt{ScaleMaintainer}_k$, and create a singleton instance $\texttt{ScaleMaintainer}_0$ maintaining $S_0 = \{j\}$ with initial data $[\bar{x}]_{S_0} = e^{s_t}\hat{x}_j$, $[\bar{v}]_{S_0} = v_j/(1-\kappa) - \log(e^{s_t}\hat{x}_j)$ and $\tau_0 = 0$. Going from $k = 1$ to $k = K$, we merge $\texttt{ScaleMaintainer}_{k-1}$ into $\texttt{ScaleMaintainer}_k$ until the invariant (3.16) holds again. For example, if before the sparse step we have $|S_1| = 1$, $|S_2| = 3$ and $|S_3| = 2$, we will perform 3 consecutive merges, so that afterwards we have $|S_1| = |S_2| = 0$ and $|S_3| = 7$.

To merge two $\texttt{ScaleMaintainer}_{k-1}$ into $\texttt{ScaleMaintainer}_k$, we let $S'_k = S_{k-1} \cup S_k$ and initialize a new `ScaleMaintainer` instance with $[\bar{x}]_{S'_k} = [\hat{x}]_{S'_k}$,[4] $[\bar{v}]_{S'_k} = [v]_{S'_k}/(1-\kappa) - \log[\hat{x}]_{S'_k}$ and $\tau_k = 0$; this takes $\widetilde{O}(|S'_k|) = \widetilde{O}(2^k)$ time due to the invariant (3.16). Noting that a merge at level k can

[3] The data structures `ApproxExpMaintainer` and `ScaleMaintainer` structure support two additional operations necessary for our algorithm: efficient approximate sampling from x_t and maintenance of a running sum of $\hat{x}_\tau$. Given the normalization constant approximation, the implementation of these operations is fairly straightforward, so we do not discuss them in the introduction.

[4] More precisely, for every $j \in S'_k$ we set $\bar{x}_j = \hat{x}_j + \varepsilon \max_{i\in S'_k} \hat{x}_i$, where ε is a small padding constant that ensures the bounded multiplicative range necessary for correct operation of `ScaleMaintainer`.

only happen once in every $\Omega(2^k)$ updates, we conclude that the amortized cost of merges at each level is $\widetilde{O}(1)$, and (since $K = \widetilde{O}(1)$), so is the cost of the sparse update.

Back to distribution design (sampling from the sum). Our data structure enables us to compute the iteration (3.8) and query coordinates of the iterates x_t and y_t in $\widetilde{O}(1)$ amortized time. However, we cannot compute $\tilde{\delta}^{\mathsf{x}}$ using the distribution (3.11) because we do not have an efficient way of sampling from $|y_t - y_0|$; Taylor approximation techniques are not effective for approximating the absolute value because it is not smooth. To overcome this final barrier, we introduce a new design which we call "sampling from the sum,"

$$p_{ij}(x, y) = \left(\frac{1}{3}y_i + \frac{2}{3}[y_0]_i\right) \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}. \tag{3.17}$$

Sampling from the modified distribution is simple, as our data structure allows us to sample from y_t. Moreover, we show that the distribution (3.17) satisfies a relaxed version of (3.9) where the LHS is replaced by a local norm as before, and the RHS is replaced by $L^2(V_{x_0}(x_t) + V_{y_0}(y_t))$, where $V_x(x')$ is the KL divergence between x and x'. In Table 3.5 we list the sampling distributions we design for variance reduction in the different domain geometries.

3.1.3 Related work

Variance reduction for matrix games. Matrix games, the canonical form of discrete zero-sum games, have long been studied in economics [426]. It is well-known that the classical mirror descent (i.e. no-regret) method yields an algorithm with running time $\widetilde{O}\left(\mathsf{nnz}(A)L^2\epsilon^{-2}\right)$ [414]. Subsequent work [253, 415, 420, 140] improve this runtime as described above. Our work builds on the extragradient scheme of [415] as well as the gradient estimation and clipping technique of [140].

[432] apply standard variance reduction [300] to bilinear ℓ_2-ℓ_2 games by sampling elements proportional to squared matrix entries. Using proximal-point acceleration they obtain a runtime of $\widetilde{O}\left(\mathsf{nnz}(A) + \|A\|_{\mathrm{F}}\sqrt{\mathsf{nnz}(A)\max\{m,n\}}\epsilon^{-1}\log\frac{1}{\epsilon}\right)$, a rate we recover using our algorithm. However, in this setting the mirror-prox method has runtime $\widetilde{O}\left(\|A\|_{\mathrm{op}}\,\mathsf{nnz}(A)\epsilon^{-1}\right)$, which may be better than the result of [432] by a factor of $\sqrt{mn/\mathsf{nnz}(A)}$ due to the discrepancy in the norm of A. Naive application of [432] to ℓ_1 domains results in even greater potential losses. [484] extend the method of [432] to smooth functions using general Bregman divergences, but their extension is unaccelerated and appears limited to a ϵ^{-2} rate.

[121] propose a variance-reduced extragradient method with applications to generative adversarial training. In contrast to our algorithm, which performs extragadient steps in the outer loop, the method of [121] performs stochastic extragradient steps in the inner loop, using finite-sum variance reduction as in [300]. [121] analyze their method in the convex-concave setting, showing improved stability over direct application of the extragradient method to noisy gradients. However, their complexity guarantees are worse than those of linear-time methods. Following up on [121], [394]

Setting	p_{ij}	q_{ij}
ℓ_1-ℓ_1	$y_i \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}$	$x_j \cdot \frac{A_{ij}^2}{\|A_{:j}\|_2^2}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{A_{ij}^2}{\|A\|_{\mathrm{F}}^2}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\propto x_j^2 \cdot \mathbf{1}_{A_{ij}\neq 0}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{A_{ij} \cdot x_j^2}{\sum_{k\in[n]} \|A_{:k}\|_1 \cdot x_k^2}$
ℓ_2-ℓ_2	$\frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{\|A_{:j}\|_1^2}{\sum_{k\in[n]} \|A_{:k}\|_1^2} \cdot \frac{\|A_{ij}\|}{\|A_{:j}\|_1}$
ℓ_2-ℓ_2	$\frac{{y_i}^2}{\|y\|_2^2} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{{x_j}^2}{\|x\|_2^2} \cdot \frac{\|A_{ij}\|}{\|A_{:j}\|_1}$

Table 3.4: The distributions p, q used in our coordinate gradient estimator. Comments: The estimator is of the form $\tilde{g}(x,y) = \left(\frac{1}{p_{ij}} y_i A_{ij} \cdot e_j, -\frac{1}{q_{lk}} A_{lk} x_k \cdot e_l\right)$ where $i, j \sim p$ and $l, k \sim q$.

Setting	p_{ij}	q_{ij}
ℓ_1-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}$	$\frac{x_j + 2[x_0]_j}{3} \cdot \frac{A_{ij}^2}{\|A_{:j}\|_2^2}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{A_{ij}^2}{\|A\|_{\mathrm{F}}^2}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\propto [x - x_0]_j^2 \cdot \mathbf{1}_{A_{ij}\neq 0}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{\|A_{ij}\| \cdot [x - x_0]_j^2}{\sum_{k\in[n]} \|A_{:k}\|_1 \cdot [x - x_0]_k^2}$
ℓ_2-ℓ_2	$\frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{\|A_{:j}\|_1^2}{\sum_{k\in[n]} \|A_{:k}\|_1^2} \cdot \frac{\|A_{ij}\|}{\|A_{:j}\|_1}$
ℓ_2-ℓ_2	$\frac{[y - y_0]_i^2}{\|y - y_0\|_2^2} \cdot \frac{\|A_{ij}\|}{\|A_{i:}\|_1}$	$\frac{[x - x_0]_j^2}{\|x - x_0\|_2^2} \cdot \frac{\|A_{ij}\|}{\|A_{:j}\|_1}$

Table 3.5: The distributions p, q used for our reduced variance coordinate gradient estimator. Comments: The estimator is of the form $\tilde{g}(x,y) = \left(A^\top y + \frac{1}{p_{ij}}(y_i - y_{0,i})A_{ij} \cdot e_j,\ -Ax - \frac{1}{q_{lk}} A_{lk}(x_k - x_{0,k}) \cdot e_l\right)$ where $i, j \sim p$ and $l, k \sim q$ and x_0, y_0 is a reference point.

propose to reduce the variance of the stochastic extragradient method by using the same stochastic sample for both the gradient and extragradient steps. In the Euclidean strongly convex case, they show a convergence guarantee with a relaxed variance assumption, and in the noiseless full-rank bilinear case they recover the guarantees of [398]. In the general convex case, however, they only show an ϵ^{-2} rate of convergence.

Coordinate methods for matrix games. Updating a single coordinate at a time—or more broadly computing only a single coordinate of the gradient at every iteration—is a well-studied and successful technique in optimization [540]. Selecting coordinates at random is key to obtaining strong performance guarantees: [498] show this for linear regression, [475] show this for ℓ_1 regularized linear models, and [423] shows this for general smooth minimization. Later works [347, 553, 425] propose accelerated coordinate methods. These works share two common themes: selecting the gradient coordinate from a non-uniform distribution (see also [455]), and augmenting the 1-sparse stochastic gradient with a dense momentum term. These techniques play important roles in our development as well.

To reap the full benefits of coordinate methods, iterations must be very cheap, ideally taking near-constant time. However, most coordinate methods require super-constant time, typically in the form of a vector-vector computation. Even works that consider coordinate methods in a primal-dual context [477, 22, 547, 407, 476] perform the coordinate updates only on the dual variable and require a vector-vector product (or more generally a component gradient computation) at every iteration.

A notable exception is the work of [531, 532] which develops a primal-dual stochastic coordinate method for solving Markov decision processes, essentially viewing them as ℓ_∞-ℓ_1 bilinear saddle-point problems. Using a tree-based ℓ_1 sampler data structure similar to the ℓ_1 sampler we use for simplex domains for the sublinear case, the method allows for $\widetilde{O}(1)$ iterations and a potentially sublinear runtime scaling as ϵ^{-2}. [499] also consider bilinear saddle-point problems and variance reduction. Unlike our work, they assume a separable domain, use uniform sampling, and do not accelerate their variance reduction scheme with extra-gradient steps. The separable domain makes attaining constant iteration cost time much simpler, since there is no longer a normalization factor to track, but it also rules out applications to the simplex domain. While [499] report promising empirical results, their theoretical guarantees do not improve upon prior work.

Our work develops coordinate methods with $\widetilde{O}(1)$ iteration cost for new types of problems. Furthermore, it maintains the iteration efficiency even in the presence of dense components arising from the update, thus allowing for acceleration via an extra-gradient scheme.

Data structures for optimization. Performing iterations in time that is asymptotically smaller than the number of variables updated at every iteration forces us to carry out the updates implicitly using data structures; several prior works employ data structures for exactly the same reason. One of the most similar examples comes from [347], who design a data structure for an accelerated coordinate method in Euclidean geometry. In our terminology, their data structure allows performing

each iteration in time $O(\mathsf{rcs})$ while implicitly updating variables of size $O(n)$. [203] design a data structure based on balanced search trees that supports efficient Euclidean projection to the ℓ_1 ball of vector of the form $u + s$ where u is in the ℓ_1 ball and s is sparse. They apply it in a stochastic gradient method for learning ℓ_1 regularized linear classifier with sparse features. Among the many applications of this data structure, [407] adapt it to efficiently compute Euclidean projections into the intersection of the simplex and a χ^2 ball for 1-sparse updates. [476] and [531, 532], among others, use binary tree data structures to perform multiplicative weights projection to the simplex and sampling from the iterates.

A recent work of [486] develops a data structure which is similar to `ApproxExpMaintainer`, for updates arising from a primal-dual method to efficiently solve ℓ_∞ regression. Their data structure was also designed to handle updates to a simplex variable which summed a structured dense component and a sparse component. However, the data structure design of that work specifically exploited the structure of the maximum flow problem in a number of ways, such as bounding the sizes of the update components and relating these bounds to how often the entire data structure should be restarted. Our data structure can handle a broader range of structured updates to simplex variables and has a much more flexible interface, which is crucial to the development of our variance-reduced methods as well as our applications.

Another notable use of data structures in optimization appears in second order methods, where a long line of work uses them to efficiently solve sequences of linear systems and approximately compute iterates [316, 27, 349, 144, 352, 515, 517]. Finally, several works on low rank optimization make use of sketches to efficiently represent their iterates and solutions [141, 543].

Numerical sparsity. Measures of numerical sparsity, such as the ℓ_2/ℓ_∞ or ℓ_1/ℓ_2 ratios, are continuous and dimensionless relaxations of the ℓ_0 norm. The *stable rank* of a matrix A measures the numerical sparsity of its singular values (specifically, their squared ℓ_2/ℓ_∞ ratio) [146].

For linear regression, stochastic methods generally outperform exact gradient methods only when A is has low stable rank, cf. discussion in [111, Section 4.3], i.e., numerically sparse singular values. In recent work, [260] develop algorithms for linear regression and eigenvector problems for matrices with numerically sparse entries (as opposed to singular values). Our work further broadens the scope of matrix problems for which we can benefit from numerical sparsity. Moreover, our results have implications for regression as well, improving on [260] in certain numerically sparse regimes.

In recent work by [48], the authors develop primal-dual sublinear methods for ℓ_1-regularized linear multi-class classification (bilinear games in ℓ_1-ℓ_∞ geometry), and obtain complexity improvements depending on the numerical sparsity of the problem. Similarly to our work, careful design of the sampling distribution plays a central role in [48]. They also develop a data structure that allows iteration cost independent of the number of classes. However, unlike our work, [48] rely on sampling entire rows and columns, have iteration costs linear in $n + m$, and do not utilize variance reduction. We believe that our techniques can yield improvements in their setting.

3.1.4 Chapter organization

In Section 3.2, we set up our terminology, notation, the interfaces of our data structures, and the different matrix access models we consider. In Section 3.3 we develop our algorithmic framework: we present coordinate stochastic gradient methods in Section 3.3.1 and their reduced variance counterparts in Section 3.3.2. In Section 3.4 we apply both methods to solving ℓ_1-ℓ_1 matrix games; we show how to implement the method using our data structures and analyze the runtime. In Section 3.5, we discuss in detail the implementation and analysis of our data structures. Finally, in Section 3.6 we specialize our results to obtain algorithms for minimum enclosing ball and maximum inscribed ball problems as well as linear regression. Many proof details as well as our algorithms for other domain setups, i.e. ℓ_2-ℓ_1 and ℓ_2-ℓ_2 are deferred to the appendix.

For brevity in the main body of this chapter, we provide our "basic" instantiation of our variance reduction framework (specialized with row-column sampling estimators, as opposed to coordinate sampling estimators) in Appendix B.5.

3.2 Preliminaries

In Section 3.2.1, we abstract the properties of the different domains we handle into a general notion of a "local norm" setup under which we develop our results. In Section 3.2.2, we give the definition and optimality criterion of the bilinear saddle-point problem we study. In Section 3.2.3, we give the matrix access models used in the algorithms we design. In Section 3.2.4, we summarize the interfaces and complexity of the data structures we design, deferring their detailed implementations to Section 3.5.

3.2.1 Local norm setups

The analyses of our algorithms cater to the geometric of each specific domain. To express our results generically, for each pair of domains $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, we define an associated "local norm setup", which contains various data tailored for our analyses. While initially this notation may appear complicated or cumbersome, later it helps avoid redundancy in the paper. Further, it clarifies the structure necessary to generalize our methods to additional domain geometries.

Definition 2. *A* local norm setup *is the quintuplet* $(\mathcal{Z}, \|\cdot\|_\cdot, r, \Theta, \text{clip})$ *, where*

1. $\mathcal{Z}$ *is a compact and convex subset of* $\mathcal{Z}^* := \mathbb{R}^n \times \mathbb{R}^m$.
2. $\|\cdot\|_\cdot$ *is a local norm: for every* $z \in \mathcal{Z}$*, the function* $\|\cdot\|_z : \mathcal{Z}^* \to \mathbb{R}_{\geq 0}$ *is a norm on* $\mathcal{Z}^*$.
3. $r : \mathcal{Z} \to \mathbb{R}$ *is a convex distance generating function: its induced Bregman divergence*

$$V_z(z') := r(z') - r(z) - \langle \nabla r(z), z' - z \rangle .$$

	ℓ_1-ℓ_1	ℓ_2-ℓ_1	ℓ_2-ℓ_2
$\mathcal{X}$	Δ^n	$\mathbb{B}^n$	$\mathbb{B}^n$
$\mathcal{Y}$	Δ^m	Δ^m	$\mathbb{B}^m$
$\|\delta\|_z$	$\sqrt{\sum_{k\in[n+m]}[z]_k[\delta]_k^2}$	$\sqrt{\|\delta^{\mathsf{x}}\|_2^2 + \sum_{i\in[m]}[z^{\mathsf{y}}]_i[\delta^{\mathsf{y}}]_i^2}$	$\|\delta\|_2$
r	$\sum_{k\in[n+m]}[z]_k \log[z]_k$	$\frac{1}{2}\|z^{\mathsf{x}}\|_2^2 + \sum_{i\in[m]}[z^{\mathsf{y}}]_i \log[z^{\mathsf{y}}]_i$	$\frac{1}{2}\|z\|_2^2$
Θ	$\log(mn)$	$\frac{1}{2} + \log(m)$	1
$\mathrm{clip}(\delta)$	$\mathrm{sign}(\delta) \circ \min\{1, \lvert\delta\rvert\}$	$(\delta^{\mathsf{x}}, \mathrm{sign}(\delta^{\mathsf{y}}) \circ \min\{1, \lvert\delta^{\mathsf{y}}\rvert\})$	δ

Table 3.6: Local norm setups. Comments: In each case, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, Δ^n is the probability simplex $\{x \mid x \in \mathbb{R}^n_{\geq 0}, \mathbf{1}_n^\top x = 1\}$, $\mathbb{B}^n$ is the Euclidean ball $\{x \mid x \in \mathbb{R}^n, \|x\|_2 \leq 1\}$, the operations sign, min, and $|\cdot|$ are performed entrywise on a vector, and $\circ$ stands for the entrywise product between vectors.

satisfies

$$\langle \gamma, z - z' \rangle - V_z(z') \leq \frac{1}{2}\|\gamma\|_*^2 := \frac{1}{2}\max_{s\in\mathcal{Z}} \|\gamma\|_s^2 \quad \textit{for all } z, z' \in \mathcal{Z} \textit{ and } \gamma \in \mathcal{Z}^*. \tag{3.18}$$

4. $\Theta = \max_{z,z'\in\mathcal{Z}}\{r(z) - r(z')\}$ *is the range of* r. *For* $z^* \in \mathrm{argmin}_{z\in\mathcal{Z}} r(z)$ *we have* Θ *is an upper bound on the range of* $V_{z^*}(z) \leq \Theta$ *for all* $z \in \mathcal{Z}$.

5. $\mathrm{clip} : \mathcal{Z}^* \to \mathcal{Z}^*$ *is a mapping that enforces a local version of* (3.18)*:*

$$|\langle \mathrm{clip}(\gamma), z - z' \rangle| - V_z(z') \leq \|\gamma\|_z^2 \quad \textit{for all } z, z' \in \mathcal{Z} \textit{ and } \gamma \in \mathcal{Z}^*, \tag{3.19}$$

and satisfies the distortion guarantee

$$|\langle \gamma - \mathrm{clip}(\gamma), z \rangle| \leq \|\gamma\|_z^2 \quad \textit{for all } z \in \mathcal{Z} \textit{ and } \gamma \in \mathcal{Z}^*. \tag{3.20}$$

Table 3.6 summarizes the three local norm setups we consider. Throughout the paper,

for a vector $z \in \mathcal{X} \times \mathcal{Y}$, we denote its $\mathcal{X}$ and $\mathcal{Y}$ blocks by z^{x} and z^{y}.

In addition, we write coordinate i of any vector v as $[v]_i$.

Proposition 8. *The quintuplets* ($\mathcal{Z}$, $\|\cdot\|_.$, r, Θ, clip) *in Table 3.6 satisfy the local norm setup requirements in Definition 2.*

While Proposition 8 is not new, for completeness and compatibility with our notation we prove it in Appendix B.1.

In each local norm setup, we slightly overload notation and use $\|\cdot\|$ (without a subscript) to

denote the dual norm of $\|\cdot\|_*$, i.e., $\|\eta\| := \max_{\delta:\|\delta\|_* \le 1} \delta^\top \eta$. In each domain geometry $\|\cdot\|$ and $\|\cdot\|_*$ are as follows:

$$\begin{array}{lll} \|\eta\| = \sqrt{\|\eta^{\mathsf{x}}\|_1^2 + \|\eta^{\mathsf{y}}\|_1^2} & \|\delta\|_* = \sqrt{\|\delta^{\mathsf{x}}\|_\infty^2 + \|\delta^{\mathsf{y}}\|_\infty^2} & \text{for } \ell_1\text{-}\ell_1 \\ \|\eta\| = \sqrt{\|\eta^{\mathsf{x}}\|_2^2 + \|\eta^{\mathsf{y}}\|_1^2} & \|\delta\|_* = \sqrt{\|\delta^{\mathsf{x}}\|_2^2 + \|\delta^{\mathsf{y}}\|_\infty^2} & \text{for } \ell_2\text{-}\ell_1 \\ \|\eta\| = \|\eta\|_2 & \|\delta\|_* = \|\delta\|_2 & \text{for } \ell_2\text{-}\ell_2\ . \end{array} \tag{3.21}$$

3.2.2 The problem and optimality criterion

Throughout, we consider the bilinear saddle point problem

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x,y), \text{ where } f(x,y) := y^\top A x + b^\top x - c^\top y, \text{ for } A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^n \text{ and } c \in \mathbb{R}^m. \tag{3.22}$$

We will always assume that every row and column of A has at least one nonzero entry (else removing said row or column does not affect the problem value), so that the number of nonzeros nnz is at least $m+n-1$. To simplify the exposition of the ℓ_1-ℓ_1 and ℓ_2-ℓ_1 setups we will assume $b = \mathbf{0}_n$ and $c = \mathbf{0}_m$ as is standard in the literature. Adding linear terms to these setups is fairly straightforward and does not affect the complexity (up to logarithmic factors) of our designed algorithms using data structures designed in this chapter (specifically `ApproxExpMaintainer` in Section 3.2.4); see Section 3.6 for an example. The gradient mapping associated with (3.22) for $z = (z^{\mathsf{x}}, z^{\mathsf{y}}) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ is

$$g(z) := (\nabla_x f(z), -\nabla_y f(z)) = (A^\top z^{\mathsf{y}} + b, -Az^{\mathsf{x}} + c). \tag{3.23}$$

The mapping g is continuous and monotone, where we call g monotone if and only if

$$\langle g(z') - g(z), z' - z\rangle \ge 0,\ \forall z, z' \in \mathcal{Z}.$$

This holds due to the convexity-concavity (indeed, bilinearity) of function f. Our goal is to design randomized algorithms for finding an (expected) ϵ-accurate saddle point $z \in \mathcal{Z}$ such that, in expectation,

$$\mathbb{E}\mathrm{Gap}(z) := \mathbb{E}\left[\max_{y'\in\mathcal{Y}} f(z^{\mathsf{x}}, y') - \min_{x'\in\mathcal{X}} f(x', z^{\mathsf{y}})\right] \le \epsilon. \tag{3.24}$$

In order to do so, we aim to find a sequence $z_1, z_2, \ldots, z_K$ with (expected) low *average regret*, i.e., such that $\mathbb{E}\max_{u\in\mathcal{Z}} \left\{\frac{1}{K}\sum_{k=1}^K \langle g(z_k), z_k - u\rangle\right\} \le \epsilon$. Due to bilinearity of f we have

$$\mathbb{E}\mathrm{Gap}\left(\frac{1}{K}\sum_{k=1}^K z_k\right) = \mathbb{E}\max_{u\in\mathcal{Z}}\left\{\frac{1}{K}\sum_{k=1}^K \langle g(z_k), z_k - u\rangle\right\} \le \epsilon.$$

Finally, we make the explicit assumption that whenever we are discussing an algorithm in this

chapter with a simplex domain (e.g. in ℓ_1-ℓ_1 or ℓ_2-ℓ_1 case), the quantity L_{co}/ϵ is bounded by $(m+n)^3$, as otherwise we are in the high-accuracy regime where the runtimes of interior point methods or cutting-plane methods [350, 296] are favorable. Specifically for ℓ_1-ℓ_1 matrix games in this regime, interior-point methods [349, 144, 517] are always faster, see footnote in Section 3.1.1. We make this assumption for notational convenience when discussing logarithmic factors depending on multiple quantities, such as m, n, L_{co}, and ϵ^{-1}.

3.2.3 Matrix access models

We design randomized algorithms which require accessing and sampling from the matrix $A \in \mathbb{R}^{n\times m}$ in a variety of ways. Here, we list these operations, where we assume each takes constant time. Specific algorithms only require access to a subset of this list; we make a note of each algorithm's requirements when presenting it.

A1. For $i, j \in [m] \times [n]$, return A_{ij}.

A2. For $i \in [m]$ and $p \in \{1, 2\}$, draw $j \in [n]$ with probability $|A_{ij}|^p / \|A_{i:}\|_p^p$.

A3. For $j \in [n]$ and $p \in \{0, 1, 2\}$, draw $i \in [m]$ with probability $|A_{ij}|^p / \|A_{:j}\|_p^p$.

A4. For $i \in [m]$ ($j \in [n]$) and $p \in \{1, 2\}$, return $\|A_{i:}\|_p$ ($\|A_{:j}\|_p$).

A5. For $p \in \{1, 2\}$, return $\max_{i\in[m]} \|A_{i:}\|_p$, $\max_{j\in[n]} \|A_{:j}\|_p$, nnz, rcs, and $\|A\|_{\mathrm{F}}$.

Given any representation of the matrix as a list of nonzero entries and their indices, we can always implement the access modes above (in the assumed constant time) with $O(\mathsf{nnz})$ time preprocessing; see e.g. [526] for an implementation of the sampling (in a unit cost RAM model). Our variance-reduced algorithms have an additive $O(\mathsf{nnz})$ term appearing in their runtimes due to the need to compute at least one matrix-vector product to implement gradient estimators. Thus, their stated runtime bounds hold independently of matrix access assumptions.

3.2.4 Data structure interfaces

We rely on data structures to maintain and sample from the iterates of our algorithms. Below, we give a summary of the operations supported by our data structures and their runtime guarantees. We show how to implement these data structures in Section 3.5.

`IterateMaintainer`$_p$

Given $p \in \{1, 2\}$, we design a data structure `IterateMaintainer`$_p$ which maintains an implicit representation of the current iterate $x \in \mathbb{R}^n$ and a running sum s of all iterates. At initialization,

Category	Function	Runtime
initialize	`Init`(x_0, v): $x \leftarrow x_0$, $s \leftarrow 0$	$O(n)$
update	`Scale`(c): $x \leftarrow cx$	$O(1)$
	`AddSparse`(j, c): $[x]_j \leftarrow [x]_j + c$ (if $p = 1$, we require $c \geq -[x]_j$)	$O(\log n)^\dagger$
	`AddDense`(c): $x \leftarrow x + cv$ (supported if $p = 2$)	$O(1)$
	`UpdateSum`(): $s \leftarrow s + x$	$O(1)$
query	`Get`(j): Return $[x]_j$	$O(1)$
	`GetSum`(j): Return $[s]_j$	$O(1)$
	`GetNorm`(): Return $\|x\|_p$	$O(1)$
sample†	`Sample`(): Return j with probability $[x]_j^p / \|x\|_p^p$	$O(\log n)$

† An alternative implementation does not support `Sample`, but performs `AddSparse` in time $O(1)$.

this data structure takes as input the initial iterate x_0 to be maintained and for $p = 2$, the data structure also takes as input a fixed vector v. It then supports the following operations.

The implementation of `IterateMaintainer`$_p$ is given in Section 3.5.1. In Sections B.3.2 and B.4.3 we use variants of this data structure `WeightedIterateMaintainer`$_2$ and `CenteredIterateMaintainer`$_2$, and defer the detailed discussions of their implementations to Appendix B.8.

`ApproxExpMaintainer`

To maintain multiplicative weights updates with a fixed dense component, we design a data structure `ApproxExpMaintainer` initialized with an arbitrary point $x_0 \in \Delta^n$, a direction $v \in \mathbb{R}^n$, a decay constant $\kappa \in [0, 1]$ and an approximation error parameter ε. In order to specify the implementation of our data structure, we require the following definition.

Definition 3 (β-padding)**.** *For $x, x' \in \Delta^n$, we say x' is a β-padding of x if $x' = \tilde{x} / \|\tilde{x}\|_1$, for a point $\tilde{x} \in \mathbb{R}^n_{\geq 0}$ with $\tilde{x} \geq x$ entrywise and $\|\tilde{x} - x\|_1 \leq \beta$.*

Notions similar to β-padding appear in previous literature [331]. A key technical property of β-paddings is that they do not increase entropy significantly (see Lemma 34).

`ApproxExpMaintainer` has maintains two vectors $x, \hat{x} \in \Delta^n$ that, for an error tolerance parameter ε, satisfy the invariant

$$\hat{x} \text{ is a } \varepsilon\text{-padding of } x. \tag{3.25}$$

an error tolerance parameter ε. We now specify the interface, where $\circ$ denotes elementwise product, $[x^\kappa]_j = [x]_j^\kappa$ denotes elementwise power, $\Pi_\Delta(z) = z / \|z\|_1$ normalizes $z \in \mathbb{R}^n_{\geq 0}$ to lie in the simplex, and $\|s\|_0$ denotes the number of nonzeroes in vector s. To state our runtimes, we define

$$\omega := \max\left(\frac{1}{1-\kappa}, \frac{n}{\lambda\varepsilon}\right).$$

For most of our applications of `ApproxExpMaintainer`, ω is a polynomial in m and n (our iterate dimensions), so $\log(\omega) = O(\log(mn))$ (with the exception of our maximum inscribed ball application, where our runtimes additionally depend polylogarithmically on the size of the hyperplane shifts b; see Remark 2). We defer a more fine-grained runtime discussion to Section 3.5.3.

Category	Function	Runtime
initialize	`Init`$(x_0, v, \kappa, \varepsilon, \lambda) : \kappa \in [0,1)$, $\varepsilon > 0$, $\min_j [x_0]_j \geq \lambda$	$O(n \log n \log^2 \omega)$
update	`MultSparse`(g): $x \leftarrow \varepsilon$-padding of $\Pi_\Delta(x \circ \exp(g))$	$O(\|g\|_0 \log^2 n \log^2 \omega)$
	`DenseStep`(): $x \leftarrow \Pi_\Delta(x^\kappa \circ \exp(v))$	$O(\log n)$
	`UpdateSum`(): $s \leftarrow s + \hat{x}$ (recall invariant (3.25))	$O(\log n \log \omega)$
query	`Get`(j): Return $[\hat{x}]_j$	$O(\log n \log \omega)$
	`GetSum`(j): Return $[s]_j$	$O(\log^2 \omega)$
sample	`Sample`(): Return j with probability $[\hat{x}]_j$	$O(\log n \log \omega)$

The role of `ApproxExpMaintainer` is in to efficiently implement the regularized and reduced-variance stochastic mirror descent steps of the form (3.8). To do this, we initialize the data structure with $v = (1-\kappa)\log x_0 - \eta\kappa g_0^\times$. Then, the iteration (3.8) consists of calling `DenseStep`() followed by `MultSparse`$(-\eta\kappa\tilde{\delta}^\times)$.

3.3 Framework

In this section, we develop our algorithmic frameworks. The resulting algorithms have either sublinear or variance-reduced complexities. We develop our sublinear coordinate method framework in Section 3.3.1, and its variance-reduced counterpart in Section 3.3.2.

3.3.1 Sublinear coordinate methods

In Section 3.3.1 we introduce the concept of a *local gradient estimator*, which allow stronger guarantees for stochastic mirror descent with clipping (Algorithm 10) via local norms analysis. Then, in Section 7 we state the form of the specific local gradient estimators we use in our coordinate methods, and motivate the values of L_{co} in Table 3.1.

Convergence analysis

Definition 4. *For local norm setup ($\mathcal{Z}$, $\|\cdot\|_\cdot$, r, Θ, clip) , we call a stochastic gradient estimator $\tilde{g} : \mathcal{Z} \to \mathcal{Z}^*$ an* L-local estimator *if it satisfies the following properties for all $z \in \mathcal{Z}$:*

1. *Unbiasedness: $\mathbb{E}[\tilde{g}(z)] = g(z)$.*
2. *Second moment bound: for all $w \in \mathcal{Z}$, $\mathbb{E}[\|\tilde{g}(z)\|_w^2] \leq L^2$.*

The following lemma shows that L-local estimators are unbiased for L-bounded operators.

Lemma 30. *A gradient mapping that admits an L-local estimator satisfies $\|g(z)\|_* \leq L$ for all $z \in \mathcal{Z}$.*

Proof. For every $z \in \mathcal{Z}$, the function $\|\cdot\|_z^2$ is convex. Thus by Jensen's inequality,

$$\|g(z)\|_z^2 = \|\mathbb{E}\tilde{g}(z)\|_z^2 \leq \mathbb{E}\,\|\tilde{g}(z)\|_z^2 \leq L^2.$$

Taking supremum over $z \in \mathcal{Z}$ gives $\|g(z)\|_*^2 \leq L^2$. □

We note that the same result *does not* hold for $\tilde{g}$ because maximum and expectation do not commute. That is, $\mathbb{E}\,\|\tilde{g}\|_*^2$ is not bounded by L^2. This fact motivates our use of local norms analysis.

Below, we state Algorithm 10, stochastic mirror descent with clipping, and a guarantee on its rate of convergence using local gradient estimators. We defer the proof to Appendix B.2 and note here that it uses the "ghost iterates" technique due to [416] in order to rigorously bound the expected regret with respect to the best response to our iterates, rather than a pre-specified point. This technique is purely analytical and does not affect the algorithm. We also note that the second inequality in Proposition 9 holds with any convex-concave function f; the first uses bilinearity of our problem structure.

Algorithm 10: Stochastic mirror descent

Input: Matrix $A \in \mathbb{R}^{m\times n}$, L-local gradient estimator $\tilde{g}$, clipping function $\mathrm{clip}(\cdot)$;
Output: A point with $O(\frac{\Theta}{\eta T} + \eta L^2)$ expected duality gap ;
Parameters: Step-size η, number of iterations T;
$z_0 \leftarrow \mathrm{argmin}_{z\in\mathcal{Z}} r(z)$;
for $t = 1, \ldots, T$ **do**
  $z_t \leftarrow \arg\min_{z\in\mathcal{Z}} \left\{\langle \mathrm{clip}(\eta\tilde{g}(z_{t-1})), z\rangle + V_{z_{t-1}}(z)\right\}$;
Return: $\frac{1}{T+1}\sum_{t=0}^{T} z_t$;

Proposition 9. *Let $(\mathcal{Z}, \|\cdot\|_\cdot, r, \Theta, \mathrm{clip})$ be a local norm setup, let $L, \epsilon > 0$, and let $\tilde{g}$ be an L-local estimator. Then, for $\eta \leq \frac{\epsilon}{9L^2}$ and $T \geq \frac{6\Theta}{\eta\epsilon} \geq \frac{54L^2\Theta}{\epsilon^2}$, Algorithm 10 outputs a point $\bar{z}$ such that*

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \leq \mathbb{E}\left[\sup_{u\in\mathcal{Z}} \frac{1}{T+1}\sum_{t=0}^{T} \langle g(z_t), z_t - u\rangle\right] \leq \epsilon.$$

Coordinate gradient estimators

We now state the general form which our local gradient estimators $\tilde{g}$ take. At a point $z \in \mathcal{Z}$, for specified sampling distributions $p(z), q(z)$, sample $i^{\mathsf{x}}, j^{\mathsf{x}} \sim p(z)$ and $i^{\mathsf{y}}, j^{\mathsf{y}} \sim q(z)$. Then, define

$$\tilde{g}(z) := \left(\frac{A_{i^{\mathsf{x}} j^{\mathsf{x}}} [z^{\mathsf{y}}]_{i^{\mathsf{x}}}}{p_{i^{\mathsf{x}} j^{\mathsf{x}}}(z)} e_{j^{\mathsf{x}}}, \frac{-A_{i^{\mathsf{y}} j^{\mathsf{y}}} [z^{\mathsf{x}}]_{j^{\mathsf{y}}}}{q_{i^{\mathsf{y}} j^{\mathsf{y}}}(z)} e_{i^{\mathsf{y}}} \right) + g(0) \quad \text{where} \quad g(0) = (b, c). \tag{3.26}$$

It is clear that regardless of the distributions $p(z), q(z)$, for the gradient operator in (I.20), $\tilde{g}(z)$ is an unbiased gradient estimator ($\mathbb{E}[\tilde{g}(z)] = g(z)$) and $\tilde{g}(z) - g(0)$ is 2-sparse.

Optimal values of L_{co}. In the remainder of this section we assume for simplicity the $g(0) = 0$ (i.e. the objective f in (3.22) has not linear terms). Here we compute the optimal values of L for local gradient estimators (see Definition 4) of the form (3.26) for each of the local norm setups we consider. This motivates the values of L_{co} we derive in the following sections. First, in the ℓ_1-ℓ_1 case, the second moment of $\|\tilde{g}^{\mathsf{x}}(z)\|_{w^{\mathsf{x}}}^2$ (the local norm of the $\mathcal{X}$ block of $\tilde{g}(z)$ at point w) is

$$\mathbb{E}\left[[w^{\mathsf{x}}]_{j^{\mathsf{x}}} \left(\frac{A_{i^{\mathsf{x}} j^{\mathsf{x}}} [z^{\mathsf{y}}]_{i^{\mathsf{x}}}}{p_{i^{\mathsf{x}} j^{\mathsf{x}}}(z)} \right)^2 \right] = \sum_{i \in [m], j \in [n]} \frac{A_{ij}^2 [z^{\mathsf{y}}]_i^2 [w^{\mathsf{x}}]_j}{p_{ij}(z)} \geq \left(\sum_{i \in [m], j \in [n]} |A_{ij}| [z^{\mathsf{y}}]_i \sqrt{[w^{\mathsf{x}}]_j} \right)^2.$$

Since $z^{\mathsf{y}} \in \Delta^m$ and $\sqrt{w^{\mathsf{x}}} \in \mathbb{B}^n$ with $\left\| \sqrt{w^{\mathsf{x}}} \right\|_2 = 1$, the above lower bound is in the worst case $\max_i \|A_{i:}\|_2^2$. Similarly, the best possible bound on the $\mathcal{Y}$ is $\max_j \|A_{:j}\|_2^2$. Therefore, in the ℓ_1-ℓ_1 setup, no local estimator has parameter L smaller than L_{co} in Table 3.1.

Next, in the ℓ_2-ℓ_2 case, the (ℓ_2) second moment of the $\mathcal{X}$ block is

$$\mathbb{E}\left[\left(\frac{A_{i^{\mathsf{x}} j^{\mathsf{x}}} [z^{\mathsf{y}}]_{i^{\mathsf{x}}}}{p_{i^{\mathsf{x}} j^{\mathsf{x}}}(z)} \right)^2 \right] = \sum_{i \in [m], j \in [n]} \frac{A_{ij}^2 [z^{\mathsf{y}}]_i^2}{p_{ij}(z)} \geq \left(\sum_{i \in [m], j \in [n]} |A_{ij}| [z^{\mathsf{y}}]_i \right)^2 = \left(\sum_{i \in [m]} \|A_{i:}\|_1 [z^{\mathsf{y}}]_i \right)^2.$$

In the worst case, this is at least $(\sum_{i \in [m]} \|A_{i:}\|_1)^2$; similarly, the best second moment bound for the $\mathcal{Y}$ block is $(\sum_{j \in [n]} \|A_{:j}\|_1)^2$, which means that L_{co} is similarly unimprovable in the ℓ_2-ℓ_2 setup.

Finally, in the ℓ_2-ℓ_1 case, where $\mathcal{X} = \mathbb{B}^n$ and $\mathcal{Y} = \Delta^m$, we again have that the ℓ_2 second moment of the $\mathcal{X}$ (ball) block is at least

$$\mathbb{E}\left[\left(\frac{A_{i^{\mathsf{x}} j^{\mathsf{x}}} [z^{\mathsf{y}}]_{i^{\mathsf{x}}}}{p_{i^{\mathsf{x}} j^{\mathsf{x}}}(z)} \right)^2 \right] \geq \left(\sum_{i \in [m], j \in [n]} |A_{ij}| [z^{\mathsf{y}}]_i \right)^2.$$

Here, since $z^{\mathsf{y}} \in \Delta^m$, the worst-case lower bound of the variance is $\max_i \|A_{i:}\|_1^2$. Further, the local

norm (at w) second moment of the $\mathcal{Y}$ (simplex) block is at least

$$\mathbb{E}\left[[w^{\mathsf{y}}]_{i^{\mathsf{y}}}\left(\frac{A_{i^{\mathsf{y}}j^{\mathsf{y}}}[z^{\mathsf{x}}]_{j^{\mathsf{y}}}}{q_{i^{\mathsf{y}}j^{\mathsf{y}}}(z)}\right)^2\right] \geq \left(\sum_{i\in[m],j\in[n]}|A_{ij}|[z^{\mathsf{x}}]_j\sqrt{[w^{\mathsf{y}}]_i}\right)^2.$$

Since $z^{\mathsf{x}} \in \mathbb{B}^n$ and $\sqrt{w^{\mathsf{y}}} \in \mathbb{B}^m$, in the worst case this second moment can be as high as $\||A|\|_{\text{op}}$, where we use $|A|$ to denote the elementwise absolute value of A. This is better than the L_{co} in Table 3.1, suggesting there is room for improvement here. However, the sampling probabilities inducing this optimal variance bound are of the form

$$q_{ij}(z;w) \propto |A_{ij}|\sqrt{[w^{\mathsf{y}}]_i}\cdot[z^{\mathsf{x}}]_j,$$

and it unclear how to efficiently sample from this distribution. Improving our ℓ_2-ℓ_1 gradient estimator (or proving that no improvement is possible) remains an open problem.

3.3.2 Variance-reduced coordinate methods

In this section, we develop the algorithmic framework we use in our variance-reduced methods. We first define a type of "centered-local" gradient estimator, modifying the local gradient estimators of the previous section. We then give the general form of a variance-reduced method and analyze it in the context of our gradient estimators and the error incurred by our data structure maintenance.

General convergence result

Definition 5. *For local norm setup ($\mathcal{Z}$, $\|\cdot\|_\cdot$, r, Θ, clip), and given a reference point $w_0 = (w_0^{\mathsf{x}}, w_0^{\mathsf{y}})$, we call a stochastic gradient estimator $\tilde{g}_{w_0} : \mathcal{Z} \to \mathcal{Z}^*$ an* L-centered-local estimator *if it satisfies the following properties:*

1. *Unbiasedness: $\mathbb{E}[\tilde{g}_{w_0}(z)] = g(z)$.*
2. *Relative variance bound: for all $w \in \mathcal{Z}$, $\mathbb{E}[\|\tilde{g}_{w_0}(z) - g(w_0)\|_w^2] \leq L^2 V_{w_0}(z)$.*

Remark 1. *Similarly to Lemma 30, a gradient mapping that admits an L-centered-local estimator also satisfies $\|g(z) - g(w_0)\|_*^2 \leq L^2 V_{w_0}(z)$, by Jensen's inequality.*

Algorithm 11 below is an approximate variant of the variance reduction algorithm in an earlier version of this work [111] which closely builds upon the "conceptual prox-method" of [415]. However, for self-containment of this chapter we specialize our presentation of our variance reduction framework to our implementation in the coordinate sampling setting.

The algorithm repeatedly calls a stochastic oracle $\mathcal{O} : \mathcal{Z} \to \mathcal{Z}$ to produce intermediate iterates, and then performs an extragradient (linearized) proximal step using the intermediate iterate. The

main modification compared to [111] is Line 8, which accommodates slight perturbations to the extra-gradient step results. These perturbations arise due to input requirements of our data structures: we slightly pad coordinates in simplex blocks to ensure they are bounded away from zero.

Algorithm 11: `OuterLoop`($\mathcal{O}$) (conceptual prox-method [415])

1 **Input:** Target approximation quality $\varepsilon_{\text{outer}}$, $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle $\mathcal{O}(z)$ for gradient mapping g and some $\varepsilon_{\text{inner}} < \varepsilon_{\text{outer}}$, distance-generating r;
2 **Parameters:** Number of iterations K;
3 **Output:** Point $\bar{z}_K$ with $\mathbb{E}\,\text{Gap}(\bar{z}) \le \frac{\alpha\Theta}{K} + \varepsilon_{\text{outer}}$;
4 $z_0 \leftarrow \text{argmin}_{z\in\mathcal{Z}} r(z)$;
5 **for** $k = 1, \ldots, K$ **do**
6 $\quad z_{k-1/2} \leftarrow \mathcal{O}(z_{k-1})$ ▷ We implement $\mathcal{O}(z_{k-1})$ by calling `InnerLoop`$(z_{k-1}, \tilde{g}_{z_{k-1}}, \alpha)$;
7 $\quad z_k^\star := \text{Prox}^\alpha_{z_{k-1}}(g(z_{k-1/2})) = \text{argmin}_{z\in\mathcal{Z}} \left\{ \left\langle g\left(z_{k-1/2}\right), z\right\rangle + \alpha V_{z_{k-1}}(z) \right\}$;
8 $\quad z_k \leftarrow$ any point satisfying $V_{z_k}(u) - V_{z_k^\star}(u) \le \frac{\varepsilon_{\text{outer}} - \varepsilon_{\text{inner}}}{\alpha}$, for all $u \in \mathcal{Z}$
9 **Return:** $\bar{z}_K = \frac{1}{K}\sum_{k=1}^K z_{k-1/2}$;

The following definition summarizes the key property of the oracle $\mathcal{O}$.

Definition 6. *Let operator g be monotone and $\alpha, \varepsilon_{\text{inner}} > 0$. An $(\alpha, \varepsilon_{\text{inner}})$-*relaxed proximal oracle *for g is a (possibly randomized) map $\mathcal{O} : \mathcal{Z} \to \mathcal{Z}$ such that $z' = \mathcal{O}(z)$ satisfies*

$$\mathbb{E}\left[\max_{u\in\mathcal{Z}} \left\{ \langle g(z'), z' - u\rangle - \alpha V_z(u) \right\}\right] \le \varepsilon_{\text{inner}}.$$

The following proposition shows that despite the error permitted tolerated in Line 8, the algorithm still converges with rate $1/K$. We defer its proof to Appendix B.2.

Proposition 10. *Let $\mathcal{O}$ be an $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle with respect to gradient mapping g, distance-generating r with range at most Θ and some $\varepsilon_{\text{inner}} \le \varepsilon_{\text{outer}}$. Let $z_{1/2}, z_{3/2}, \ldots, z_{K-1/2}$ be iterates of Algorithm 11 and let $\bar{z}_K$ be its output. Then*

$$\mathbb{E}\,\text{Gap}(\bar{z}_K) \le \mathbb{E}\max_{u\in\mathcal{Z}} \frac{1}{K}\sum_{k=1}^K \left\langle g(z_{k-1/2}), z_{k-1/2} - u\right\rangle \le \frac{\alpha\Theta}{K} + \varepsilon_{\text{outer}}.$$

Algorithm 12 is a variant of the variance-reduced inner loop of an earlier version of this work [111], adapted for local norms and inexact iterates (again, due to approximations made by the data structure). It tolerates error in three places:

1. Instead of estimating the gradient at the previous iterate w_{t-1}, we estimate it at a point $\hat{w}_{t-1}$ such that $w_{t-1} - \hat{w}_{t-1}$ has small norm and similar divergence from the reference point w_0 (Line 5).

2. Instead of letting the next iterate be the exact mirror descent step $w_t^\star$, we let be a point w_T that is close to $w_t^\star$ in norm and has similar divergences to from w_0 and to any any point in $\mathcal{Z}$ (Line 7).

3. The output $\tilde{w}$ can be an approximation of the average of the iterates, as long as its difference to the true average has bounded norm (Line 8).

We quantify the effect of these approximations in Proposition 11, which gives a runtime guarantee for Algorithm 12 (where we recall the definition of $\|\cdot\|$ as the dual norm of $\|\cdot\|_*$, see (3.21)). The proof is deferred to Appendix B.2.

Algorithm 12: `InnerLoop`$(w_0, \tilde{g}_{w_0}, \varphi)$

1 **Input:** Initial $w_0 \in \mathcal{Z}$, L-centered-local gradient estimator $\tilde{g}_{w_0}$, oracle quality $\alpha > 0$;
2 **Parameters:** Step size η, number of iterations T, approximation tolerance φ;
3 **Output:** Point $\tilde{w}$ satisfying Definition 6;
4 **for** $t = 1, \ldots, T$ **do**
5 $\quad \hat{w}_{t-1} \approx w_{t-1}$ satisfying (a) $V_{w_0}(\hat{w}_{t-1}) - V_{w_0}(w_{t-1}) \le \frac{\varphi}{\alpha}$ and (b) $\|\hat{w}_{t-1} - w_{t-1}\| \le \frac{\varphi}{LD}$;
6 $\quad w_t^\star \leftarrow \operatorname{argmin}_{w \in \mathcal{Z}} \left\{ \langle w, \operatorname{clip}(\eta \tilde{g}_{w_0}(\hat{w}_{t-1}) - \eta g(w_0)) + \eta g(w_0) \rangle + \frac{\alpha\eta}{2} V_{w_0}(w) + V_{w_{t-1}}(w) \right\}$;
7 $\quad w_t \approx w_t^\star$ satisfying
$\quad\quad$ (a) $\max_u \left[V_{w_t}(u) - V_{w_t^\star}(u) \right] \le \eta\varphi$,
$\quad\quad$ (b) $V_{w_0}(w_t) - V_{w_0}(w_t^\star) \le \frac{\varphi}{\alpha}$, and
$\quad\quad$ (c) $\|w_t - w_t^\star\| \le \frac{\varphi}{2LD}$
8 **Return:** $\tilde{w} \approx \frac{1}{T} \sum_{t=1}^T w_t$ satisfying $\left\| \tilde{w} - \frac{1}{T} \sum_{t=1}^T w_t \right\| \le \frac{\varphi}{LD}$.

Proposition 11. *Let $(\mathcal{Z}, \|\cdot\|, r, \Theta, \text{clip})$ be any local norm setup. Let $w_0 \in \mathcal{Z}$, $\alpha \ge \varepsilon_{\text{inner}} > 0$, and $\tilde{g}_{w_0}$ be an L-centered-local estimator for some $L \ge \alpha$. Assume the domain is bounded by $\max_{z \in \mathcal{Z}} \|z\| \le D$, that g is L-Lipschitz, i.e. $\|g(z) - g(z')\|_* \le L \|z - z'\|$, that g is LD-bounded, i.e. $\max_{z \in \mathcal{Z}} \|g(z)\|_* \le LD$, and that $\hat{w}_0 = w_0$. Then, for $\eta = \frac{\alpha}{10L^2}$, $T \ge \frac{6}{\eta\alpha} \ge \frac{60L^2}{\alpha^2}$, and $\varphi = \frac{\varepsilon_{\text{inner}}}{6}$, Algorithm 12 outputs a point $\hat{w} \in \mathcal{Z}$ such that*

$$\mathbb{E} \max_{u \in \mathcal{Z}} \left[\langle g(\tilde{w}), \tilde{w} - u \rangle - \alpha V_{w_0}(u) \right] \le \varepsilon_{\text{inner}}, \tag{3.27}$$

i.e. Algorithm 12 is an $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle.

Remark 2 (Assumption of boundedness on g)**.** *The assumption that g is LD-bounded in the dual norm is immediate from other assumptions used in Proposition 11 in the case of the applications in Section 3.4, where we develop methods for solving ℓ_1-ℓ_1 matrix games and assume that $g(0) = 0$. In applications in Section 3.6, due to the existence of extra linear terms b, $c \neq 0$, all complexity bounds will have an additional dependence on $\log(\|[b; c]\|_*)$ which we pay in the implementation of data structure* `ApproxExpMaintainer` *(i.e. the parameter L in the bound on g is larger if $\|[b; c]\|_*$ is large). We hide this extra polylogarithmic factor in the $\widetilde{O}$ notation.*

We also remark that (up to constants) the bounds on the range of $\varepsilon_{\text{inner}} \le \alpha \le L$ in the statement of Proposition 11 correspond to the cases where the inner and outer loop consist of a single iteration.

Variance-reduced coordinate gradient estimators

We now state the general form which our centered-local estimators $\tilde{g}_{w_0}$ take, given a reference point $w_0 \in \mathcal{Z}$. At a point z, for sampling distributions $p(z; w_0), q(z; w_0)$ to be specified, sample $i^{\mathsf{x}}, j^{\mathsf{x}} \sim p(z; w_0)$ and $i^{\mathsf{y}}, j^{\mathsf{y}} \sim q(z; w_0)$. Then, define

$$\tilde{g}_{w_0}(z) = \left(\frac{A_{i^{\mathsf{x}} j^{\mathsf{x}}} [z^{\mathsf{y}} - w_0^{\mathsf{y}}]_{i^{\mathsf{x}}}}{p_{i^{\mathsf{x}} j^{\mathsf{x}}}(z; w_0)} e_{j^{\mathsf{x}}}, \frac{-A_{i^{\mathsf{y}} j^{\mathsf{y}}} [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_{j^{\mathsf{y}}}}{q_{i^{\mathsf{y}} j^{\mathsf{y}}}(z; w_0)} e_{i^{\mathsf{y}}} \right) + g(w_0). \tag{3.28}$$

It is clear that regardless of the distributions $p(z; w_0), q(z; w_0)$, this is an unbiased gradient estimator ($\mathbb{E}[\tilde{g}_{w_0}(z)] = g(z)$). Furthermore, $\tilde{g}_{w_0}(z) - g(w_0)$ is always 2-sparse.

Finally, we remark that we give another instantiation of our variance reduction framework using gradient estimators based on sampling rows and columns in Appendix B.5. While the resulting complexity bounds do not improve upon our coordinate methods (up to logarithmic factors), the estimator design and analysis is conceptually simpler, and the interested reader may first read Appendix B.5 as a precursor to our coordinate-based sampling methods.

3.4 Matrix games

In this section we instantiate the algorithmic framework of Section 3.3 in ℓ_1-ℓ_1 setup without linear terms, i.e. $b = c = 0$ in the objective (3.22). This is the fundamental "matrix game" problem

$$\min_{x \in \Delta^m} \max_{y \in \Delta^n} y^\top A x.$$

We give two algorithms for approximately solving matrix games. In Section 3.4.1 we develop a stochastic coordinate method based on Algorithm 10 with potentially sublinear runtime $\widetilde{O}\left((L_{\mathsf{co}}^{1,1}/\epsilon)^2\right)$. In Section B.3 we develop a coordinate variance-reduction based on Algorithm 11 with runtime $\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}}^{1,1}/\epsilon\right)$ that improves on the former runtime whenever it is $\Omega(\mathsf{nnz})$. In both cases we have

$$L_{\mathsf{co}}^{1,1} := \max\left\{ \max_i \|A_{i:}\|_2, \max_j \|A_{:j}\|_2 \right\} \tag{3.29}$$

as in Table 3.1.

Instantiations for the ℓ_2-ℓ_1 and ℓ_2-ℓ_2 setups follow similarly. We carry them out in Appendices B.3 (for stochastic coordinate methods) and B.4 (for variance reduction methods).

Remark 3. *For simplicity in this section (and the remaining implementations in Appendices B.3, B.4), we will set $g(0) = 0$ whenever the setup is not ℓ_2-ℓ_2, as is standard in the literature. We defer a discussion of how to incorporate arbitrary linear terms in simplex domains to Section 3.6; up to additional logarithmic terms in the runtime, this extension is supported by* `ApproxExpMaintainer`.

Assumptions. Throughout (for both Sections 3.4.1 and 3.4.2), we assume access to entry queries,

ℓ_2 norms of rows and columns, and ℓ_2 sampling distributions for all rows and columns. We use the ℓ_1-ℓ_1 local norm setup (Table 3.6). We also define $L_{\max} := \|A\|_{\max} = \max_{i\in[m],j\in[n]} |A_{ij}|$.

3.4.1 ℓ_1-ℓ_1 sublinear coordinate method

Gradient estimator

For $z \in \Delta^n \times \Delta^m$ and desired accuracy $\epsilon > 0$, we specify the sampling distributions $p(z), q(z)$:

$$p_{ij}(z) := [z^{\mathsf{y}}]_i \frac{A_{ij}^2}{\|A_{i:}\|_2^2} \quad \text{and} \quad q_{ij}(z) := [z^{\mathsf{x}}]_j \frac{A_{ij}^2}{\|A_{:j}\|_2^2}. \tag{3.30}$$

We first state and prove the local properties of this estimator.

Lemma 31. *In the ℓ_1-ℓ_1 setup, estimator (3.26) using the sampling distribution in (3.30) is a $\sqrt{2}L_{\mathsf{co}}^{1,1}$-local estimator.*

Proof. Unbiasedness holds by definition. For arbitrary w^{x}, we have the variance bound:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{g}^{\mathsf{x}}(z)\|_{w^{\mathsf{x}}}^2\right] &\le \sum_{i\in[m],j\in[n]} p_{ij}(z)\cdot\left([w^{\mathsf{x}}]_j\cdot\left(\frac{A_{ij}[z^{\mathsf{y}}]_i}{p_{ij}(z)}\right)^2\right) = \sum_{i\in[m],j\in[n]} [w^{\mathsf{x}}]_j \frac{A_{ij}^2[z^{\mathsf{y}}]_i^2}{p_{ij}(z)} \\ &\le \sum_{i\in[m],j\in[n]} [w^{\mathsf{x}}]_j[z^{\mathsf{y}}]_i \|A_{i:}\|_2^2 \le \max_{i\in[m]} \|A_{i:}\|_2^2 \le (L_{1,1}^{\mathrm{co}})^2.\end{aligned}$$

Similarly, we have

$$\mathbb{E}\left[\|\tilde{g}^{\mathsf{y}}(z)\|_{w^{\mathsf{y}}}^2\right] \le (L_{1,1}^{\mathrm{co}})^2.$$

The definition $\|\tilde{g}(z)\|_w^2 = \|\tilde{g}^{\mathsf{x}}(z)\|_{w^{\mathsf{x}}}^2 + \|\tilde{g}^{\mathsf{y}}(z)\|_{w^{\mathsf{y}}}^2$ yields the claimed variance bound. □

Implementation details

In this section, we discuss the details of how to leverage the $\texttt{IterateMaintainer}_1$ data structure to implement the iterations of our algorithm. The algorithm we analyze is Algorithm 10, using the local estimator defined in (3.26), and the distribution (3.30). We choose

$$\eta = \frac{\epsilon}{18\left(L_{\mathsf{co}}^{1,1}\right)^2} \text{ and } T = \left\lceil \frac{6\Theta}{\eta\epsilon} \right\rceil \ge \frac{108\left(L_{\mathsf{co}}^{1,1}\right)^2 \log(mn)}{\epsilon^2}.$$

Lemma 31 implies that our estimator satisfies the remaining requirements for Proposition 9, giving the duality gap guarantee in T iterations. In order to give a runtime bound, we claim that each iteration can be implemented in $\log(mn)$ time, with $O(m+n)$ additional runtime.

Data structure initializations and invariants. At the start of the algorithm, we spend $O(m+n)$ time initializing data structures via $\texttt{IM}_1^{\mathsf{x}}.\texttt{Init}(\frac{1}{n}\mathbf{1}_n, \mathbf{0}_n)$ and $\texttt{IM}_1^{\mathsf{y}}.\texttt{Init}(\frac{1}{m}\mathbf{1}_m, \mathbf{0}_m)$, where $\texttt{IM}_1^{\mathsf{x}}, \texttt{IM}_1^{\mathsf{y}}$ are

appropriate instantiations of $\texttt{IterateMaintainer}_1$ data structures. Throughout, we preserve the invariant that the points maintained by $\texttt{IM}_1^{\mathsf{x}}, \texttt{IM}_1^{\mathsf{y}}$ correspond to the x and y blocks of the current iterate z_t at iteration t of the algorithm.

Iterations. For simplicity, we only discuss the runtime of updating the x block as the y block follows symmetrically. We divide each iteration into the following substeps, each of which we show runs in time $O(\log mn)$. We refer to the current iterate by $z = (z^{\mathsf{x}}, z^{\mathsf{y}})$, and the next iterate by $w = (w^{\mathsf{x}}, w^{\mathsf{y}})$.

Sampling. Recall that

$$p_{ij}(z) := [z^{\mathsf{y}}]_i \frac{A_{ij}^2}{\|A_{i:}\|_2^2}.$$

We first sample coordinate i via $\texttt{IM}_1^{\mathsf{y}}.\texttt{Sample}()$ in $O(\log m)$. Next, we sample $j \in [n]$ with probability proportional to A_{ij}^2 using the data structure corresponding to $A_{i:}$ in $O(1)$ by assumption of the matrix access model.

Computing the gradient estimator. To compute $c := \text{clip}(A_{ij}[z^{\mathsf{y}}]_i/p_{ij})$, it suffices to compute A_{ij}, $[z^{\mathsf{y}}]_i$, and p_{ij}. Using an entry oracle for A we obtain A_{ij}, and we get $[z^{\mathsf{y}}]_i$ by calling $\texttt{IM}_1^{\mathsf{y}}.\texttt{Get}(i)$. Computing p_{ij} using the precomputed $\|A_{i:}\|_2$ and the values of $A_{ij}, [z^{\mathsf{y}}]_i$ therefore takes $O(1)$ time.

Performing the update. For the update corresponding to a proximal step, we have

$$w^{\mathsf{x}} \leftarrow \Pi_{\mathcal{X}} \left(z^{\mathsf{x}} \circ \exp(-\eta \tilde{g}^{\mathsf{x}}(z)) \right) = \frac{z^{\mathsf{x}} \circ \exp(-\eta \tilde{g}^{\mathsf{x}}(z))}{\|z^{\mathsf{x}} \circ \exp(-\eta \tilde{g}^{\mathsf{x}}(z))\|_1}.$$

We have computed $\tilde{g}^{\mathsf{x}}(z)$, so to perform this update, we call

$$\begin{aligned}
&\xi \leftarrow \texttt{IM}_1^{\mathsf{x}}.\texttt{Get}(j);\\
&\texttt{IM}_1^{\mathsf{x}}.\texttt{AddSparse}(j, (\exp(-\eta c) - 1)\xi);\\
&\texttt{IM}_1^{\mathsf{x}}.\texttt{Scale}(\texttt{IterateMaintainer}^{\mathsf{x}}.\texttt{GetNorm}()^{-1});\\
&\texttt{IM}_1^{\mathsf{x}}.\texttt{UpdateSum}().
\end{aligned}$$

By assumption, each operation takes time $O(\log n)$, giving the desired iteration complexity. It is clear that at the end of performing these operations, the invariant that $\texttt{IM}_1^{\mathsf{x}}$ maintains the x block of the iterate is preserved.

Averaging. After T iterations, we compute the average point $\bar{z}^{\mathsf{x}}$:

$$[\bar{z}^{\mathsf{x}}]_j \leftarrow \frac{1}{T} \cdot \texttt{IM}_1^{\mathsf{x}}.\texttt{GetSum}(j), \forall j \in [n].$$

By assumption, this takes $O(n)$ time.

Algorithm guarantee

Theorem 10. *In the ℓ_1-ℓ_1 setup, the implementation in Section 3.4.1 has runtime*

$$O\left(\frac{\left(L_{\mathsf{co}}^{1,1}\right)^2 \log^2(mn)}{\epsilon^2} + m + n\right),$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that $\mathbb{E}\mathrm{Gap}(\bar{z}) \leq \epsilon$..

Proof. The runtime follows from the discussion in Section 3.4.1. The correctness follows from Proposition 9. □

Remark 4. *Using our* `IterateMaintainer`$_1$ *data structure, the ℓ_1-ℓ_1 algorithm of [253] runs in time $O(\mathsf{rcs}\,\|A\|_{\max}^2 \log^2(mn)/\epsilon^2)$, where rcs is the maximum number of nonzeros in any row or column. Our runtime universally improves upon it since $(L_{1,1}^{\mathrm{co}})^2 \leq \mathsf{rcs}\|A\|_{\max}^2$.*

3.4.2 ℓ_1-ℓ_1 variance-reduced coordinate method

Gradient estimator

Given reference point $w_0 \in \Delta^n \times \Delta^m$, for $z \in \Delta^n \times \Delta^m$ and a parameter $\alpha > 0$, we specify the sampling distributions $p(z; w_0), q(z; w_0)$:

$$p_{ij}(z; w_0) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2} \quad \text{and} \quad q_{ij}(z; w_0) \quad := \frac{[z^{\mathsf{x}}]_j + 2[w_0^{\mathsf{x}}]_j}{3} \cdot \frac{A_{ij}^2}{\|A_{:j}\|_2^2}. \tag{3.31}$$

We remark that this choice of sampling distribution, which we term "sampling from the sum" (of the current iterate and reference point), may be viewed as a computationally-efficient alternative to the distribution specified in [111], which was based on "sampling from the difference". In particular, sampling from the difference is an operation which to the best of our knowledge is difficult to implement in sublinear time, so we believe that demonstrating that this alternative distribution suffices may be of independent interest. In order to show its correctness, we need the following claim, whose proof we defer to Appendix B.4.1.

Lemma 32. *For $y, y' \in \Delta^m$, divergence $V_y(y')$ generated by $r(y) = \sum_{i\in[m]} [y]_i \log [y]_i - [y]_i$ satisfies*

$$V_y(y') \geq \frac{1}{2}\|y' - y\|^2_{\frac{3}{2y+y'}} = \frac{1}{2}\sum_{i\in[m]} \frac{([y]_i - [y']_i)^2}{\frac{2}{3}[y]_i + \frac{1}{3}[y']_i}.$$

We now show the local properties of this estimator.

Lemma 33. *In the ℓ_1-ℓ_1 setup, estimator* (3.28) *using the sampling distribution in* (3.31) *is a $\sqrt{2}L_{\mathsf{co}}^{1,1}$-centered-local estimator.*

Proof. Unbiasedness holds by definition. For arbitrary w^{x}, we have the variance bound:

$$\begin{aligned}
\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{x}}_{w_0}(z) - g^{\mathsf{x}}(w_0)\right\|^2_{w^{\mathsf{x}}}\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z;w_0)\cdot\left([w^{\mathsf{x}}]_j\cdot\left(\frac{A_{ij}\left([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i\right)}{p_{ij}(z;w_0)}\right)^2\right)\\
&= \sum_{i\in[m],j\in[n]} [w^{\mathsf{x}}]_j \frac{A_{ij}^2\left([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i\right)^2}{p_{ij}(z;w_0)}\\
&\le \sum_{i\in[m],j\in[n]} [w^{\mathsf{x}}]_j \frac{([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i)^2}{\frac{1}{3}[z^{\mathsf{y}}]_i + \frac{2}{3}[w_0^{\mathsf{y}}]_i}\left\|A_{i:}\right\|_2^2\\
&\le 2\left(\max_i \left\|A_{i:}\right\|_2^2\right) V_{w_0^{\mathsf{y}}}(z^{\mathsf{y}}),
\end{aligned}$$

where in the last inequality we used Lemma 32. Similarly, we have for arbitrary w^{y},

$$\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{y}}_{w_0}(z) - g^{\mathsf{y}}(w_0)\right\|^2_{w^{\mathsf{y}}}\right] \le 2\left(\max_j \left\|A_{:j}\right\|_2^2\right) V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}}).$$

Combining these and using

$$\left\|\tilde{g}_{w_0}(z) - g(w_0)\right\|^2_w := \left\|\tilde{g}^{\mathsf{x}}_{w_0}(z) - g^{\mathsf{x}}(w_0)\right\|^2_{w^{\mathsf{x}}} + \left\|\tilde{g}^{\mathsf{y}}_{w_0}(z) - g^{\mathsf{y}}(w_0)\right\|^2_{w^{\mathsf{y}}}$$

yields the desired variance bound. □

Implementation details

In this section, we discuss the details of how to leverage the `ApproxExpMaintainer` data structure to implement the iterations of our algorithm. We first state one technical lemma on the effect of β-padding (Definition 3) on increasing entropy, used in conjunction with the requirements of Proposition 11 to bound the error tolerance required by our `ApproxExpMaintainer` data structure. The proof is deferred to Appendix B.4.1.

Lemma 34. *Let $x' \in \Delta^n$ be a β-padding of $x \in \Delta^n$. Then,*

$$\sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} x_j \log x_j \le \frac{\beta n}{e} + \beta(1+\beta).$$

This leads to the following divergence bounds which will be used in this section.

Lemma 35. *Let $x' \in \Delta^n$ be a β-padding of $x \in \Delta^n$. Then*

$$V_{x'}(u) - V_x(u) \le \beta,\ \forall u \in \mathcal{Z}$$

and if $\|\log(x_0)\|_\infty \le M$*, then*

$$V_{x_0}(x') - V_{x_0}(x) \le \beta\left(2M + \frac{n}{e} + 1 + \beta\right)$$

Proof. Throughout this proof, let $\tilde{x}$ be the point in Definition 3 such that $\|\tilde{x} - x\|_1 \le \beta$ and $x' = \tilde{x}/\|\tilde{x}\|_1$.

The first claim follows from expanding

$$V_{x'}(u) - V_x(u) = \sum_{j\in[n]} u_j \log \frac{x_j}{x'_j} = \sum_{j\in[n]} u_j \log\left(\frac{x_j}{\tilde{x}_j} \cdot \|\tilde{x}\|_1\right) \le \log(\|\tilde{x}\|_1) \le \beta.$$

The first inequality used $u \in \Delta^n$ and $\tilde{x} \ge x$ entrywise, and the last inequality used $\log(1+\beta) \le \beta$.

For the second claim, we have by the triangle inequality

$$\|x - x'\|_1 \le \|x - \tilde{x}\|_1 + \|\tilde{x} - x'\|_1 \le \beta + (\|\tilde{x}\|_1 - 1)\|x'\|_1 \le 2\beta.$$

The claim then follows from expanding

$$V_{x_0}(x') - V_{x_0}(x) = \sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} x_j \log x_j + \langle \log x_0, x - x'\rangle,$$

and applying Lemma 34. □

The algorithm we analyze is Algorithm 11 with $K = 3\alpha\Theta/\epsilon$, $\varepsilon_{\text{outer}} = 2\epsilon/3$, $\varepsilon_{\text{inner}} = \epsilon/3$ using Algorithm 12 as an $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle. The specific modification we perform to define the iterates $\{z_k\}$ of Algorithm 11 as modifications of the ideal iterates $\{z_k^\star\}$ uses the following definition.

Definition 7. *For a simplex variable $x' \in \Delta^n$, we define* $\mathsf{truncate}(x', \delta)$ *to be the point $x \in \Delta^n$ with $x_j \propto \max(x'_j, \delta)$ for all $j \in [n]$. For a variable z on two blocks, we overload notation and define* $\mathsf{truncate}(z, \delta)$ *to be the result of applying* $\mathsf{truncate}(\cdot, \delta)$ *to each simplex block of z.*

For our implementation, in each step of Algorithm 11, we will compute the point $z_k^\star$ exactly, and apply the operation $z_k \leftarrow \mathsf{truncate}(z_k^\star, \delta)$, for $\delta = \frac{\varepsilon_{\text{outer}} - \varepsilon_{\text{inner}}}{\alpha(m+n)}$. We now quantify the effect of truncation in terms of Bregman divergence to an arbitrary point.

Lemma 36 (Effect of truncation)**.** *Let $x' \in \Delta^n$, and let $x = \mathsf{truncate}(x', \delta)$. Then, for any $u \in \Delta^n$, and where divergences are with respect to entropy,*

$$V_x(u) - V_{x'}(u) \le \delta n.$$

Proof. Note that x is a δn-padding of x', as it is the result of adding at most δ to each coordinate and renormalizing to lie in the simplex. Consequently, the result follows from Lemma 35. □

Lemma 255 thus implies our iterates satisfy the requirements of Algorithm 11. Our implementation of Algorithm 12 will use approximation tolerance $\varphi = \varepsilon_{\text{inner}}/6 = \epsilon/18$, where we always set

$$L_{\mathsf{co}}^{1,1} \geq \alpha \geq \varepsilon_{\text{inner}}. \tag{3.32}$$

This matches the requirements of Proposition 11. In the implementation of Algorithm 12, we use the centered-local gradient estimator defined in (3.28), using the sampling distribution (3.31). For each use of Algorithm 12, we choose

$$\eta = \frac{\alpha}{20\left(L_{\mathsf{co}}^{1,1}\right)^2} \text{ and } T = \left\lceil \frac{6}{\eta\alpha} \right\rceil \geq \frac{120\left(L_{\mathsf{co}}^{1,1}\right)^2}{\alpha^2}.$$

Our discussion will follow in four steps: first, we discuss the complexity of all executions in Algorithm 11 other than calls to the oracles. Next, we discuss the complexity of all initializations of `ApproxExpMaintainer` data structures. Then, we discuss the complexity of all other iterations of Algorithm 12. For simplicity, when discussing Algorithm 12, we will only discuss implementation of the x-block, and the y-block will follow symmetrically, while most runtimes are given considering both blocks. Lastly, we discuss complexity of computing the average iterate in the end of the inner loop. Altogether, the guarantees of Proposition 10 and Proposition 11 imply that if the guarantees required by the algorithm hold, the expected gap of the output is bounded by ϵ.

Outer loop extragradient steps. Overall, we execute $K = 3\alpha\Theta/\epsilon$ iterations of Algorithm 11, with $\varepsilon_{\text{outer}} = 2\epsilon/3$, $\varepsilon_{\text{inner}} = \epsilon/3$ to obtain the desired gap, where $\Theta = \log(mn)$ in the ℓ_1-ℓ_1 setup. We spend $O(\mathsf{nnz})$ time executing each extragradient step in Algorithm 11 exactly to compute iterates $z_k^\star$, where the dominant term in the runtime is computing each $g(z_{k-1/2})$, for $k \in [K]$. We can maintain the average point $\bar{z}$ throughout the duration of the algorithm, in $O(m+n)$ time per iteration. Finally, we spend an additional $O(m+n)$ time per iteration applying `truncate` to each iterate $z_k^\star$.

Data structure initializations and invariants.

We consider the initialization of data structures for implementing an $(\alpha, \varepsilon_{\text{inner}} = \epsilon/3)$-relaxed proximal oracle with error tolerance $\varphi = \epsilon/18$. First, note that the point w_0^{x} used in the initialization of every inner loop, by the guarantees of `truncate` operation, has no two coordinates with multiplicative ratio larger than $\delta = \epsilon/(3\alpha(m+n)) \geq (m+n)^{-4}$, by our choice $\alpha \leq L_{\mathsf{co}}^{1,1}$ (3.32) and our assumptions on $L_{\mathsf{co}}^{1,1}/\epsilon$ (cf. Section 3.2.2). Since clearly a simplex variable in Δ^n has a coordinate at least $1/n$, the entries of w_0 are lower bounded by $\lambda = (m+n)^{-5}$.

Next, we discuss the initial parameters given to $\texttt{AEM}^{\mathsf{x}}$, an instance of `ApproxExpMaintainer` which will support necessary operations for maintaining the x variable (we will similarly initialize an instance $\texttt{AEM}^{\mathsf{y}}$). Specifically, the invariant that we maintain throughout the inner loop is that in iteration t, the"exact vector" x maintained by $\texttt{AEM}^{\mathsf{x}}$ corresponds to the x block of the current iterate w_t, and the "approximate vector" $\hat{x}$ maintained corresponds to the x block of the approximate

iterate $\hat{w}_t$, as defined in Algorithm 12.

We will now choose $\tilde{\varepsilon}$ so that if $\texttt{AEM}^{\times}$ is initialized with error tolerance $\tilde{\varepsilon}$, all requirements of Proposition 11 (e.g. the bounds stipulated in Algorithm 12) are met. We first handle all divergence requirements. In a given iteration, denote the x blocks of $w_t^\star$, w_t and $\hat{w}_t$ by $x_t^\star$, x_t and $\hat{x}_t$ respectively, and recall $\texttt{AEM}^{\times}$ guarantees x_t is a $\tilde{\varepsilon}$-padding of $x_t^\star$, and $\hat{x}_t$ is a $\tilde{\varepsilon}$-padding of $x_t^\star$. The former of these guarantees is true by the specification of `MultSparse` (which will be used in the implementation of the step, see "Performing the update" below), and the latter is true by the invariant on the points supported by $\texttt{AEM}^{\times}$. Lines 5 and 7 of Algorithm 12 stipulate the divergence requirements, where $x_0 := w_0^{\times}$,

$$\max\left\{V_{x_0}(x_t) - V_{x_0}\left(x_t^\star\right), V_{x_0}\left(\hat{x}_t\right) - V_{x_0}\left(x_t\right)\right\} \leq \frac{\varphi}{2\alpha} = \frac{\epsilon}{36\alpha} \tag{3.33}$$

and

$$\max_u \left[V_{x_t}(u) - V_{x_t^\star}(u)\right] \leq \frac{\eta\varphi}{2} = \frac{\eta\epsilon}{36}. \tag{3.34}$$

Clearly, combining this guarantee with a similar guarantee on the y blocks yields the desired bound. Since we derived $\|\log w_0\|_\infty \leq 5\log(mn)$, we claim that choosing

$$\tilde{\varepsilon} \leq \frac{\epsilon}{36\alpha(m+n)}$$

suffices for the guarantees in (3.33). By the first part of Lemma 35, for all sufficiently large $m+n$,

$$\max\left\{V_{x_0}(x_t) - V_{x_0}\left(x_t^\star\right), V_{x_0}\left(\hat{x}_t\right) - V_{x_0}\left(x_t\right)\right\} \leq \tilde{\varepsilon}\left(10\log(mn) + \frac{n}{e} + 1 + \tilde{\varepsilon}\right) \leq \frac{\epsilon}{36\alpha}.$$

Similarly for guarantees in (3.34), by the second part of Lemma 35 we know it suffices to choose

$$\tilde{\varepsilon} \leq \frac{\epsilon^2}{720\left(L_{\mathsf{co}}^{1,1}\right)^2} \leq \frac{\epsilon\alpha}{720\left(L_{\mathsf{co}}^{1,1}\right)^2} = \frac{\eta\epsilon}{36}.$$

Here, we used the restriction $\varepsilon_{\text{inner}} \leq \alpha \leq L_{\mathsf{co}}^{1,1}$. Next, the norm requirements of Algorithm 12 (the guarantees in Lines 5, 7, and 8) imply we require

$$\tilde{\varepsilon} \leq \frac{\epsilon}{18\sqrt{2}L_{\mathsf{co}}^{1,1}},$$

where we used that g is $\|A\|_{\max} \leq L_{\mathsf{co}}^{1,1}$-Lipschitz and the diameter of $\mathcal{Z}$ is bounded by $\sqrt{2}$. Using our assumptions on the size of parameters in Section 3.2.2, it suffices to set the error tolerance

$$\tilde{\varepsilon} = (m+n)^{-8}.$$

To give the remainder of specified parameters, $\texttt{AEM}^{\mathsf{x}}$ is initialized via $\texttt{Init}(w_0^{\mathsf{x}}, v, \kappa, \tilde{\varepsilon})$ for

$$\kappa := \frac{1}{1+\eta\alpha/2},\ v := (1-\kappa)\log w_0^{\mathsf{x}} - \eta\kappa g^{\mathsf{x}}(w_0).$$

To motivate this form of updates, note that each iteration of Algorithm 12 requires us to compute

$$\operatorname{argmin}\left\{\langle c_t e_j + g^{\mathsf{x}}(w_0), x\rangle + \frac{\alpha}{2}V_{w_0^{\mathsf{x}}}(x) + \frac{1}{\eta}V_{w_t^{\mathsf{x}}}(x)\right\}.$$

We can see that the solution to this update is given by

$$\left[w_{t+1}^{\star}\right]^{\mathsf{x}} \leftarrow \Pi_{\Delta}\left([w_t^{\mathsf{x}}]^{\kappa} \circ \exp\left((1-\kappa)\log w_0^{\mathsf{x}} - \eta\kappa g^{\mathsf{x}}(w_0)\right) \circ \exp(-\eta\kappa c_t e_j)\right). \tag{3.35}$$

This form of update is precisely supported by our choice of κ and v, as well as the `DenseStep` and `MultSparse` operations. By the choice of parameters, we note that $1-\kappa \geq (m+n)^{-8}$.

Finally, in order to support our sampling distributions and gradient computations, we compute and store the vectors w_0 and $g(w_0)$ in full using $O(\mathsf{nnz}(A))$ time at the beginning of the inner loop. In $O(m+n)$ time, we also build two data structures which allow us to sample from entries of the given fixed vectors w_0^{x}, and w_0^{y}, in constant time respectively.

Following Section 3.2.4, we defined the parameter

$$\omega := \max\left(\frac{1}{1-\kappa},\ \frac{n}{\lambda\tilde{\epsilon}}\right) \leq (m+n)^{13},\ \text{so that}\ \log(\omega) = O(\log(mn)).$$

Altogether, these initializations take time $O(\mathsf{nnz} + (m+n)\log^3(mn))$, following Section 3.2.4.

Inner loop iterations. We discuss how to make appropriate modifications to the x-block. For simplicity we denote our current iterate as z, and the next iterate as w. Also, we denote $\hat{z}$ as the concatenation of implicit iterates that the two `ApproxExpMaintainer` copies maintain (see Section 3.2.4 for more details), which is $\tilde{\varepsilon}$ close in ℓ_1 distance to z, the prior iterate, so that we can query or sample entries from $\hat{z}$ using $\texttt{AEM}^{\mathsf{x}}$ and $\texttt{AEM}^{\mathsf{y}}$. Each inner loop iteration consists of using a gradient estimator at $\hat{z}$ satisfying $\|\hat{z}-z\|_1 \leq \tilde{\varepsilon}$, sampling indices for the computation of $\tilde{g}_{w_0}(\hat{z})$, computing the sparse part of $\tilde{g}_{w_0}(\hat{z})$, and performing the approximate update to the iterate. We show that we can run each substep using data structure $\texttt{AEM}^{\mathsf{x}}$ in time $O(\log^4(mn))$, within the error tolerance of Proposition 11 due to the definition of $\tilde{\varepsilon}$. Combining with our discussion of the complexity of initialization, this implies that the total complexity of the inner loop, other than outputting the average iterate, is

$$O(T\log^4(mn) + \mathsf{nnz} + (m+n)\log^3(mn)) = O\left(\frac{\left(L_{\mathsf{co}}^{1,1}\right)^2 \cdot \log^4(mn)}{\alpha^2} + \mathsf{nnz} + (m+n)\log^3(mn)\right).$$

Sampling. Recall that the distribution we sample from is given by

$$p_{ij}(\hat{z}; w_0) := \frac{[\hat{z}^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}.$$

First, with probability $2/3$, we sample a coordinate i from the precomputed data structure for sampling from w_0^{y} in constant time; otherwise, we sample i via $\texttt{AEM}^{\mathsf{y}}.\texttt{Sample}()$. Then, we sample an entry of $A_{i:}$ proportional to its square via the precomputed data structure (cf. Section 3.2.3) in constant time. This takes in total $O(\log mn)$ time.

Computing the gradient estimator. Proposition 11 requires us to compute the sparse component of the gradient estimator (3.28) at point $\hat{z}$. To do this for the x block, we first query $[w_0^{\mathsf{x}}]_j$ and $[\hat{z}^{\mathsf{y}}]_i \leftarrow \texttt{AEM}^{\mathsf{y}}.\texttt{Get}(i)$, and then access the precomputed norm $\|A_{i:}\|_2$ and entry A_{ij}. We then compute

$$c = \text{clip}\left(A_{ij}\left[\hat{z}^{\mathsf{y}} - w_0^{\mathsf{y}}\right]_i \cdot \frac{3}{[\hat{z}^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i} \cdot \frac{\|A_{i:}\|_2^2}{A_{ij}^2}\right).$$

By the guarantees of `ApproxExpMaintainer`, this takes total time bounded by $O(\log(mn))$.

Performing the update. To perform the update, by observing the form of steps in Algorithm 12 with our choice of entropy regularizer, the update form given by the regularized mirror-descent step is (as derived in the discussion of the initialization of $\texttt{AEM}^{\mathsf{x}}$, see (3.35))

$$[w^{\star\mathsf{x}} \leftarrow \Pi_\Delta((w^{\mathsf{x}})^\kappa \circ \exp((1-\kappa)\log w_0^{\mathsf{x}} - \eta\kappa g^{\mathsf{x}}(w_0) - \eta\kappa c e_j)).$$

To implement this, recalling our choice of the vector v in the initialization of $\texttt{AEM}^{\mathsf{x}}$, it suffices to call

$$\begin{aligned}
&\texttt{AEM}^{\mathsf{x}}.\texttt{DenseStep}();\\
&\texttt{AEM}^{\mathsf{x}}.\texttt{MultSparse}(-\eta\kappa c e_j);\\
&\texttt{AEM}^{\mathsf{x}}.\texttt{UpdateSum}().
\end{aligned}$$

By assumption, each operation takes time bounded by $O(\log^4(mn))$, where we note the vector used in the `MultSparse` operation is 1-sparse. The implementation of this update is correct up to a $\tilde{\varepsilon}$-padding, whose error we handled previously. By the discussion in the data structure initialization section, this preserves the invariant that the x block of the current iterate is maintained by $\texttt{AEM}^{\mathsf{x}}$.

Average iterate computation. At the end of each run of Algorithm 12, we compute and return the average iterate via calls $\texttt{AEM}^{\mathsf{x}}.\texttt{GetSum}(j)$ for each $j \in [n]$, and scaling by $1/T$, and similarly query $\texttt{AEM}^{\mathsf{y}}$. The overall complexity of this step is $O((m+n)\log^2(mn))$. The correctness guarantee, i.e. that the output approximates the average in ℓ_1 norm up to φ/LD, is given by the choice of $\tilde{\varepsilon}$ and

the guarantees of `ApproxExpMaintainer`, where in this case the domain size D is bounded by $\sqrt{2}$. This is never the dominant factor in the runtime, as it is dominated by the cost of initializations.

Algorithm guarantee

Theorem 11. *In the ℓ_1-ℓ_1 setup, let $\mathsf{nnz}' := \mathsf{nnz} + (m+n)\log^3(mn)$. The implementation in Section 3.4.2 with the optimal choice of $\alpha = \max\left(\epsilon/3, L_{\mathsf{co}}^{1,1}\log^2(mn)/\sqrt{\mathsf{nnz}'}\right)$ has runtime*

$$O\left(\left(\mathsf{nnz}' + \frac{\left(L_{\mathsf{co}}^{1,1}\right)^2\log^4(mn)}{\alpha^2}\right)\frac{\alpha\log(mn)}{\epsilon}\right) = O\left(\mathsf{nnz}' + \frac{\sqrt{\mathsf{nnz}'}L_{\mathsf{co}}^{1,1}\log^3(mn)}{\epsilon}\right)$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \le \epsilon.$$

Proof. The correctness of the algorithm is given by the discussion in Section 3.4.2 and the guarantees of Proposition 10 with $K = 3\alpha\Theta/\epsilon$, $\varepsilon_{\text{outer}} = 2\epsilon/3$, $\varepsilon_{\text{inner}} = \epsilon/3$, Proposition 11 with $\varphi = \epsilon/15$, and the data structure `ApproxExpMaintainer` with our choice of

$$\tilde{\varepsilon} := (m+n)^{-8},$$

to meet the approximation conditions in Line 5, 7 and 8 of Algorithm 12. The runtime bound is given by the discussion in Section 3.4.2, and the optimal choice of α is clear. □

3.5 Data structure implementation

In this section, we give implementations of our data structures, fulfilling the interface and runtime guarantees of Section 3.2.4. In Section 3.5.1 we provide the implementation of `IterateMaintainer`$_p$ for $p \in \{1,2\}$ used for sublinear coordinate methods. In Section 3.5.2, we provide an implementation of `ApproxExpMaintainer` used in variance-reduced coordinate methods for simplex domains, provided we have an implementation of a simpler data structure, `ScaleMaintainer`, which we then provide in Section 3.5.3.

3.5.1 `IterateMaintainer`$_p$

The `IterateMaintainer`$_p$, $p \in \{1,2\}$ data structure is described in Section 3.2.4 and used for tracking the iterates in our fully stochastic methods and the Euclidean part of our the iterates in our variance-reduced methods. The data structure maintains an internal representation of x, the current iterate, and s, a running sum of all iterates. The main idea behind the efficient implementation of the data structure is to maintain x and s as a linear combination of sparsely-updated vectors. In particular, the data structure has the following state: scalars ξ_u, ξ_v, σ_u, σ_v, ι, ν; vectors u, u', v,

and the scalar $\|v\|_2^2$; the vector v is only relevant for variance reduction and is therefore set 0 for the non-Euclidean case $p = 1$.

We maintain the following invariants on the data structure state at the end of every operation:

- $x = \xi_u u + \xi_v v$, the internal representation of x
- $s = u' + \sigma_u u + \sigma_v v$, the internal representation of running sum s
- $\iota = \langle x, v \rangle$, the inner product of the iterate with fixed vector v
- $\nu = \|x\|_p$, the appropriate norm of the iterate

In addition, to support sampling, our data structure also maintains a binary tree dist_x of depth $O(\log n)$. Each leaf node is associated with a coordinate $j \in [n]$, and each internal node is associated with a subset of coordinates corresponding to leaves in its subtree. For the node corresponding to $S \subseteq [n]$ (where S may be a singleton), we maintain the sums $\sum_{j \in S} [u]_j^p$, $\sum_{j \in S} [u]_j [v]_j$, and $\sum_{j \in S} [v]_j^p$.

We now give the implementation of each operation supported by $\texttt{IterateMaintainer}_d$, followed by proofs of correctness and of the runtime bounds when applicable.

Initialization

- $\texttt{Init}(x_0, v)$. Runs in time $O(n)$.

  If $p = 1$ set $v \leftarrow \mathbf{0}_n$; otherwise we compute and store $\|v\|_2^2$. Initialize the remaining data structure state as follows: $(\xi_u, \xi_v, u) \leftarrow (1, 0, x_0)$, $(\sigma_u, \sigma_v, u') \leftarrow (0, 0, \mathbf{0}_n)$, $(\iota, \nu) \leftarrow (\langle x_0, v \rangle, \|x_0\|_p)$. Initialize dist_x, storing the relevant sums in each internal node.

It is clear that $x = \xi_u u + \xi_v v$, $s = u' + \sigma_u u + \sigma_v v$, and that the invariants of ι, ν hold. Each step takes $O(n)$ time; for the first 4 steps this is immediate, and the final recursing upwards from the leaves spends constant time for each internal node, where there are $O(n)$ nodes.

Updates

- $\texttt{Scale}(c)$: $x \leftarrow cx$. Runs in time $O(1)$.

  Multiply each of ξ_u, ξ_v, ν, ι by c.

- $\texttt{AddSparse}(j, c)$: $[x]_j \leftarrow [x]_j + c$, with the guarantee $c \geq -[x]_j$ if $p = 1$. Runs in time $O(\log n)$.

  1. $u \leftarrow u + \frac{c}{\xi_u} e_j$.
  2. $u' \leftarrow u' - \frac{c\sigma_u}{\xi_u} e_j$.
  3. If $p = 1$, $\nu \leftarrow \nu + c$. If $p = 2$, $\nu \leftarrow \sqrt{\nu^2 + 2c[\xi_u u + \xi_v v]_j + c^2}$.
  4. $\iota \leftarrow \iota + c[v]_j$.

5. For internal nodes of dist_x on the path from leaf j to the root, update $\sum_{j\in S}[u]_j^p$, $\sum_{j\in S}[u]_j[v]_j$ appropriately.

- `AddDense`(c): $x \leftarrow x + cv$. Runs in time $O(1)$. (Supported only for $p = 2$).

  Set $\xi_v \leftarrow \xi_v + c$, $\nu \leftarrow \sqrt{\nu^2 + 2c\iota + c^2 \|v\|_2^2}$, and $\iota \leftarrow \iota + c\|v\|_2^2$.

- `UpdateSum`(): $s \leftarrow s + x$. Runs in time $O(1)$.

  Set $\sigma_u \leftarrow \sigma_u + \xi_u$ and $\sigma_v \leftarrow \sigma_v + \xi_v$.

Each of the runtime bounds clearly hold; we now demonstrate that the necessary invariants are preserved. Correctness of `Scale` and `UpdateSum` are clear. Regarding correctness of `AddSparse`, note that (ignoring the v terms when $p = 1$)

$$\xi_u\left(u + \frac{c}{\xi_u}e_j\right) + \xi_v v = \xi_u u + \xi_v v + ce_j,$$
$$\left(u' - \frac{c\sigma_u}{\xi_u}e_j\right) + \sigma_u\left(u + \frac{c}{\xi_u}e_j\right) + \sigma_v v = u' + \sigma_u u + \sigma_v v.$$

When $p = 1$, the update to ν is clearly correct. When $p = 2$, because only $[x]_j$ changes,

$$[\xi_u u + \xi_v v + ce_j]_j^2 = [\xi_u u + \xi_v v]_j^2 + 2c[\xi_u u + \xi_v v]_j + c^2,$$
$$([\xi_u u + \xi_v v + ce_j]_j) \cdot [v]_j = ([\xi_u u + \xi_v v]_j) \cdot [v]_j + c[v]_j.$$

Thus, the updates to the norm and inner product are correct. Regarding correctness of `AddDense` when $p = 2$, we have

$$\xi_u u + (\xi_v + c)v = \xi_u u + \xi_v v + cv,$$
$$\|x + cv\|_2^2 = \nu^2 + 2c\iota + c^2\|v\|_2^2,$$
$$\langle x + cv, v\rangle = \iota + c\|v\|_2^2.$$

Here, we use the invariants that $\nu = \|x\|_2$ and $\iota = \langle x, v\rangle$.

Queries

- `Get`(j): Return $[x]_j$. Runs in time $O(1)$.

  Return $\xi_u[u]_j + \xi_v[v]_j$.

- `GetSum`(j): Return $[s]_j$. Runs in time $O(1)$.

  Return $[u']_j + \sigma_u[u]_j + \sigma_v[v]_j$.

- `Norm`(): Return $\|x\|_p$. Runs in time $O(1)$.

Return ν.

By our invariants, each of these operations is correct.

Sampling

The method `Sample` returns a coordinate j with probability proportional to $[x]_j^p$ in time $O(\log n)$. To implement it, we recursively perform the following procedure, where the recursion depth is at most $O(\log n)$, starting at the root node and setting $S = [n]$:

1. Let S_1, S_2 be the subsets of coordinates corresponding to the children of the current node.
2. Using scalars ξ_u, ξ_v, and the maintained $\sum_{j\in S_i}[u]_j^p$, $\sum_{j\in S_i}[u]_j[v]_j$, $\sum_{j\in S_i}[v]_j^p$ when appropriate, compute $\sum_{j\in S_i}[x]_j^p = \sum_{j\in S_i}[\xi_u u + \xi_v v]_j^p$ for $i \in \{1, 2\}$.
3. Sample a child $i \in \{1, 2\}$ of the current node proportional to $\sum_{j\in S_i}[x]_j^p$ by flipping an appropriately biased coin. Set $S \leftarrow S_i$.

It is clear that this procedure samples according to the correct probabilities. Furthermore, step 2 can be implemented in $O(1)$ time using precomputed values, so the overall complexity is $O(\log n)$.

3.5.2 `ApproxExpMaintainer`

In this section, we give the implementation of `ApproxExpMaintainer` which supports dense update to simplex mirror descent iterates. For convenience, we restate its interface, where we recall the notation $\|g\|_0$ for the number of nonzero entries in g, Definition 3 of an ε-padding, the invariant

$$\hat{x} \text{ is a } \varepsilon\text{-padding of } x, \tag{3.36}$$

and the notation

$$\omega := \max\left(\frac{1}{1-\kappa},\ \frac{n}{\lambda\epsilon}\right).$$

Category	Function	Runtime
initialize	`Init`$(x_0, v, \kappa, \varepsilon, \lambda) : \kappa \in [0, 1)$, $\varepsilon > 0$, $\min_j [x_0]_j \geq \lambda$	$O(n \log n \log^2 \omega)$
update	`MultSparse`(g): $x \leftarrow \varepsilon$-padding of $\Pi_\Delta(x \circ \exp(g))$	$O(\|g\|_0 \log^2 n \log^2 \omega)$
	`DenseStep`(): $x \leftarrow \Pi_\Delta(x^\kappa \circ \exp(v))$	$O(\log n)$
	`UpdateSum`(): $s \leftarrow s + \hat{x}$ (recall invariant (3.25))	$O(\log n \log \omega)$
query	`Get`(j): Return $[\hat{x}]_j$	$O(\log n \log \omega)$
	`GetSum`(j): Return $[s]_j$	$O(\log^2 \omega)$
sample	`Sample`(): Return j with probability $[\hat{x}]_j$	$O(\log n \log \omega)$

We build `ApproxExpMaintainer` out of a simpler data structure called `ScaleMaintainer`, which maintains the simplex projection of fixed vectors raised elementwise to arbitrary powers; this suffices to support consecutive `DenseStep` calls without `MultSparse` calls between them. To add support for `MultSparse`, we combine $O(\log n)$ instances of `ScaleMaintainer` in a formation resembling a binomial heap: for every entry updated by `MultSparse` we delete it from the `ScaleMaintainer` instance currently holding it, put it in a new singleton `ScaleMaintainer` instance (after appropriate scaling due to `MultSparse`), and merge this singleton into existing instances. We now give a brief description of the `ScaleMaintainer` interface, and based on it, describe the implementation of `ApproxExpMaintainer`. We will provide the implementation of `ScaleMaintainer` in Section 3.5.3.

`ScaleMaintainer` is initialized with vectors $\bar{x} \in \mathbb{R}^{n'}_{\geq 0}$ and $\bar{\delta} \in \mathbb{R}^{n'}$ (with $n' \leq n$) and supports efficient approximate queries on vectors of the form

$$x[\sigma] := \bar{x} \circ \exp\left(\sigma\bar{\delta}\right),$$

for any scalar $\sigma \in [\sigma_{\min}, 1]$. More specifically, the data structure allows efficient computation of $\|x[\sigma]\|_1$ (to within small multiplicative error ε_{scm}), as well as entry queries, sampling and running sum accumulation from a vector $\hat{x}[\sigma]$ satisfying

$$\hat{x}[\sigma] \text{ is a } \varepsilon_{\text{scm}}\text{-padding of } \Pi_\Delta\left(\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right) = \frac{x[\sigma]}{\|x[\sigma]\|_1}. \tag{3.37}$$

We make the following assumptions on the input to the data structure:

$$\lambda_{\text{scm}} \leq [\bar{x}]_i \leq 1 \text{ for all } i \in [n'] \quad \text{and} \quad \sigma \in (\sigma_{\min}, 1).$$

The upper bounds on $\bar{x}$ and σ are arbitrary, and we may choose λ_{scm} and $\sigma_{\min}$ to be very small since the data structure runtime depends on them only logarithmically. To summarize this dependence, we define

$$\omega_{\text{scm}} := \max\left\{\frac{1}{\sigma_{\min}}, \frac{n}{\lambda_{\text{scm}}\varepsilon_{\text{scm}}}\right\}.$$

With these assumptions and notation, we define the formal interface of `ScaleMaintainer`.

Category	**Function**	**Runtime**
initialize	`Init`$(\bar{x}, \bar{\delta}, \sigma_{\min}, \varepsilon_{\text{scm}}, \lambda_{\text{scm}})$	$O(n' \log n \log^2 \omega_{\text{scm}})$
update	`Del`(j): Remove coordinate j from $\bar{x}$, $\bar{\delta}$	$O(1)$
	`UpdateSum`(γ, σ): $s \leftarrow s + \gamma\hat{x}[\sigma]$, with $\hat{x}[\sigma]$ defined in (3.37)	$O(\log \omega_{\text{scm}})$
query	`Get`(j): Return $[\hat{x}[\sigma]]_j$	$O(\log \omega_{\text{scm}})$
	`GetSum`(j): Return $[s]_j$.	$O(\log^2 \omega_{\text{scm}})$
	`GetNorm`(σ): Return $1 \pm \varepsilon$ approx. of $\left\|\bar{x} \circ \exp(\sigma\bar{\delta})\right\|_1$	$O(\log \omega_{\text{scm}})$
sample	`Sample`(σ): Return j with probability $[\hat{x}[\sigma]]_j$	$O(\log n \log \omega_{\text{scm}})$

`ApproxExpMaintainer` **state**

Throughout this section, we denote $K := \lceil \log n \rceil$. `ApproxExpMaintainer` maintains a partition of $[n]$ into K sets $S_1 \ldots, S_K$ (some of them possibly empty) that satisfy the invariant

$$|S_k| \leq 2^k \text{ for all } k \in [K]. \tag{3.38}$$

We refer to the index k as "rank" and associate with each rank $k \in [K]$ the following data

1. Scalar $\gamma_k \geq 0$ and nonnegative integer τ_k.
2. Vectors $\bar{x}_k, \bar{\delta}_k \in \mathbb{R}^{|S_k|}$ such that $\lambda_{\text{scm}} \leq [\bar{x}_k]_i \leq 1$ for all $i \in [|S_k|]$, where $\lambda_{\text{scm}} = \min(\varepsilon/n, \lambda)$.
3. A `ScaleMaintainer` instance, denoted $\texttt{ScaleMaintainer}_k$, initialized with $\bar{x}_k, \bar{\delta}_k$ and λ_{scm} defined above, $\sigma_{\min} = 1 - \kappa$ and $\varepsilon_{\text{scm}} = \varepsilon/10$, so that $\log \omega_{\text{scm}} = O(\log \omega)$.

`ApproxExpMaintainer` also maintains a vector $u \in \mathbb{R}^n$ for auxiliary running sum storage.

Define the vector $\delta \in \mathbb{R}^n$ by

$$[\delta]_{S_k} = \log\left(\gamma_k \left[\bar{x}_k \circ \exp\left((1-\kappa^{\tau_k})\bar{\delta}_k\right)\right]\right), \ k \in \{1, \ldots, K\}, \tag{3.39}$$

where $[\delta]_{S_k}$ denotes the coordinates of δ in S_k. Recall that x denotes the point in Δ^n maintained throughout the operations of `ApproxExpMaintainer`; we maintain the key invariant that the point x is proportional to $\exp(\delta)$, i.e.,

$$x = \frac{\exp(\delta)}{\|\exp(\delta)\|_1}. \tag{3.40}$$

Specifically, we show in Section 3.5.2 that our implementation of `DenseStep` modifies $\bar{x}$, $\bar{\delta}$, $\{\tau_k\}_{k=0}^K$, $\{\gamma_k\}_{k=0}^K$ so that the resulting effect on δ, per definition (3.40), is

$$\delta \leftarrow \kappa\delta + v. \tag{3.41}$$

Similarly, our implementation of `MultSparse` modifies the state so that the resulting effect on δ is

$$\frac{\exp(\delta)}{\|\exp(\delta)\|_1} \leftarrow \varepsilon\text{-padding of } \frac{\exp(\delta + v)}{\|\exp(\delta + v)\|_1}. \tag{3.42}$$

We remark that the role of γ_k is to scale $\bar{x}_k$ so that it lies coordinatewise in the range $[\lambda_{\text{scm}}, 1]$, conforming to the `ScaleMaintainer` input requirement. This is also the reason we require the ε-padding operation in the definition of `MultSparse`.

ε-padding point $\hat{x}$

We now concretely define the point $\hat{x}$, which is the ε-padding of x that `ApproxExpMaintainer` maintains. Let

$$\Gamma := \sum_{k=1}^{K} \gamma_k \texttt{ScaleMaintainer}_k.\texttt{GetNorm}(1-\kappa^{\tau_k}), \tag{3.43}$$

be the approximation of $\exp(\delta)$ derived from the `ScaleMaintainer` instances. For any $j \in [n]$, let k_j be such that $j \in S_{k_j}$, and let i_j be the index of j in S_{k_j}. The jth coordinate of $\hat{x}$ is

$$[\hat{x}]_j := \frac{\gamma_{k_j} \texttt{ScaleMaintainer}_{k_j}.\texttt{GetNorm}(1-\kappa^{\tau_{k_j}})}{\Gamma} \cdot \texttt{ScaleMaintainer}_{k_j}.\texttt{Get}(i_j, 1-\kappa^{\tau_{k_j}}). \tag{3.44}$$

Since for each k, $\sum_{j\in S_k} \texttt{ScaleMaintainer}_k.\texttt{Get}(j, 1-\kappa^{\tau_k}) = \texttt{ScaleMaintainer}_k.\texttt{GetNorm}(1-\kappa^{\tau_k})$ we have that $\hat{x} \in \Delta^n$. We now prove that $\hat{x}$ is a ε-padding of x. To do so, we prove the following lemma.

Lemma 37. *Let $\varepsilon_{\text{scm}} \le \frac{1}{10}$ and $\{S_k\}_{k=1}^K$ be a partition of $[n]$. Suppose for each $k \in [K]$, $\hat{x}_k \in \Delta^{|S_k|}$ is an ε_{scm}-padding of $x_k \in \Delta^{|S_k|}$. Further, suppose we have positive scalars $\{\nu_k\}_{k=1}^K$, $\{\hat{\nu}_k\}_{k=1}^K$ satisfying*

$$(1-\varepsilon_{\text{scm}})\nu_k \le \hat{\nu}_k \le (1+\varepsilon_{\text{scm}})\nu_k, \text{ for all } 1 \le k \le K.$$

Then, for $N = \sum_{k=1}^K \nu_k$ and $\hat{N} = \sum_{k=1}^K \hat{\nu}_k$, we have that $\hat{x} := \sum_{k=1}^K \frac{\hat{\nu}_k}{\hat{N}} \hat{x}_k$ is a $10\varepsilon_{\text{scm}}$-padding of $x := \sum_{k=1}^K \frac{\nu_k}{N} x_k$.

Proof. For every $k \in [K]$, let $\tilde{x}_k$ to be such that $\tilde{x}_k \ge x_k$ elementwise, $\hat{x}_k = \tilde{x}_k / \|\tilde{x}_k\|_1$, and $\|\tilde{x}_k - x_k\|_1 \le \varepsilon_{\text{scm}}$. Consider the point

$$\tilde{x} := \sum_{k=1}^{K} \frac{\tilde{\nu}_k}{\hat{N}} \tilde{x}_k, \quad \text{where} \quad \tilde{\nu}_k := \hat{\nu}_k \frac{\max_{k\in[K]} \|\tilde{x}_k\|_1}{\|\tilde{x}_k\|_1} \cdot \frac{1+\varepsilon_{\text{scm}}}{1-\varepsilon_{\text{scm}}},$$

so that $\hat{x} = \tilde{x}/\|\tilde{x}\|_1$. Since $\tilde{x}_k \ge x_k$ elementwise, $\hat{\nu}_k \ge (1-\varepsilon_{\text{scm}})\nu_k$ and $\hat{N} \le (1+\varepsilon_{\text{scm}})N$, we have that $\tilde{x} \ge x$ elementwise. Furthermore, we have $\hat{\nu}_k \le (1+\varepsilon_{\text{scm}})\nu_k$ and $\hat{N} \ge (1-\varepsilon_{\text{scm}})N$, and the properties $\tilde{x}_k \ge x_k$ and $\|\tilde{x}_k - x_k\|_1$ imply $1 \le \|\tilde{x}_k\|_1 \le 1+\varepsilon_{\text{scm}}$ as well as $\frac{\max_{k\in[K]}\|\tilde{x}_k\|_1}{\|\tilde{x}_k\|_1} \le 1+\varepsilon_{\text{scm}}$. Therefore

$$\|\tilde{x} - x\|_1 \le \sum_{k=1}^{K} \frac{\nu_k}{N} \left\| \frac{(1+\varepsilon_{\text{scm}})^3}{(1-\varepsilon_{\text{scm}})^2} \tilde{x}_k - x_k \right\|_1 \le \left(\frac{(1+\varepsilon_{\text{scm}})^3}{(1-\varepsilon_{\text{scm}})^2} - 1 \right)(1+\varepsilon_{\text{scm}}) + \varepsilon_{\text{scm}} \le 10\varepsilon_{\text{scm}},$$

where the final bound is verified numerically for $\varepsilon_{\text{scm}} \le 1/10$. □

The `ScaleMaintainer` interface guarantees that calls to $\texttt{ScaleMaintainer}_k.\texttt{GetNorm}$ return $\left\|\bar{x}_k \circ \exp((1-\kappa^{\tau_k})\bar{\delta}_k)\right\|_1$ to within a $1 \pm \varepsilon_{\text{scm}}$ multiplicative factor, and moreover that `Get` returns

entries from an ε_{scm}-padding of $\Pi_\Delta(\bar{x}_k \circ \exp((1-\kappa^{\tau_k})\bar{\delta}_k))$. Thus, applying Lemma 37 with our definition of $\hat{x}$ in (3.44) yields that $\hat{x}$ is a $10\varepsilon_{\text{scm}} = \varepsilon$-padding of x.

`ApproxExpMaintainer` initialization and updates

We give the implementation and prove runtimes of `Init`, `MultSparse`, `DenseStep`, and `UpdateSum`.

`Init`. Upon initialization of `ApproxExpMaintainer`, we set $\gamma_K = \max_{j\in[n]}[x_0]_j$ and $\tau_k = 0$ for all k. We let $S_K = [n]$ (so that $S_k = \emptyset$ for all $k < K$) and instantiate a single instance of `ScaleMaintainer` of rank K with parameters

$$\bar{x}_K = \frac{x_0}{\gamma_K},\ \bar{\delta}_K = \frac{v}{1-\kappa} - \log x_0,\ \varepsilon_{\text{scm}} = \frac{\varepsilon}{10},\ \lambda_{\text{scm}} = \min\left(\frac{\varepsilon}{n},\ \lambda\right).$$

It is clear that the invariant (3.40) holds at initialization, and that the coordinates of $\bar{x}_K$ lie in the appropriate range, since we assume that $x_0 \in [\lambda, 1]^n$. We will use the same choices of ε_{scm}, λ_{scm} for every `ScaleMaintainer` instance. The overall complexity of this operation is $O(n \log n \log^2 \omega)$.

`MultSparse`. We state the implementation of `MultSparse`, prove that the resulting update is (3.42), and finally give its runtime analysis. We perform $\texttt{MultSparse}(g)$ in sequence for each nonzero coordinate of g. Let j denote such nonzero coordinate and let k_j be such that $j \in S_{k_j}$; the operation consists of the following steps.

1. Remove j from S_{k_j} and delete the corresponding coordinate from $\texttt{ScaleMaintainer}_{k_j}$ (via a call to `Del`).
2. Let $S_0 = j$ and initialize $\texttt{ScaleMaintainer}_0$ with initial data $\bar{x}$ and $\bar{\delta}$ described below.
3. For k going from 1 to K, set $S_k \leftarrow S_k \cup S_{k-1}$ and $S_{k-1} = \emptyset$, merging $\texttt{ScaleMaintainer}_k$ and $\texttt{ScaleMaintainer}_{k-1}$ as described below. If the new set S_k satisfies $|S_k| \le 2^k$, break the loop; else, proceed to the next k.

We now state the initial data given to each `ScaleMaintainer` upon initialization in the steps above. Whenever a $\texttt{ScaleMaintainer}_k$ is created supporting $S_k \subseteq [n]$, we first compute δ_i for each $i \in S_k$ according to (3.40). When creating the singleton instance $\texttt{ScaleMaintainer}_0$ we perform the update

$$\delta_j \leftarrow \delta_j + g_j; \tag{3.45}$$

this implements multiplication of the jth coordinate by $\exp(g_j)$. To instantiate $\texttt{ScaleMaintainer}_k$, we set $\tau_k = 0$, $\gamma_k \leftarrow \max_{i\in S_k} \exp([\delta]_i)$ and modify δ according to

$$[\delta]_{S_k} \leftarrow \max\left\{[\delta]_{S_k}, \log\left(\lambda_{\text{scm}} \cdot \gamma_k\right)\right\}. \tag{3.46}$$

In other words, we raise very small entries of $[\delta]_{S_k}$ to ensure that the ratio between any two entries

of $[\exp(\delta)]_{S_k}$ is in the range $[\lambda_{\text{scm}}^{-1}, \lambda_{\text{scm}}]$. We then give $\texttt{ScaleMaintainer}_k$ the initial data

$$\bar{x}_k = \frac{1}{\gamma_k} [\exp(\delta)]_{S_k}, \ \bar{\delta}_k = \left[\frac{v}{1-\kappa} - \delta\right]_{S_k}. \tag{3.47}$$

It is clear that entries of $\bar{x}_k$ are in the range $[\lambda_{\text{scm}}, 1]$, and invariant (3.40) holds at initialization, as $\tau_k = 0$. Therefore, the operation (3.45) implements $x \leftarrow \Pi_\Delta(x \circ \exp(g))$ exactly; it remains to show that the operation (3.46) amounts to an ε-padding of $\exp(\delta)/\left\|\exp(\delta)\right\|_1$. We do so by invoking the following lemma, substituting the values of δ before (3.45) for δ^- and δ^+ respectively, and $\lambda_{\text{scm}} \leq \varepsilon/n$ for ρ.

Lemma 38. *Let $\delta^-, \delta^+ \in \mathbb{R}^n$ satisfy*

$$[\delta^+]_i = \max\left\{[\delta^-]_i, \max_j [\delta^-]_j + \log\rho\right\} \text{ for all } j \in [n] \text{ and } \rho \leq 1.$$

Then, $\exp(\delta^+)/\left\|\exp(\delta^+)\right\|_1$ is a ρn-padding of $\exp(\delta^-)/\left\|\exp(\delta^-)\right\|_1$.

Proof. Let $x = \exp(\delta^-)/\left\|\exp(\delta^-)\right\|_1$, $x' = \exp(\delta^+)/\left\|\exp(\delta^+)\right\|_1$, and $\tilde{x} = \exp(\delta^+)/\left\|\exp(\delta^-)\right\|_1$. Clearly $x' = \tilde{x}/\left\|\tilde{x}\right\|_1$ and $\tilde{x} \geq x$ element-wise. Moreover, letting $M = \max_j \exp([\delta^-]_j)$, we have

$$\left\|\tilde{x} - x\right\|_1 = \frac{\left\|\exp(\delta^+) - \exp(\delta^-)\right\|_1}{\left\|\exp(\delta^-)\right\|_1} \leq \frac{\rho M |\{i \mid [\delta^+]_i \neq [\delta^-]_i\}|}{\left\|\exp(\delta^-)\right\|_1} \leq \frac{\rho M \cdot n}{\left\|\exp(\delta^-)\right\|_1} \leq \rho n,$$

establishing the ρn-padding property. □

Finally, we discuss runtime. Recall that the cost of initializing a `ScaleMaintainer` with $|S|$ elements is $O(|S| \log n \log^2 \omega)$. So, step 1 of our implementation of `MultSparse`, i.e., calling `Del` once and initializing a rank-1 `ScaleMaintainer` per nonzero element, costs $O(\left\|g\right\|_0 \log n \log^2 \omega)$. We now discuss costs of merging in step 2. We show these merges cost an amoritized $O(\log^2 n \log^2 \omega)$ per nonzero coordinate of g, leading to the claimed bound. Specifically, we show the cost of T deletions and initializations due to nonzero entries of `MultSparse` arguments is $O(T \log^2 n \log^2 \omega)$. Consider the number of times a rank-k set can be created through merges: we claim it is upper bounded by $O\left(T/2^k\right)$. It follows that the overall complexity of step 2 is

$$O\left(\sum_{k=0}^{K} 2^k \frac{T}{2^k} \log n \log^2 \omega\right) = O(T \log^2 n \log^2 \omega).$$

The claimed bound on the number of rank-k merges holds because at least 2^{k-1} deletions (and hence that many `MultSparse` calls) must occur between consecutive rank-k merges. To see this, for each k, maintain a potential Φ_k for the sum of cardinalities of all rank ℓ sets for $\ell < k$. Each deletion increases Φ_k by at most 1. For an insertion merge to create a rank-k set, Φ_k must have been at least

$2^{k-1}+2$; after the merge, it is 0, as in its creation, all rank-ℓ sets for $\ell < k$ must have been merged. So, there must have been at least 2^{k-1} deletions in between merges.

`DenseStep`. To implement `DenseStep` we simply increment $\tau_k \leftarrow \tau_k + 1$ for all k; clearly, this takes time $O(\log n)$. We now show that (3.40) is maintained, i.e., that the resulting update to the variable δ under `DenseStep` is (3.41). Recall that $\texttt{ScaleMaintainer}_k$ is initialized according (3.47). Clearly, (3.40) holds at initialization for the set S_k, as $1-\kappa^0 = 0$. We now show that it continues to hold after any number of `DenseStep` calls. Let δ_0 be the value of $[\delta]_{S_k}$ when $\texttt{ScaleMaintainer}_k$ is initialized, and let δ_τ be the value of $[\delta]_{S_k}$ after τ calls to `DenseStep`, each performing the update $x \leftarrow \Pi_\Delta(x^\kappa \circ \exp(v))$. This is consistent with the update $\delta_{\tau+1} = \kappa\delta_\tau + [v]_{S_k}$, which requires

$$\delta_\tau = \kappa^\tau \delta_0 + \sum_{\tau'=0}^{\tau-1} \kappa^{\tau'}[v]_{S_k} = \kappa^\tau\delta_0 + \frac{1-\kappa^\tau}{1-\kappa}[v]_{S_k} = \log(\gamma_k \bar{x}_k) + (1-\kappa^\tau)\bar{\delta}_k,$$

where in the final transition we substituted $\delta_0 = \log(\gamma_k\bar{x}_k)$ and $[v]_{S_k} = (1-\kappa)[\bar{\delta}_k + \delta_0]$ according to (3.47). We see that the required of form of δ_τ is identical to its definition (3.39) and consequently that (3.40) holds.

`UpdateSum`. We maintain the running sum s via the invariant

$$[s]_{S_k} = [u]_{S_k} + \texttt{ScaleMaintainer}_k\texttt{.GetSum}(),\ \forall k \in [K], \tag{3.48}$$

which we preserve in two separate procedures. First, whenever `UpdateSum`() is called, we compute the quantity Γ defined in (3.43), and for each $k \in [K]$ call

$$\texttt{ScaleMaintainer}_k\texttt{.UpdateSum}\left(\frac{\gamma_k \texttt{ScaleMaintainer}_k\texttt{.GetNorm}(1-\kappa^{\tau_k})}{\Gamma}\right).$$

It is straightforward to see that this indeed preserves the invariant (3.48) for our definition of $\hat{x}$ in (3.44), and takes time $O(\log n \log\omega)$. Next, whenever a coordinate is deleted from a $\texttt{ScaleMaintainer}_k$ instance, or an entire $\texttt{ScaleMaintainer}_k$ instance is deleted due to a merge operation, we update

$$u_j \leftarrow u_j + \texttt{ScaleMaintainer}_k\texttt{.GetSum}(j)$$

for every deleted coordinate j, or j involved in the merge, respectively. We charge the cost of this operations to that of new `ScaleMaintainer` instance, which we accounted for in the analysis of `MultSparse`.

Queries

`Get`(j). Recalling our definition of $\hat{x}$ (3.44), we compute Γ in time $O(\log n \log\omega)$ by obtaining $\texttt{GetNorm}(1-\kappa^{\tau_k})$ for each k, and then call $\texttt{Get}(j, 1-\kappa^{\tau_k})$ in time $O(\log\omega)$ for the relevant k.

`GetSum`(j). Recalling our definition of s (3.48), we implement `GetSum`(j) in $O(\log \omega)$ time via a single call to `GetSum` on the relevant `ScaleMaintainer` instance, and querying a coordinate of u.

`Sample`. Recalling (3.44), we first compute Γ, as well as all γ_k`ScaleMaintainer`$_k$.`GetNorm`$(1 - \kappa^{\tau_k})$, in $O(\log n \log \omega)$ time. We then sample an instance `ScaleMaintainer`$_k$, for $0 \leq k \leq K$, proportional to the value γ_k`ScaleMaintainer`$_k$.`GetNorm`$(1 - \kappa^{\tau_k})$, in $O(\log n)$ time. Finally, for the sampled instance, we call `ScaleMaintainer`$_k$.`Sample`$(1-\kappa^{\tau_k})$ to output a coordinate in $O(\log n \log \omega)$ time. By the definition of $\hat{x}$ used by `ApproxExpMaintainer`, as well as the definitions used by each `ScaleMaintainer`$_k$ instance, it is clear this preserves the correct sampling probabilities.

3.5.3 `ScaleMaintainer`

Finally, we provide a self-contained treatment of `ScaleMaintainer`, the main building block in the implementation of `ApproxExpMaintainer` described above.

Interface

For ease of reference we restate the interface of `ScaleMaintainer`, where for the sake of brevity we drop the subscript scm from ε and λ, and use n rather than n' to denote the input dimension. Recall that for the vectors $\bar{x}$ and $\bar{\delta}$ given at initialization, the data structure keeps track of vectors of the form

$$\hat{x}[\sigma] := \text{a } \varepsilon\text{-padding of } \Pi_\Delta\left(\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right), \tag{3.49}$$

where σ is any scalar in the range $\{0\} \cup [\sigma_{\min}, 1]$.

The implementation of the data structure relies on three internal parameters: polynomial approximation order $p \in \mathbb{N}$, truncation threshold $R \geq 0$, and σ discretization level $K \in \mathbb{N}$. To satisfy the accuracy requirements we set these as

$$R = \Theta(1) \log \frac{1}{\varepsilon\lambda}, \quad p = \Theta(1) \log \frac{1}{\varepsilon\lambda}, \quad \text{and} \quad K = \left\lceil \log \frac{1}{\sigma_{\min}} \right\rceil;$$

we give the runtime analysis in terms of these parameters.

Category	**Function**	**Runtime**
initialize	`Init`$(\bar{x}, \bar{\delta}, \sigma_{\min}, \varepsilon, \lambda)$: require $\bar{x} \in [\lambda, 1]^n$	$O(npK \log n)$
update	`Del`(j): Remove coordinate j from $\bar{x}$, $\bar{\delta}$	$O(1)$
	`UpdateSum`(γ, σ): $s \leftarrow s + \gamma\hat{x}[\sigma]$	$O(p)$
query	`Get`(j, σ): Return $[\hat{x}[\sigma]]_j$	$O(p)$
	`GetSum`(j): Return $[s]_j$.	$O(pK)$
	`GetNorm`(σ): Return $1 \pm \varepsilon$ approx. of $\left\|\bar{x} \circ \exp(\sigma\bar{\delta})\right\|_1$	$O(p)$
sample	`Sample`(σ): Return j with probability $[\hat{x}[\sigma]]_j$	$O(p \log n)$

Overview

We now outline our design of `ScaleMaintainer`, where the main challenge is supporting efficient `GetNorm` operations under no assumptions on the numerical range of the input $\bar{\delta}$.

Exponential approximation via Taylor expansion. Our main strategy is to replace the exponential in the definition of $\hat{x}[\sigma]$ with its Taylor expansion of order $p = O(\log \frac{n}{\varepsilon\lambda})$, giving the following approximation to the $\texttt{GetNorm}(\sigma)$

$$\left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1 \approx \left\langle \bar{x}, \sum_{q=0}^{p} \frac{1}{q!}(\sigma\bar{\delta})^q \right\rangle = \sum_{q=0}^{p} \frac{\sigma^q}{q!} \left\langle \bar{x}, \bar{\delta}^q \right\rangle,$$

where qth powers are applied to $\bar{\delta}$ elementwise. By pre-computing all the inner products $\{\langle \bar{x}, \bar{\delta}^q \rangle\}_{q=0}^{p}$ at initialization, we may evaluate this Taylor approximation of `GetNorm` in time $O(p)$. The validity of the approximation relies on the following well-known fact.

Fact 2 (Theorem 4.1 in [466])**.** *Let $\varepsilon', R \geq 0$. A Taylor series $f_p(t) = \sum_{q=0}^{p} \frac{t^q}{q!}$ of degree $p = O(R + \log \frac{1}{\varepsilon'})$ satisfies*

$$|\exp(t) - f_p(t)| \leq \exp(t)\varepsilon' \quad \textit{for all } t \in [-R, 0].$$

Truncating small coordinates and σ discretization. For Fact 2 to directly imply the desired approximation guarantee for `GetNorm`, the entries of $\sigma\bar{\delta}$ must all lie in $[-R, 0]$ for some $R = \widetilde{O}(1)$. However, this will not hold in general, as our data structure must support any value of $\bar{\delta}$. For a fixed value of σ, we can work instead with a shifted and truncated version of $\bar{\delta}$, i.e.,

$$\tilde{\delta}[\sigma, \mu] := \max\{\bar{\delta} - \mu, -R/\sigma\},$$

where the offset μ is roughly the maximum element of $\bar{\delta}$. Fact 2 allows us to approximate the exponential of $\sigma\tilde{\delta}[\sigma, \mu]$, and for $R = \Theta(\log(\frac{n}{\varepsilon\lambda}))$ we argue that the truncation of the smallest entries of δ results in small multiplicative error. Unfortunately, the dependence of $\tilde{\delta}[\sigma, \mu]$ on σ would defeat the purpose of efficient computation, because it is impossible to precompute $\{\langle \bar{x}, \tilde{\delta}[\sigma, \mu]^q \rangle\}_{q=0}^{p}$ for every $\sigma \in [\sigma_{\min}, 1]$. To address this, we argue that truncation of the form $\tilde{\delta}[\hat{\sigma}, \mu]$ is accurate enough for any $\sigma \in [\hat{\sigma}/2, \hat{\sigma}]$. Therefore, it suffices to to discretize $[\sigma_{\min}, 1]$ into $K = \lceil \log \frac{1}{\sigma_{\min}} \rceil$ levels

$$\hat{\sigma}_k := 2^{k-1}\sigma_{\min}$$

and precompute $\langle \bar{x}, \tilde{\delta}[\hat{\sigma}_k, \mu]^q \rangle$ for every $k \in [K]$ in $q \leq p$. This allows us to compute $\texttt{GetNorm}(\sigma)$ in $O(p) = \widetilde{O}(1)$ time, with $O(npK) = \widetilde{O}(n)$ preprocessing time.

Supporting deletion via lazy offset selection. Had the dataset not supported deletions, we could have simply set μ to be the largest entry of $\bar{\delta}$ (independent of k). However, with deletions the

largest entry of $\bar{\delta}$ could change, potentially invalidating the truncation. To address this, we maintain a different threshold μ_k for every $k \in [K]$, and argue that the approximation remains valid if the invariant

$$\bar{\delta}_{\max} \le \mu_k \le \bar{\delta}_{\max} + \frac{R}{2\hat{\sigma}_k} \text{ for every } k \in [K] \tag{3.50}$$

holds, where $\bar{\delta}_{\max} := \max_j \bar{\delta}_j$. Writing

$$\tilde{\delta}[k] := \tilde{\delta}[\hat{\sigma}_k, \mu_k] = \max\left\{\bar{\delta} - \mu_k, -\frac{R}{\hat{\sigma}_k}\right\} \text{ for every } k \in [K], \tag{3.51}$$

the data structure only needs to maintain μ_k and $\left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle$ for every $k \in [K]$ in $q \le p$.

When deleting coordinate j, for every k we test whether the invariant (3.50) remains valid.[5] If it does, we keep μ_k the same and implement deletion (for this value of k) in time $O(p) = \widetilde{O}(1)$ by subtracting $[\bar{x}]_j[\tilde{\delta}[k]]_j^q$ from $\left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle$ for every $q \le p$. If the invariant is no longer valid, we reset μ_k to the new value of $\bar{\delta}_{\max}$ and recompute $\left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle$ for every $q \le p$. Note that the re-computation time is proportional to the number of un-truncated coordinates in the newly defined $\tilde{\delta}[k]$. The key observation here is that every re-computation decreases μ_k by at least $R/(2\hat{\sigma}_k)$ and so no element of $\bar{\delta}$ can remain un-truncated for more than two re-computation. Therefore, the cost of recomputing inner products due to deletions, for the entire lifetime of the data structure, is at most $O(npK) = \widetilde{O}(n)$, which we charge to the cost of initialization.

Explicit expression for $\hat{x}[\sigma]$. Following the preceding discussion, for any $\sigma \ge \sigma_{\min}$ we set

$$k^\star = \left\lceil \log_2 \frac{\sigma}{\sigma_{\min}} \right\rceil, \quad \text{so that} \quad \sigma \in \left[\frac{\hat{\sigma}_{k^\star}}{2}, \hat{\sigma}_{k^\star}\right],$$

and define

$$Z[\sigma] := e^{\sigma\mu_{k^\star}} \sum_{q=0}^{p} \frac{\sigma^q}{q!} \left\langle \bar{x}, \tilde{\delta}[k^\star]^q \right\rangle \approx \left\| \bar{x} \circ \exp\left(\sigma\bar{\delta}\right) \right\|_1 \tag{3.52}$$

$$\hat{x}[\sigma] := \frac{e^{\sigma\mu_{k^\star}}}{Z[\sigma]} \sum_{q=0}^{p} \frac{\sigma^q}{q!} \, \bar{x} \circ \tilde{\delta}[k^\star]^q \approx \Pi_\Delta\left(\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right), \tag{3.53}$$

with $\tilde{\delta}$ as defined in (3.51).

Correctness

We now prove that the approximation guarantees of `ScaleMaintainer` hold.

Proposition 12. *There exist $R = O(1) \cdot \log \frac{n}{\varepsilon\lambda}$ and $p = O(1) \cdot \log \frac{n}{\varepsilon\lambda}$ such that for all $\sigma \in [\sigma_{\min}, 1]$, if the invariant* (3.50) *holds we have that $Z[\sigma]$ is an ε multiplicative approximation of $\left\| \bar{x} \circ \exp\left(\sigma\bar{\delta}\right) \right\|_1$*

[5] We can query the maximum entry of $\bar{\delta}$ under deletions in $O(1)$ time via a standard data structure, e.g. a doubly-linked list of the sorted entries of $\bar{\delta}$.

and $\hat{x}[\sigma]$ *is a* ε*-padding of* $\Pi_\Delta\left(\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right)$*, with* $Z[\sigma]$ *and* $\hat{x}[\sigma]$ *defined in Eq.s* (3.52) *and* (3.53) *respectively.*

Proof. To simplify notation, we write $\mu = \mu_{k^\star}$ and $\hat{\sigma} = \hat{\sigma}_{k^\star}$. We begin by noting that the inequalities (3.50) and $\sigma \leq \hat{\sigma}$ imply that $\sigma\tilde{\delta}_i[k^\star] \in [-R, 0]$ for every $i \in [n]$ and we may therefore apply Fact 2 to obtain

$$\sum_{q=0}^{p} \frac{\sigma^q}{q!}\, \bar{x}_j \tilde{\delta}_i[k^\star]^q \geq (1-\varepsilon')\bar{x}_i \exp(\sigma\tilde{\delta}_i[k]) \geq (1-\varepsilon')e^{-\sigma\mu}\bar{x}_i \exp(\sigma\bar{\delta}_i[k]) \tag{3.54}$$

for every $i \in [n]$. Therefore, we have

$$Z[\sigma] \geq (1-\varepsilon') \left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1. \tag{3.55}$$

Similarly, we have

$$\sum_{q=0}^{p} \frac{\sigma^q}{q!}\, \bar{x}_j \tilde{\delta}_i[k^\star]^q \leq (1+\varepsilon')\bar{x}_i \exp(\sigma\tilde{\delta}_i[k]) \leq (1+\varepsilon')e^{-\sigma\mu}\bar{x}_i(\exp(\sigma\bar{\delta}_i[k]) + \exp(-\sigma R/\hat{\sigma}))$$

Note that the condition (3.50) also implies that $\tilde{\delta}_j[k^\star] \geq -R/(2\hat{\sigma})$ for some $j \in [n]$ (namely the maximal element of $\bar{\delta}$). Using also $\bar{x}_j \geq \lambda$, we have

$$e^{\sigma\mu}\exp(-\sigma R/\hat{\sigma}) \leq \exp(-\sigma R/(2\hat{\sigma}))\frac{\bar{x}_j}{\lambda}\exp(\sigma\bar{\delta}_j).$$

Taking $R \geq 2\log\frac{2n}{\lambda\varepsilon'}$ and recalling that $\sigma \geq \hat{\sigma}/2$, we have $\exp(-\sigma R/(2\hat{\sigma})) \leq \lambda\varepsilon'/(2n)$ and consequently

$$e^{\sigma\mu}\exp(-\sigma R/\hat{\sigma}) \leq \varepsilon'\bar{x}_j\exp(\sigma\bar{\delta}_j) \leq \frac{\varepsilon'}{n}\left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1.$$

Substituting back and using $\bar{x}_i \leq 1$ and $\varepsilon' < 1$ gives

$$e^{\sigma\mu}\sum_{q=0}^{p} \frac{\sigma^q}{q!}\, \bar{x}_j \tilde{\delta}_i[k^\star]^q \leq (1+\varepsilon')\bar{x}_i\exp(\sigma\bar{\delta}_i) + \frac{\varepsilon'}{n}\left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1. \tag{3.56}$$

Summing over $i \in [n]$, we obtain

$$Z[\sigma] \leq (1+2\varepsilon')\left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1 \tag{3.57}$$

Therefore, $Z[\sigma]$ is a $2\varepsilon'$-multiplicative approximation of $\left\|\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right\|_1$.

It remains to show that $\hat{x}[\sigma]$ is a ε-padding of $x[\sigma] := \Pi_\Delta\left(\bar{x} \circ \exp\left(\sigma\bar{\delta}\right)\right)$. First, if we define $\tilde{x} = \frac{1+2\varepsilon'}{1-\varepsilon'}\hat{x}[\sigma]$ then the bounds (3.54) and (3.57) imply that $\tilde{x} \geq x[\sigma]$ elementwise. Also, the

bounds (3.55) and (3.56) imply that

$$\hat{x}_i[\sigma] - x_i[\sigma] \leq \frac{(1+\varepsilon')x_i[\sigma] + \varepsilon'/n}{1-\varepsilon'}$$

for every $i \in [n]$. Therefore, for $\varepsilon' < 1/10$,

$$\|\tilde{x} - x[\sigma]\|_1 \leq \left(\frac{1+2\varepsilon'}{1-\varepsilon'}\right)^2 - 1 \leq 10\varepsilon',$$

so that $\hat{x}[\sigma]$ is a $10\varepsilon'$ padding of $x[\sigma]$. Taking $\varepsilon' = \varepsilon/10$ concludes the proof. □

Implementation: data structure state and initialization

Besides storing $\bar{x}$ and $\bar{\delta}$, the data structure maintains the following fields.

1. An offset $\mu_k \in \mathbb{R}$ for every $k \leq K = \left\lceil \log \frac{1}{\sigma_{\min}} \right\rceil$, initialized as $\mu_k = \max_j [\bar{\delta}]_j$ for all k.

2. A balanced binary tree with n leaves. For node v in the tree, $k \in [K]$ and $q \in \{0, \ldots, p\}$, we store

$$A_v[k,q] := \left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle_{S_v},$$

   where $\tilde{\delta}[k] = \max\{\bar{\delta} - \mu_k, -R/\hat{\sigma}_k\}$ as before, the set S_v contains the leaves in the subtree rooted in v, and $\langle a, b\rangle_S := \sum_{i \in S} a_i b_i$. When referring to the root of the tree we omit the subscript, i.e., we write

$$A[k,q] := \left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle.$$

3. A vector $u \in \mathbb{R}^n$ and coefficients $c_{k,q} \in \mathbb{R}$ for every $k \in [K]$ and $q \in \{0, \ldots, p\}$, for maintaining the running sum. We initialize them all to be 0. The running sum obeys the following invariant:

$$s = u + \sum_{k=1}^{K} \sum_{q=0}^{p} \frac{c_{k,q}}{q!} \bar{x} \circ \tilde{\delta}[k]^q. \tag{3.58}$$

4. A doubly linked list of the sorted entries of $\bar{\delta}$, with a pointer to the maximal element of $\bar{\delta}$ as well as pointers to the largest element smaller than $\mu_k - R/\hat{\sigma}_k$ for every $k \in [K]$.

Initializing the data structure for maintaining the maximum element takes time $O(n \log n)$ due to the need to sort $\bar{\delta}$. With it, initializing μ_k is trivial and so is the initialization of u and $c_{q,k}$. Initializing the data stored in the binary tree takes time $O(npK)$, since for every value k and q and internal node v with children v', v'' we can recursively compute $A_v[k,q]$ as $A_{v'}[k,q] + A_{v''}[k,q]$. We will also charge some additional deletion costs to the initialization runtime, resulting in the overall complexity $O(npK \log n)$.

Implementation: queries and sampling

`GetNorm`(σ). We compute $k^\star = \left\lceil \log_2 \frac{\sigma}{\sigma_{\min}} \right\rceil$ and return $Z[\sigma] = e^{\sigma k^\star} \sum_{q=0}^{p} \frac{\sigma^q}{q!} A[k^\star, q]$. Clearly, this takes $O(p)$ time and Proposition 12 provides the claimed approximation guarantee.

`Get`(j, σ). We compute $Z[\sigma]$ and $k^\star$ as described above and return $\frac{e^{\sigma \mu_{k^\star}}}{Z[\sigma]} \sum_{q=0}^{p} \frac{\sigma^q}{q!} \bar{x}_j \tilde{\delta}_j[k^\star]^q$ in accordance with the form (3.53) of $\hat{x}[\sigma]$. Again, this takes $O(p)$ time and Proposition 12 provides the claimed approximation guarantee.

`GetSum`(j). Recalling the invariant (3.58), we return $u_j + \sum_{k=1}^{K} \sum_{q=0}^{p} \frac{c_{k,q}}{q!} \bar{x}_j \tilde{\delta}_j[k]^q$ in time $O(pK)$.

`Sample`(σ). We perform a random walk from the root of our binary tree data structure to a leaf. A each internal node v with children v' and v'', we select node v' with probability

$$\frac{\langle \mathbf{1}, \hat{x}[\sigma] \rangle_{S_{v'}}}{\langle \mathbf{1}, \hat{x}[\sigma] \rangle_{S_v}} = \frac{\sum_{q=0}^{p} \frac{\sigma^q}{q!} A_{v'}[k^\star, q]}{\sum_{q=0}^{p} \frac{\sigma^q}{q!} A_v[k^\star, q]},$$

and otherwise select v'', where $k^\star = \left\lceil \log_2 \frac{\sigma}{\sigma_{\min}} \right\rceil$. We return the index associated with the leaf in which we end the walk; the probability of returning index j is exactly $[\hat{x}[\sigma]]_j$. Each step in the walk takes time $O(p)$ and there are $O(\log n)$ steps, so the total time is $O(p \log n)$.

Implementation: updates

`UpdateSum`(σ). Recalling the invariant (3.58) and the form (3.53) of $\hat{x}[\sigma]$, we compute $k^\star$ and $Z[\sigma]$ as in the `GetNorm` implementation, and update update

$$c_{k^\star, q} \leftarrow c_{k^\star, q} + \frac{e^{\sigma \mu_{k^\star}} \sigma^q}{Z[\sigma]}$$

for every $q \in \{0, \ldots, p\}$. This takes time $O(p)$.

`Del`(j). We set $[\bar{\delta}_j] \leftarrow -\infty$, remove the element corresponding to index j from the doubly linked list, and perform the following operations for each $k \in [K]$ separately. First, we check if the new maximum element of $\bar{\delta}$ is a least $\mu_k - R/(2\hat{\sigma}_k)$. If it is, we leave μ_k unchanged and we simply update

$$A_v[k, q] \leftarrow A_v[k, q] - \bar{x}_j \tilde{\delta}_j[k]^q$$

for every $q \leq p$ and node v on the path from the root to the leaf corresponding to index j. Since the length of the path is $O(\log n)$, this update takes time $O(p \log n)$.

Otherwise, the new maximum element is less than $\mu_k - R/(2\hat{\sigma}_k)$, and we must change μ_k in order to maintain the invariant (3.50). Let μ_k^{new} be the new maximum element of $\bar{\delta}$, and let

$$U_k = \left\{ i \;\middle|\; [\bar{\delta}]_i \geq \mu_k^{\text{new}} + \frac{R}{\hat{\sigma}_k} \right\}$$

be the new set of un-truncated indices. (We find the elements in this set when we update the pointer

to the first element smaller than $\mu_k - R/\hat{\sigma}_k$). We recompute $A_v[k,q] = \left\langle \bar{x}, \tilde{\delta}[k]^q \right\rangle_{S_v}$ for every $q \le p$ and every node v with a child in U_k. Performing the computation recursively from leaf to root, this take at most $O(|U_k| p \log n)$ time. To maintain the invariant (3.58) as the definition of $\tilde{\delta}[k]$ changes, we update

$$u_j \leftarrow u_j + \sum_{q=0}^{p} \frac{c_{k,q}}{q!} \bar{x}_j \left([\bar{\delta}_j - \mu_k^{\text{new}}]^q - [\max\left\{\bar{\delta}_j - \mu_k, -R/\hat{\sigma}_k\right\}]^q \right) \quad \text{for every } j \in U_k;$$

this update takes $O(|U_k|p)$ time. Finally, we update $\mu_k \leftarrow \mu_k^{\text{new}}$.

Summing over $k \in [K]$, deletion operations of the first kind (with μ_k unchanged) take at most $O(Kp \log n)$ time per call to `Del`. Operations of the second kind (with μ_k decreased) take time $O(Np \log n)$ throughout the data structure lifetime, where $N = \sum_{t \ge 1} \sum_{k=1}^{K} |U_k^{(t)}|$ and for each $k \in [K]$ we write $U_k^{(1)}, U_k^{(2)}, \ldots$ to denote the different sets U_k generated by all calls to `Del`. For each k, if μ_k is decreased at all then it must decrease by at least $R/(2\hat{\sigma}_k)$. Therefore, by definition of U_k, an index j can belong to $U_k^{(t)}$ for at most 2 values of t. Consequently, we have $N = O(nK)$. Therefore, deletion operations of the second kind contribute at most $O(nKp \log n)$ to the total runtime, which we charge to initialization.

3.6 Applications

In this section, we leverage the techniques of this chapter to obtain improved runtimes for solving certain structured optimization problems.

In Sections 3.6.1 and 3.6.2, we use a variant of our variance-reduced coordinate method in the ℓ_2-ℓ_1 setup to obtain algorithms for solving the maximum inscribed ball (Max-IB) and minimum enclosing ball (Min-EB) problems. Our algorithms improve upon the runtimes of those in [22] by a factor depending on the sparsity of the matrix. This improvement stems from a preprocessing step in [22] where the input is randomly rotated to improve a norm dependence of the algorithm. Our methods avoid this preprocessing and obtain runtimes dependent on the both the sparsity and numerical sparsity of the data, providing universal improvements in the sparse regime, in the non-degenerate case where the span of the points is full-rank.

In Section 3.6.3, we use the results of our variance-reduced algorithm in the ℓ_2-ℓ_2 setup (cf. Section B.4.2) to obtain improved regression algorithms for a variety of data matrices, including when the matrix is numerically sparse or entrywise nonnegative.

Our methods in this section rely on an extension of the outer loop of this chapter (Algorithm 11) for *strongly monotone minimax problems*, developed in our previous work [111]. Specifically, for a separable regularizer $r(x,y) = r^{\mathsf{x}}(x) + r^{\mathsf{y}}(y)$ on a joint space for any of our setups, consider the

following composite bilinear minimax problem:

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x,y) := y^\top Ax + \mu^{\mathsf{x}}\phi(x) - \mu^{\mathsf{y}}\psi(y), \text{ where } \phi = V^{\mathsf{x}}_{x'},\ \psi = V^{\mathsf{y}}_{y'}. \tag{3.59}$$

We call such problem a $(\mu^{\mathsf{x}},\mu^{\mathsf{y}})$-strongly monotone problem; this is a special case of a generalization of the notion of strong convexity, in the case of convex minimization. For general strongly-monotone problems, [111] provided a variant of Algorithm 11 with the following guarantee.

Proposition 13 (Proposition 5, [111])**.** *For problem* (3.59)*, denote* $\mu := \sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}$ *and* $\rho := \sqrt{\mu^{\mathsf{x}}/\mu^{\mathsf{y}}}$*. Let* $\mathcal{O}$ *be an* (α,ε)*-relaxed proximal oracle for operator* $g(x,y) := (\nabla_x f(x,y), -\nabla_y f(x,y))$*, let* Θ *be the range of* r*, and let* $\|(\nabla_x f(z), -\nabla_y f(z'))\|_* \le G$*, for all* $z,z' \in \mathcal{Z}$*. Let* z_K *be the output of* K *iterations of* `OuterLoopStronglyMonotone`*, Algorithm 7 of [111]. Then*

$$\mathbb{E}\mathrm{Gap}(z_K) \le \sqrt{2}G\sqrt{\left(\left(\frac{\alpha}{\mu+\alpha}\right)^K\left(\rho+\frac{1}{\rho}\right)\Theta + \frac{\varepsilon}{\mu}\right)}.$$

Each iteration $k\in[K]$ *consists of one call to* $\mathcal{O}$*, producing a point* $z_{k-1/2}$*, and one step of the form*

$$z_k \leftarrow \left\{\langle g(z_{k-1/2}), z\rangle + \alpha\hat{V}_{z_{k-1}}(z) + \mu\hat{V}_{z_{k-1/2}}(z)\right\}, \tag{3.60}$$

where $\hat{V} := \rho V^{\mathsf{x}} + \rho^{-1}V^{\mathsf{y}}$*. In particular, by setting*

$$\varepsilon = \frac{\mu\epsilon^2}{4G^2},$$

using $K = \widetilde{O}\alpha/\mu$ *iterations, we have the guarantee* $\mathbb{E}\mathrm{Gap}(z_K) \le \epsilon$*.*

For self-containment, we give a proof of (a generalization of) Proposition 19 in Appendix B.7.

The (α,ε)-relaxed proximal oracle works similarly as in Algorithm 12 except for the additional composite terms. For completeness we include the algorithm with its theoretical guarantees and implementation in Section B.5.2 (see Algorithm 68, Proposition 58 and Section 8).

In all of our applications discussed in this section, the cost of each step (3.60) is $O(\mathsf{nnz})$, stemming from the computation of $g(x,y)$. The resulting algorithms therefore have runtime

$$\tilde{O}\left((\mathsf{nnz} + (\text{cost of implementing } \mathcal{O}))\cdot\frac{\alpha}{\mu}\right).$$

3.6.1 Maximum inscribed ball

In the maximum inscribed ball (Max-IB) problem, we are given a polyhedron $P\subset\mathbb{R}^n$ defined by m halfspaces $\{H_i\}_{i\in[m]}$, each characterized by a linear constraint $H_i = \{x\in\mathbb{R}^n : \langle a_i, x\rangle + b_i \ge 0\}$, i.e. $P = \cap_{i\in[n]}H_i$. The goal is to (approximately) find a point $x^*\in P$ that maximizes the smallest

distance to any of the bounding hyperplanes H_i, i.e.

$$x_* \in \text{argmax}_{x \in P} \min_{i \in [n]} \frac{\langle a_i, x \rangle + b_i}{\|a_i\|_2} .$$

More formally, if the optimal radius of the maximum inscribed ball is r^*, the goal is to find an ϵ-accurate solution, i.e. a point in P which has minimum distance to all bounding hyperplanes at least $(1-\epsilon)r^*$.

Given halfspace information A, b where $A_{i:} = a_i$ for all $i \in [m]$, the polytope is defined by $P = \{x \mid Ax + b \geq 0\}$. We use the following notation in this section: $B := \|b\|_\infty$, r^* is the value of the maximum inscribed ball problem, R is the radius of the minimum enclosing ball, which is defined as the Euclidean ball containing P with smallest radius possible, x^* is the center of the maximum inscribed ball, and ρ is an upper bound on the aspect ratio R/r^*. As in [22], we will make the following assumptions:

1. The polytope is bounded, and thus $m \geq n$. This is without loss of generality since when the polytope is unbounded, the aspect ratio $\rho = \infty$, and our runtime result holds trivially.

2. $\|A_{i:}\|_2^2 = 1$ for all $i \in [m]$, so $\|A\|_{2\to\infty} = 1$, by properly scaling A (one can consider the trivial case when for some i, $a_i = 0$ separately).

3. The origin is inside polytope P, i.e. $O \in P$, by properly shifting P.

We also define the following constant (see Appendix B.4.3) in this section with respect to the rescaled matrix A,

$$L_{\mathsf{co}}^{2,1} := \min\left\{L_{\mathsf{co}}^{2,1,(1)}, L_{\mathsf{co}}^{2,1,(2)}, L_{\mathsf{co}}^{2,1,(3)}\right\} \leq \sqrt{\mathsf{rcs}} \cdot L_{\mathsf{rc}}^{2,1} \leq \sqrt{\mathsf{rcs}},$$

given the definitions of $L_{\mathsf{co}}^{2,1,(1)}, L_{\mathsf{co}}^{2,1,(2)}, L_{\mathsf{co}}^{2,1,(3)}$ as in (B.33), (B.34), and (B.35), and the second assumption above (namely, that $L_{\mathsf{rc}}^{2,1} = \max_{i\in[m]} \|A_{i:}\|_2 = 1$).

[22] show that solving Max-IB is equivalent to the following minimax problem:

$$r^* := \max_{x \in \mathbb{R}^n} \min_{y \in \Delta_m} f(x,y) := y^\top A x + y^\top b, \tag{3.61}$$

and moreover, to solve the problem to ϵ-multiplicative accuracy, it suffices to find x_ϵ^* that solves the minimax problem to ϵ-multiplicative accuracy in terms of the one-sided gap of the x block, i.e.

$$\min_{y \in \Delta_m} f(x_\epsilon^*, y) \geq (1-\epsilon) f(x^*, y^*),$$

where (x^*, y^*) is the optimal saddle point of problem (3.61). We first state several bounds on the parameters of the problem from [22].

Fact 3 (Geometric properties of Max-IB)**.** *We have* $\|x^*\|_2 \leq 2R$*, and*

$$r^* = \max_{x \in \mathbb{R}^n} \min_{y \in \Delta_m} f(x, y) := y^\top A x + y^\top b \leq B \leq 2R.$$

These facts imply that we can instead consider the constrained minimax problem (where we overload our definition of f for the rest of the section):

$$r^* := \max_{x \in \mathbb{B}^n} \min_{y \in \Delta_m} f(x, y) = y^\top \tilde{A} x + y^\top b, \quad \text{where } \tilde{A} = 2R \cdot A. \tag{3.62}$$

We first use a "warm start" procedure to find a constant multiplicative estimate of r^*, which uses the strongly monotone algorithm `OuterLoopStronglyMonotone` of [111] together with Algorithm 68 of Section B.5.2 as a relaxed proximal oracle on the (μ, μ)-strongly monotone problem

$$\max_{x \in \mathbb{B}^n} \min_{y \in \Delta_m} f_\mu(x, y) := y^\top \tilde{A} x + y^\top b + \mu \sum_{i \in [m]} [y]_i \log [y]_i - \frac{\mu}{2} \|x\|_2^2,$$

and a line search over parameter μ. The following lemma is an immediate consequence of Proposition 19 and Corollary 58, whose proof we defer to Appendix B.6.1.

Lemma 39. *We can spend* $\widetilde{O}\mathsf{nnz} + \rho\sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}}^{2,1}$ *time preprocessing to obtain a* 8*-multiplicative approximation* $\hat{r}$ *of* r^**, i.e.*

$$\frac{\hat{r}}{8} \leq r^* \leq \hat{r}.$$

Finally, we use our variance-reduced coordinate algorithm, namely Algorithm 68 as a relaxed proximal oracle in `OuterLoopStronglyMonotone` together with Proposition 19 once more to solve (3.62) to the desired accuracy. The implementation in Section 8 and complexity results in Section B.4.3 yield the runtime. This implementation crucially uses our development of the `ApproxExpMaintainer` data structure in order to obtain a runtime depending directly on rcs rather than dimensions of the matrix, as well as independence on B. For completeness, a proof can be found in Appendix B.6.1.

Theorem 12. *The algorithm of Section B.4.3 can be used to find an* ϵ*-accurate solution* x_ϵ^* *to Max-IB satisfying* $\min_{y \in \Delta_m} f(x_\epsilon^*, y) \geq (1 - \epsilon) r^*$ *with high probability in time* [6]

$$\widetilde{O}\left(\mathsf{nnz} + \frac{\rho\sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}}^{2,1}}{\epsilon}\right) = \widetilde{O}\left(\mathsf{nnz} + \frac{\rho\sqrt{\mathsf{nnz} \cdot \mathsf{rcs}}}{\epsilon}\right).$$

Remark 5. *Because we assumed* $m \geq n$*, in the case* A *is dense, up to logarithmic terms our runtime improves upon the runtime of* $\widetilde{O}\rho m\sqrt{n}/\epsilon$ *in [22] by a factor of at least*

$$\sqrt{\frac{mn}{\mathsf{nnz}} \cdot \frac{m}{\mathsf{rcs}}}$$

[6] Here $\widetilde{O}$ is hiding an additional factor of $\mathrm{polylog}(\|b\|_\infty)$ due to the additional cost in the runtime of `ApproxExpMaintainer`, caused by the linear term b (see Remark 2).

generically. This is an improvement when A is sparse or column-sparse, i.e. $\mathsf{nnz} \ll mn$, *or* $\mathsf{rcs} \ll m$. *Such a saving is larger when A has numerical sparsity so that e.g.* $\left(L_{\mathsf{co}}^{2,1}\right)^2 \leq \max_{i\in[m]} \|A_{i:}\|_1^2 + \left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right) < \mathsf{rcs} \cdot \max_{i\in[m]} \|A_i\|_2^2$.

3.6.2 Minimum enclosing ball

In the minimum enclosing ball (Min-EB) problem, we are given a set of data points $\{a_1, \ldots, a_m\}$ with $a_1 = 0, \max_{i\in[m]} \|a_i\| = 1$.[7] The goal is to find the minimum radius R^* such that there exists a point x with distance at most R^* to all points. Following the presentation of [22], we consider Min-EB in an equivalent form. Define the vector b to have $b_i = \frac{1}{2}\|a_i\|_2^2$ entrywise. Then, Min-EB is equivalent to the minimax problem

$$R^* := \min_{x\in\mathbb{R}^n} \max_{y\in\Delta^m} \frac{1}{2}\sum_i y_i \|x - a_i\|_2^2 = \min_{x\in\mathbb{R}^n} \max_{y\in\Delta^m} f(x,y), \text{ where } f(x,y) := y^\top Ax + y^\top b + \frac{1}{2}\|x\|_2^2 . \tag{3.63}$$

By assumption, $\|A\|_{2\to\infty} = 1$. We let (x^*, y^*) be the optimal solution to the saddle point problem. We first state several bounds on the quantities of the problem. These bounds were derived in [22] and obtained by examining the geometric properties of the problem.

Fact 4. *The following bounds hold:* $\|x^*\|_2 \leq 1$, *and* $R^* \geq 1/8$.

To achieve a multiplicative approximation, since $R^* \geq 1/8$ by Fact 4, it suffices to obtain a pair $(x_\epsilon^*, y_\epsilon^*)$ achieving $\max_y f(x_\epsilon^*, y) - \min_x f(x, y_\epsilon^*) \leq \epsilon/8$. In light of minimax optimality, Lemma 40 (proved in Section B.6.2) shows that it suffices to consider, for $\epsilon' = \Theta(\epsilon/\log m)$, solving the following $(1, \epsilon')$-strongly monotone problem to sufficient accuracy:

$$\min_{x\in\mathbb{R}^n} \max_{y\in\Delta^m} f_{\epsilon'}(x,y) := y^\top Ax + y^\top b - \epsilon' \sum_{i\in[m]} [y]_i \log [y]_i + \frac{1}{2}\|x\|_2^2 . \tag{3.64}$$

Lemma 40. *Setting $\epsilon' = \epsilon/(32\log m)$, an $\epsilon/16$-accurate solution or (3.64) is an $\epsilon/8$-accurate solution to the original problem (3.63).*

As an immediate result of the above lemma, the runtime in Section B.4.3 and the correctness proofs of Proposition 19 and Corollary 58, we obtain the following guarantee.

Theorem 13. *The strongly monotone algorithm* `OuterLoopStronglyMonotone` *of [111], using Algorithm 68 of Section B.5.2 and the estimator of Section B.4.3 as a relaxed proximal oracle, finds an ϵ-accurate solution x_ϵ^* to Min-EB satisfying $R^* \leq \max_y f(x_\epsilon^*, y) \leq (1+\epsilon)R^*$ with high probability in time*

$$\widetilde{O}\mathsf{nnz} + \frac{\sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}}^{2,1}}{\sqrt{\epsilon}} = \widetilde{O}\mathsf{nnz} + \frac{\sqrt{\mathsf{nnz} \cdot \mathsf{rcs}}}{\sqrt{\epsilon}}.$$

[7] This can be assumed without loss of generality by shifting and rescaling as in [22] and considering the trivial case when all $a_i, i \in [m]$ are equal.

Remark 6. *When $m \geq n$,[8] up to logarithmic terms our runtime improves the $\widetilde{O}m\sqrt{n}/\sqrt{\epsilon}$ runtime of [22] by a factor of*

$$\sqrt{\frac{mn}{\mathsf{nnz}} \cdot \frac{m}{\mathsf{rcs}}}$$

generically. This is an improvement when A is sparse or column-sparse, i.e. $\mathsf{nnz} \ll mn$, or $\mathsf{rcs} \ll m$. As in Section 3.6.1, the improvement is larger when A is numerically sparse, i.e. when $\left(L_{\mathsf{co}}^{2,1}\right)^2 \leq \max_{i\in[m]} \|A_{i:}\|_1^2 + \left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right) < \mathsf{rcs} \cdot \max_{i\in[m]} \|A_i\|_2^2$.

3.6.3 Regression

We consider the standard ℓ_2 linear regression problem in a data matrix $A \in \mathbb{R}^{m\times n}$ and vector $b \in \mathbb{R}^m$, i.e. $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2$. In particular, we consider the equivalent primal-dual form,

$$\min_{x\in\mathbb{R}^n} \max_{y\in\mathbb{B}^m} f(x, y) := y^\top (Ax - b). \tag{3.65}$$

Throughout, we assume the smallest eigenvalue of $A^\top A$ is $\mu > 0$ and denote an optimal solution to (3.65) by $z^* = (x^*, y^*)$ (where x^* is the unique solution to the regression problem). Our strategy is to consider a sequence of modified problems, parameterized by $\beta > 0$, $x' \in \mathbb{R}^n$:

$$\min_{x\in\mathbb{R}^n} \max_{y\in\mathbb{B}^m} f_{x'}^{\beta}(x, y) := y^\top (Ax - b) + \frac{\beta}{2} \|x - x'\|_2^2 - \frac{\beta}{2} \|y\|_2^2 . \tag{3.66}$$

We denote the optimal solution to (3.66) by $z^*_{(\beta,x')} = (x^*_{(\beta,x')}, y^*_{(\beta,x')})$; when clear from context, for simplicity we drop β and write $z^*_{x'} = (x^*_{x'}, y^*_{x'})$ (as $\beta = \sqrt{\mu}$ throughout our algorithm). Lemma 41 (proved in Section B.6.3) states a known relation between the optimal solutions for (3.65) and (3.66).

Lemma 41. *Letting (x^*, y^*) be the optimal solution for* (3.65) *and $(x^*_{x'}, y^*_{y'})$ be the optimal solution for* (3.66)*, the following relation holds:*

$$\|x^*_{x'} - x^*\|_2 \leq \frac{1}{1 + \frac{\mu}{\beta^2}} \|x' - x^*\|_2 .$$

We give a full implementation of the regression algorithm in Algorithm 69 (see Section B.6.3), and state its correctness and runtime in Theorem 14. The algorithm repeatedly solves problems of the form (3.66) in phases, each time using Lemma 41 to ensure progress towards x^*. Observing that each subproblem is (β, β)-strongly monotone, each phase is conducted via `OuterLoopStronglyMonotone`, an algorithm of [111], using the ℓ_2-ℓ_2 algorithms of Section B.4.2 as a proximal oracle. Due to the existence of composite terms, our inner loop steps are slightly different than in Section B.4.2; we give a more formal algorithm for the relaxed proximal oracle and its implementation in Algorithm 68 and Appendix B.5.2. We remark that by a logarithmic number of restarts per phase, a standard

[8] When $m < n$, the runtime of the algorithm in [22] still holds and is sometimes faster than ours.

argument boosts Theorem 14 to a high-probability claim.

Theorem 14. *Given data matrix $A \in \mathbb{R}^{m\times n}$, vector $b \in \mathbb{R}^m$, and desired accuracy $\epsilon \in (0,1)$, assuming $A^\top A \succeq \mu I$ for $\mu > 0$, Algorithm 69 outputs an expected ϵ-accurate solution $\tilde{x}$, i.e.*

$$\mathbb{E}\left[\|\tilde{x} - x^*\|_2\right] \leq \epsilon,$$

and runs in time

$$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \frac{\max\left\{\sqrt{\sum_i \|A_{i:}\|_1^2}, \sqrt{\sum_j \|A_{:j}\|_1^2}\right\}}{\sqrt{\mu}}\right).$$

We give two settings where the runtime of Algorithm 69 improves upon the state of the art.

Entrywise nonnegative A. In the particular setting when all entries of A are nonnegative,[9] by Proposition 55 our complexity as stated in Theorem 14 improves by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$ the runtime of accelerated gradient descent [417], which is the previous state-of-the-art in certain regimes with runtime $O\left(\mathsf{nnz} \cdot \|A\|_{\mathrm{op}}/\sqrt{\mu}\right)$. This speedup is most beneficial when A is dense.

Numerically sparse A. For numerically sparse A with $\|A_{i:}\|_1/\|A_{i:}\|_2 = O(1)$, $\|A_{:j}\|_1/\|A_{:j}\|_2 = O(1)$ for all $i \in [m]$, $j \in [n]$, we can choose $\alpha = \sqrt{\mu}$ in Algorithm 69 and obtain the runtime

$$O\left(\mathsf{nnz} + \frac{\max\left\{\sum_i \|A_{i:}\|_1^2, \sum_j \|A_{:j}\|_1^2\right\}}{\mu}\right) = O\left(\mathsf{nnz} + \frac{\|A\|_{\mathrm{F}}^2}{\mu}\right)$$

using the argument in Theorem 14. Under a (similar, but weaker) numerically sparse condition $\|A_{i:}\|_1/\|A_{i:}\|_2 = O(1)$, the prior state-of-the-art stochastic algorithm [300] obtains a runtime of $O(\mathsf{nnz} + \mathsf{rcs} \cdot \|A\|_{\mathrm{F}}^2/\mu)$, and the recent state-of-the-art result in the numerically sparse regime [260] improves this to $O(\mathsf{nnz} + (\|A\|_{\mathrm{F}}^2/\mu)^{1.5})$ when $\mathsf{rcs} = \Omega(\|A\|_{\mathrm{F}}/\sqrt{\mu})$. Improving universally over both, our method gives $O(\mathsf{nnz} + \|A\|_{\mathrm{F}}^2/\mu)$ in this setting.

[9] More generally, this holds for arbitrary $A \in \mathbb{R}^{m\times n}$ satisfying $\||A|\|_{\mathrm{op}} \leq \|A\|_{\mathrm{op}}$.

Chapter 4

Faster Approximate ℓ_∞ Regression and Maximum Flow

This chapter is based on [486], with Aaron Sidford.

4.1 Introduction

The classic problem of ℓ_∞ *regression* corresponds to finding a point x^* such that

$$x^* = \text{argmin}_{x \in \mathbb{R}^m} \|Ax - b\|_\infty, \text{ for } A \in \mathbb{R}^{n \times m}, \ b \in \mathbb{R}^n.$$

In this chapter, we are primarily concerned with developing iterative algorithms for approximately solving this problem. We use OPT to denote $\|Ax^* - b\|_\infty$ and our goal is to find an ϵ-*approximate minimizer* of the ℓ_∞-regression function, i.e. a point $x \in \mathbb{R}^m$ such that

$$\mathsf{OPT} \leq \|Ax - b\|_\infty \leq \mathsf{OPT} + \epsilon.$$

This problem has fundamental implications in statistics and optimization [482, 348, 349, 487]. In many of these settings, it is also useful to design iterative method machinery for the following more general problem of finding

$$x^* = \text{argmin}_{x \in S} \|Ax - b\|_\infty, \text{ for } A \in \mathbb{R}^{n \times m}, \ b \in \mathbb{R}^n, \ S = \{x \in \mathbb{R}^m : x_j \in [l_j, r_j] \ \forall j \in [m]\}$$

for some m pairs of scalar $l_j \leq r_j$ (possibly infinite). Note that this constrained problem is strictly more general than the standard one as setting $l_j = -\infty, r_j = \infty, \ \forall j \in [m]$ recovers the unconstrained problem. In this chapter, for simplicity, the domain constraint will only be $x \in [-1, 1]^m$ (though

our results apply to the more general case; see Appendix C.1.2 for a formal statement).

Definition 8 (Box-constrained ℓ_∞ regression)**.** *We call the problem of solving, for regression matrix $A \in \mathbb{R}^{n\times m}$ and demands $b \in \mathbb{R}^n$,*

$$\min_{x\in[-1,1]^m} \|Ax - b\|_\infty ,$$

the box-constrained ℓ_∞ regression problem. We refer to any $x' \in [-1,1]^m$ such that

$$\|Ax' - b\|_\infty - \min_{x\in[-1,1]^m} \|Ax - b\|_\infty \leq \epsilon$$

as an ϵ-approximate minimizer.

Many natural optimization problems can be written in the form of box-constrained ℓ_∞ regression, e.g. the maximum flow problem and more broadly linear programming [349], and thus faster methods for solving box-constrained ℓ_∞ regression can imply faster algorithms for common problems in theoretical computer science. Therefore, the central goal of this chapter is to provide faster algorithms for computing ϵ-approximate minimizers to ℓ_∞-regression, that when specialized to the maximum flow problem, achieve faster running times.

4.1.1 Regression results

In this chapter we show how to apply ideas from the literature on coordinate descent methods (see Section 4.1.3) to obtain faster algorithms for approximately solving box-constrained ℓ_∞ regression. We show that by assuming particular sampling and smoothness oracles (which are implementable given sparsity assumptions on A), we obtain a randomized algorithm which improves upon the the classic gradient descent based methods across a broad range of parameters and attains an ϵ^{-1} dependence in the runtime. We show the following in Section 4.3.4.

Theorem 15 (Accelerated box-constrained ℓ_∞ regression)**.** *There is an algorithm initialized that ϵ-approximately minimizes the box-constrained ℓ_∞ regression problem (Definition 8) in time*

$$\tilde{O}\left(mc + \frac{\left(\min(m,n) + \sqrt{m\min(n,s)}\right) c\,\|A\|_\infty}{\epsilon}\right),$$

where each column of $A \in \mathbb{R}^{n\times m}$ has at most c non-zero entries, and the optimizer x^* has $\|x^*\|_2 \leq s$.

Note that since $s \leq m$, the runtime is always at most $\widetilde{O}(mc\,\|A\|_\infty/\epsilon)$. Moreover, Theorem 15 generically achieves a runtime of $\widetilde{O}(\sqrt{mn}c\,\|A\|_\infty/\epsilon)$ in the case $n = O(m)$. We give a proof of the following simple extension, encapsulating the general box-constrained case as well as the unconstrained case, in Appendix C.1.2, which follows via a reduction to Theorem 15. We simplified the bounds for easy statement, but we remark that as they follow by a reduction, they admit similar improvements when e.g. $n, s \ll m$.

Corollary 6. *There is an algorithm that ϵ-approximately minimizes the box-constrained ℓ_∞ regression problem*

$$\min_{x\in[-r,r]^m} \|Ax-b\|_\infty$$

in $\tilde{O}\left(mcr\|A\|_\infty/\epsilon\right)$ time where each column of $A\in\mathbb{R}^{n\times m}$ has at most c non-zero entries. Moreover, there is an algorithm that ϵ-approximately minimizes the unconstrained ℓ_∞ regression problem

$$\min_{x\in\mathbb{R}^m} \|Ax-b\|_\infty$$

in $\tilde{O}\left(mcr\|A\|_\infty/\epsilon\right)$ time, where the optimizer is x^*, and $\|x_0-x^*\|_\infty\leq r$ for some given x_0.

The only other known box-constrained ℓ_∞ regression algorithm achieving an ϵ^{-1} dependence (improving upon the standard ϵ^{-2} dependence) without paying a dimension-dependent penalty is the recent breakthrough result of [483]. Pessimistic bounds on our guarantees attain a runtime matching that of [483] across a broad range of parameters (for example in the uniform sparsity case where $mc=O(\text{nnz}(A))$). In instances with more structured regression matrices, with sharper bounds on parameters n, s, we obtain improved runtimes. These improvements are attainable by modifying the algorithm to take steps in a nonuniform diagonal norm, obtaining tighter dependences on sparsity measures of the matrix and optimal solution, which we elaborate on in Sections 4.3 and 4.5. Because of these tighter dependencies, in many parameter regimes, including those for the maximum flow problem for even slightly dense graphs, our result improves upon [483].

Our work provides an alternative approach for accelerating ℓ_∞ gradient descent for certain highly structured optimization problems, i.e. ℓ_∞ regression. Whereas Sherman's work introduced an intriguing notion of area convexity and new regularizations of ℓ_∞ regression, our results are achieved by working with the classic smoothing of the ℓ_∞ norm and by providing a new accelerated coordinate descent method. We achieve our tighter bounds by exploiting local smoothness properties of the problem and dynamically sampling by these changing smoothnesses.

Our algorithm is inspired by, and builds upon, advances in non-uniform sampling for coordinate descent [553, 449, 425], as well as extragradient proximal methods [415, 420], and is similar in spirit to work on accelerated algorithms for approximating packing and covering linear programs [23] which too works with non-standard notions of smoothness. Our work overturns conventional wisdom that these techniques do not extend nicely to ℓ_∞ regression and the maximum flow problem. Interestingly, our algorithms gain an improved dependence on dimension and sparsity over [483] in certain cases while losing the parallelism of [483]. It is an open direction for future work as to see whether or not these approaches can be combined for a more general approach to minimizing ℓ_∞-smooth functions.

4.1.2 Maximum flow results

The classic problem of maximum flow roughly asks for a graph G with m (capacitated) edges and n vertices, how to send as many units of flow can be sent from a specified "source" vertex to a specified "sink" vertex while preserving flow conservation at all other vertices and without violating edge capacity constraints (i.e. the flow cannot put more units on an edge than the edge's capacity).

The maximum flow problem is known to be easily reducible to the more general problem of *minimum congestion flow*. Instead of specifying s and t this problem takes as input a vector $d \in \mathbb{R}^V$ such that $d^\top \mathbf{1} = 0$, where $\mathbf{1}$ is the all-ones vector. The goal of minimum congestion flow is to find a flow $f \in \mathbb{R}^E$ which routes d meaning, mean that the imbalance of f at vertex v is given by d_v, and subject to this constraint minimizes the *congestion*,

$$\max_{e \in E(G)} \left| \frac{f_e}{u_e} \right|$$

where f_e is the flow on some edge, and u_e is the capacity on that edge. We refer to the vector with entries f_e/u_e as the *congestion vector*. We call any flow which routes an amount within a $1+\epsilon$ multiplicative factor to the optimum an *ϵ-approximate maximum flow*.

A recent line of work beginning in [482, 318] solves the maximum flow problem by further reducing to constrained ℓ_∞ regression. To give intuition for the reduction used in this work, broadly inspired by [482, 318], we note that maximum flow in uncapacitated graphs can be rephrased as asking for the smallest congestion of a feasible flow, namely to solve the problem

$$f^* = \text{argmin}_{Bf=d} \|f\|_\infty$$

where the restriction $Bf = d$ for B the edge-vertex incidence matrix of a graph, and d the demands, enforces the flow constraints. This can be solved up to logarithmic factors in the running time by fixing some value F for $\|f\|_\infty$ and asking to optimally solve the problem

$$f^* = \text{argmin}_{\|f\|_\infty \leq F} \|Bf - d\|_\infty$$

where we note that the constraint $\|f\|_\infty \leq F$ can be decomposed as the indicator of a box so that this objective matches the form of (4.1). The exact reduction we use has a few modifications: the box constraint is more simply replaced by $\|f\|_\infty \leq 1$, and the regression objective is in a matrix RB, where R is a combinatorially-constructed preconditioner whose goal is to improve the condition number (and convergence rate) of the problem, and the problem is scaled for capacitated graphs (for a more detailed description, see Section 4.4.2).

In this chapter we show how to modify our algorithm for structured ℓ_∞ regression in order to obtain faster algorithms for maximum flow. We do so by leveraging the tighter dependence on the domain size (in the ℓ_2 norm rather than ℓ_∞) and coordinate smoothness properties of the function

to be minimized (due to the structure of the regression matrix). In particular we show the following.

Theorem 16 (ℓ_2 accelerated approximate maximum flow)**.** *There is an algorithm that takes time $\tilde{O}(m + \max(n, \sqrt{ns})/\epsilon)$ to find an ϵ-approximate maximum flow, where s is the squared ℓ_2 norm of the congestion vector of any optimal flow.*

Our running time improves upon the previous fastest running time of this problem of $\tilde{O}(m/\epsilon)$. Since $s \leq m$ we achieve a faster running time whenever the graph is slightly dense, i.e. $m = \Omega(n^{1+\delta})$ for any constant $\delta > 0$.

Interestingly our algorithm achieves even faster running times when there is a sparse maximum flow, i.e. a maximum flow in which the average path length in the flow decomposition of the optimal flow is small. Leveraging this, in Section 4.4.4 we provide several new results on exact undirected and directed maximum flow on uncapacitated graphs as well.

Theorem 17 (Improved algorithms for exact maximum flows)**.** *There are algorithms for finding an exact maximum flow in the following types of uncapacitated graphs.*

- *There is an algorithm which finds a maximum flow in an undirected, uncapacitated graph with maximum flow value F in time $\tilde{O}(m + \min(\sqrt{mn}F^{3/4}, m^{3/4}n^{1/4}\sqrt{F}))$.*
- *There is an algorithm which finds a maximum flow in an undirected, uncapacitated graph with a maximum flow that uses at most s edges in time $\tilde{O}(m + \sqrt{ms}n^{1/4}\max(n, s)^{1/4})$.*

Each of these runtimes improves upon previous work in some range of parameters. For example, the bound of $\tilde{O}(m + m^{3/4}n^{1/4}\sqrt{F})$ for undirected, uncapacitated graphs improves upon the previous best running times of $\tilde{O}(m\sqrt{F})$ achievable by [483] whenever $n = o(m)$ and of $\tilde{O}(m + nF)$ achievable by [314] whenever $m = o(nF^{2/3})$.

We also separately include the following result (which has no dependence on the sparsity s) for finding exact flows in general uncapcitated directed graphs, as it improves upon the running time of $\tilde{O}(m \cdot \max\{m^{1/2}, n^{2/3}\})$ achieved by [242] whenever $m = \omega(n)$ and $m = o(n^{5/3})$.

Theorem 18 (Exact maximum flow for directed uncapacitated graphs)**.** *There is an algorithm which finds a maximum flow in a directed, uncapacitated graph in time $\tilde{O}(m^{5/4}n^{1/4})$. When the maximum flow is s-sparse, there is an algorithm which finds a maximum flow in a directed, uncapacitated graph in time $\tilde{O}(mn^{1/4}\max(n, s)^{1/4})$.*

Although the runtime of [242] has been improved by the recent works of [377] achieving runtime $O(m^{10/7})$ and of [348] achieving runtime $\tilde{O}(m\sqrt{n})$, which dominate our $\tilde{O}(m^{5/4}n^{1/4})$ runtime, they do it using sophisticated advances in interior point methods, whereas our algorithm operates using a *first-order method* which only queries gradient information of the objective function, rather than second-order Hessian information. In particular, our algorithm is the first to improve runtimes for directed graphs while relying only on first-order information of the objective function. We find

it interesting that our result achieves any running time improvement for unit capacity maximum flow over [242] without appealing to interior point machinery and think this may motivate further research in this area, namely designing first-order methods for structured linear programs.

4.1.3 Previous work

Here we embark on a deeper dive into the context of the problems and tools discussed in this chapter. *Solving the ℓ_∞ regression problem.* For a non-differentiable function such as $f(x) = \|x\|_\infty$, it is possible to use the toolkit for linear programming (including interior point and cutting plane [348, 349]) to obtain iterative algorithms for approximate minimization. However, these particular algorithms have a larger dependence on dimension, and it is widely believed that the iteration complexity is inherently dimension-dependent. A first-order iterative algorithm with a better dependence on dimension for approximately solving the regression problem was developed by [419] and proceeds in two stages. First, the algorithm constructs a smooth approximation to the original function, which is typically explicitly derived via regularizing the dual function using a regularizer which is both smooth and bounded in range. The smooth approximation is constructed such that approximately minimizing the approximate function is sufficient to approximately minimize the original function. Second, a first-order method such as gradient descent in a particular norm, or one of its many variants, is applied to approximately minimize the smoothed function.

One of the earlier works to develop algorithms using first-order methods under this framework to solve the regression problem is [419]. One regularizer used in this work for optimization over a dual variable in the simplex was the entropy regularizer, which yields the smooth approximation to the ℓ_∞ norm defined by $\text{smax}_\alpha(x) = \alpha \log(\sum_j \exp(x_j/\alpha))$. Essentially, the methods presented in this work converge to an ϵ-approximate solution in a number of iterations proportional to either $O(\epsilon^{-2})$ or $O(\sqrt{m}\epsilon^{-1})$, hiding problem-specific dependencies on smoothness and domain size. However, the cost of each iteration involves computing a whole gradient, which incurs another multiplicative loss of the dimension in the runtime.

Several other works which aimed to solve the regression problem via considering a smooth minimax formulation, including [415] and [420], incurred the same fundamental barrier in convergence rate. These works aimed to pose the (smooth) regression problem as finding the saddle point of a convex-concave function via a specially-constructed first-order method. The main barrier to improving prior work up to this point has been the inability to construct regularizers of small range which are strongly convex with respect to the ℓ_∞ norm. For some time, these issues posed a barrier towards finding faster algorithms for the regression problem, and many related problems.

Very recently, Sherman [483] presented an alternative method which was able to break this barrier and attain an $O(1/\epsilon)$ iteration count for finding approximate solutions to the regression problem, where each iteration can be applied in time to compute a gradient. The algorithm used was a variation of Nesterov's dual extrapolation method [420] for approximately finding a saddle point

Year	Author	Method	Iteration Complexity	Iteration Cost	Norm
2003	[419]	Smoothing	$O(\epsilon^{-2})$	$O(m)$	ℓ_∞
			$O(\epsilon^{-1})$	$O(m)$	ℓ_2
2004	[415]	Mirror prox	$O(\epsilon^{-2})$	$O(m)$	ℓ_∞
			$O(\sqrt{m}\epsilon^{-1})$	$O(m)$	ℓ_∞
2005	[420]	Dual extrapolation	$O(\epsilon^{-2})$	$O(m)$	ℓ_∞
			$O(\sqrt{m}\epsilon^{-1})$	$O(m)$	ℓ_∞
2017	[483]	Area-convexity	$O(\epsilon^{-1})$	$O(m)$	ℓ_∞
2018	This chapter	Local smoothness	$O(\sqrt{m}\epsilon^{-1})$	$\tilde{O}(d)$	ℓ_2

Table 4.1: Dependencies of algorithms for ℓ_∞ regression in $A \in \mathbb{R}^{n\times m}$ on various problem parameters. Note that there is up to an $O(\sqrt{m})$ discrepancy between the ℓ_2 and ℓ_∞ norms. Here, d is the maximum number of nonzero entries in any column of A.

in a convex-concave function, adapted to work for regularizers satisfying a weaker property known as area convexity, and an analysis of its convergence. As a corollary, this obtained the currently fastest-known algorithm for approximate maximum flow.

Abbreviated history of first-order methods, emphasizing coordinate-based methods. First-order methods for convex optimization have a long history. Gradient descent methods with error decaying in k iterations as $O(1/\sqrt{k})$ for Lipschitz functions and $O(1/k)$ for smooth functions have been well studied (for example, see [418] or [98] for a more detailed exposition), and applied in many important settings.

Nesterov gave the first gradient-based algorithm for minimizing functions smooth in the Euclidean norm which converged at the rate $O(1/k^2)$. The method is optimal in the sense that it matched known lower bounds for smooth functions. Unfortunately, this method does not apply generically to functions which are smooth in other norms, in the same way that unaccelerated variants do, without possibly paying an additional dependence on the dimension. In particular, the accelerated convergence rate depends on the regularizer that the mirror descent steps use, and thus the analysis incurs a loss based on the size of the regularizer, which is the barrier in the aforementioned ℓ_∞-smooth function case. Specifically, it is a folklore result that any function strongly-convex over $[-1, 1]^n$ in the ℓ_∞ norm has range at least $n/2$, which we show in Appendix C.1.1.

There has been much interest in applying *randomized* first order methods to more efficiently obtain an approximate minimizer on expectation, when the convex optimization problem has certain structure. One example of these randomized methods in the literature is coordinate descent, studied first in [423]. The main idea is that using crude, computationally efficient, approximations to the full gradient, one is still able to find an approximate minimizer on expectation. One benefit is that coordinate descent admits a more fine-grained analysis of convergence rate, based on structural properties of the function, i.e. the smoothness of the function in each coordinate.

Generalizations of standard coordinate descent have received much attention recently, both for their powerful theoretical and practical implications. [423] provided an accelerated version of the standard coordinate descent algorithm, but the naive implementation of its steps were inefficient, taking linear time in the dimension. The study of efficient accelerated coordinate descent methods (which converge at the rate $O(1/k^2)$ without an additional dependence on dimension) was pioneered by [347], and since then a flurry of other works, including [222, 553, 449] have improved the rate of convergence and generalized the methods to composite functions with a separable composite term, of the form $F(x) = f(x) + \sum_j \psi_j(x_j)$. We remark that our box constraint can be represented as such a separable composite term in the objective, and our constrained accelerated coordinate descent algorithm is an adaptation of such composite methods. For a more detailed history of the study of coordinate descent methods, we refer the reader to [222].

Accelerated coordinate based methods have proven to be useful in many ways when applied to problems in theoretical computer science. For example, the authors of [347] framed graph Laplacian system solvers as a coordinate descent problem to give better runtime guarantees. One particularly interesting example that highlighted the potential for using accelerated coordinate descent in minimizing entropy-based functions was the work of [23] in solving packing and covering LPs, where the constraint matrix is nonnegative, in which they also attained a $O(1/\epsilon)$ method complexity. Conventional wisdom is that these results are specific to the structure of the particular problem, so any exploration of accelerated methods in greater generality is particularly interesting.

Maximum flow. The maximum flow problem is a fundamental problem in combinatorial optimization that has been studied extensively for several decades. Until recently, the toolkit used to solve the problem has been primarily combinatorial, culminating in algorithms with runtime roughly $\tilde{O}(\min\{mn^{2/3}, m^{3/2}\})$ for finding a maximum flow in graphs with m edges and n vertices and polynomially bounded capacities [242], and $\tilde{O}(m+nF)$ for finding a maximum flow in undirected graphs with m edges, n vertices, and a maximum flow value of F [314].

Breakthroughs in the related problem of electrical flow using tools from continuous optimization and numerical linear algebra were first achieved by Spielman and Teng [491] who showed that solving a linear system in the Laplacian of a graph could be done in nearly linear time, which is equivalent to computing an electrical flow.

Notably, the electric flow problem corresponds to approximately solving an ℓ_2 regression problem $\|Ax - b\|_2$, and the maximum flow problem corresponds to approximately solving an ℓ_∞ regression problem $\|Ax - b\|_\infty$. Accordingly, using the faster algorithms for electric flow combined with a multiplicative weights approach, the authors of [139] were able to make a breakthrough to approximately solve maximum flow with a runtime of $\tilde{O}(mn^{1/3})$, where $\tilde{O}$ hides logarithmic factors. Finally, using constructions presented in [376], the authors of [482] and [318] were able to reduce this runtime to almost linear, essentially using variants of preconditioned gradient descent in the ℓ_∞ norm. This

Year	Author	Complexity	Weighted	Directed
1998	[242]	$\tilde{O}(\min(m^{3/2}, mn^{2/3}))$	Yes	Yes
1998	[313]	$\tilde{O}(m\sqrt{n}\epsilon^{-1})$	Yes	No
2002	[314]	$\tilde{O}(m + nF)$	Yes	No
2011	[139]	$\tilde{O}(mn^{1/3}\epsilon^{-11/3})$	Yes	No
2012	[343]	$\tilde{O}(mn^{1/3}\epsilon^{-2/3})$	No	No
2013	[482], [318]	$\tilde{O}(m^{1+o(1)}\epsilon^{-2})$	Yes	No
2013	[377]	$\tilde{O}(m^{10/7})$	No	Yes
2014	[348]	$\tilde{O}(mn^{1/2})$	Yes	Yes
2016	[441]	$\tilde{O}(m\epsilon^{-2})$	Yes	No
2017	[483]	$\tilde{O}(m\epsilon^{-1})$	Yes	No
2018	This chapter	$\tilde{O}(m + (n + \sqrt{ns})\epsilon^{-1})$	Yes	No

Table 4.2: Complexity of maximum flow since [242] for undirected graphs with n vertices, m edges, where s is the ℓ_2^2 of the maximum flow's congestion, and F is the maximum flow value.

runtime was reduced to $\tilde{O}(m/\epsilon^2)$ by Peng in [441] by using a recursive construction of the combinatorial preconditioner. As previously mentioned, the ϵ^{-2} dependence in the runtime was a barrier typical of algorithms for minimizing ℓ_∞-smooth functions without worse dimension dependence, and was broken in [483], who attained a runtime of $\tilde{O}(m/\epsilon)$.

4.1.4 Organization

The rest of this chapter is organized as follows. Many proofs are deferred to the appendices.

- **Section 4.2: Overview.** We introduce the definitions and notation we use throughout the chapter, and give a general framework motivating our work.
- **Section 4.3: Regression.** We first give a framework for accelerated randomized algorithms which minimize the box-constrained ℓ_∞ regression function based on uniform sampling, as well as a faster one based on non-uniform sampling which assumes access to a coordinate smoothness and sampling oracle. To do so, we develop a new analysis of coordinate descent under a box constraint, amenable to dynamic coordinate sampling distributions, and show how to accelerate it via a primal-dual proximal point method. We then give efficient implementations for these oracles for structured problems.
- **Section 4.4: Maximum flow.** We state the reduction from the maximum flow problem to box-constrained ℓ_∞ regression problem. We first show how to attain a faster algorithm for maximum flow by exploiting combinatorial structure of the flow regression problem, using the regression algorithm we developed in the prior section. We then state the improved runtimes

which follow from a randomized primal-dual variation of our regression algorithm, given in Section 4.5. Further, we give the exact maximum flow runtimes achieved via rounding the resulting approximate flow of our improved method.

- **Section 4.5: Primal-dual coordinate acceleration.** We develop an algorithm with improved runtimes for the structured ℓ_∞ regression problem which results from the maximum flow reduction, and correspondingly yields further-improved flow runtimes.

4.2 Overview

4.2.1 Basic definitions

First, we define some basic objects and properties which we use throughout this paper.

General notations. We use $\tilde{O}(f(n))$ to denote runtimes of the following form: $O(f(n)\log^c f(n))$ where c is a constant. With an abuse of notation, we let $\tilde{O}(1)$ denote runtimes hiding polynomials in $\log n$ when the variable n is clear from context, and refer to such runtimes as "nearly constant."

Generally, we work with functions whose arguments are vector-valued variables in m-dimensional space, and may depend on a linear operator $A : \mathbb{R}^m \to \mathbb{R}^n$. Correspondingly we use $j \in [m]$ and $i \in [n]$ to index into these sets of dimensions, where $[m]$ is the set $\{1, 2, \ldots m\}$. We use e_j to denote the jth standard basis vector, i.e. the vector which is 1 in dimension j and 0 everywhere else. We use $u \circ v$ to denote the vector which is the coordinate-wise product, i.e. its j^{th} coordinate is $u_j v_j$.

Matrices. In this work, we deal with matrices $A \in \mathbb{R}^{n\times m}$ unless otherwise specified. Accordingly, we index into rows of A with $i \in [n]$, and into columns with $j \in [m]$. We refer to rows of A via $A_{i:}$ or a_i when it is clear from context, and columns via $A_{:j}$. We use $\text{nnz}(A)$ to denote the number of nonzero entries of A, and assume $\text{nnz}(A) \geq n + m - 1$, else we may drop a row or column.

We use $\text{diag}(w)$ to denote the diagonal matrix whose diagonal entries are the coordinates of a vector w. We call a square symmetric matrix A positive semi-definite if for all vectors x, $x^\top Ax \geq 0$ holds. For positive semi-definite matrices A, B we apply the Loewner ordering and write $A \preceq B$ if for all vectors x, $x^\top Ax \leq x^\top Bx$ holds.

Finally, we say that a matrix is c-column-sparse if no column of A has more than c nonzero entries.

Norms. We use $\|\cdot\|$ to denote an arbitrary norm when one is not specified. For scalar valued $p \geq 1$, including $p = \infty$, we use $\|x\|_p := (\sum_j x_j^p)^{1/p}$ to denote the ℓ_p norm. For vector valued $w \in \mathbb{R}^m_{\geq 0}$, we use $\|x\|_w^2 := \sum_j w_j x_j^2$ to denote the weighted quadratic norm, and for positive semidefinite matrix A, we define $\|x\|_A^2 = x^\top Ax$. Further, we let Δ^n be the simplex in n dimensions, e.g. $p \in \Delta^n \iff \|p\|_1 = 1, p \geq 0$ entrywise.

For a norm $\|\cdot\|$, the dual norm $\|\cdot\|_*$ is defined by $\|x\|_* := \max_{\|y\|\leq 1} y^\top x$. It is well known that the dual norm of ℓ_p is ℓ_q for $1/p + 1/q = 1$. For matrix A and a vector norm $\|\cdot\|$, we define the

matrix norm $\|A\| := \max_{\|x\|=1} \|Ax\|$. For example, $\|A\|_\infty$ is the largest ℓ_1 norm of a row of A.

Functions. We will primarily be concerned with minimizing convex functions $f(x)$ subject to the argument being restricted by a box constraint, where the domain is some scaled box B_∞^c unless otherwise specified. Whenever the function is clear from context, x^* will refer to any minimizing argument of the function. We use the term ϵ-approximate minimizer of a function f to mean any point x such that $f(x^*) \leq f(x) \leq f(x^*) + \epsilon$. Furthermore, we define the OPT operator to be such that $\mathsf{OPT}(f)$ is the optimal value of f, when this optimal value is well-defined.

For differentiable functions f we let $\nabla f(x)$ be the gradient and let $\nabla^2 f(x)$ be the Hessian. We let $\nabla_j f(x)$ be the value of the j^{th} partial derivative; we also abuse notation and use it to denote the vector $\nabla_j f(x) e_j$ when it is clear from context.

Properties of functions. We say that a function is L-smooth with respect to some norm $\|\cdot\|$ if it obeys $\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\|$, the dual norm of the gradient is Lipschitz continuous. It is well known in the optimization literature that when f is convex, this is equivalent to $f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2$ for $y, x \in \mathrm{dom}(f)$ and, for twice-differentiable f, $y^\top \nabla^2 f(x) y \leq L\|y\|^2$.

We say that a function is L_j-coordinate smooth in the j^{th} coordinate if the restriction of the function to the coordinate is smooth, i.e. $|\nabla_j f(x + ce_j) - \nabla_j f(x)| \leq L_j |c| \ \forall x \in \mathrm{dom}(f), c \in \mathbb{R}$. Equivalently, for twice-differentiable convex f, $\nabla^2_{jj} f(x) \leq L_j$.

Finally, we say a function is μ-strongly convex with respect to $\|\cdot\|$ if for all x, y, $f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|^2$. When f is twice-differentiable, equivalently $y^\top \nabla^2 f(x) y \geq \mu \|y\|^2$.

Graphs. We primarily study capacitated undirected graphs $G = (V, E, u)$ with edge set $E \subseteq V \times V$, edge capacities $u : E \to \mathbb{R}_+$. When referring to graphs, we let $m = |E|$ and $n = |V|$. Throughout this paper, we assume that G is strongly connected.

We associate the following matrices with the graph G, when the graph is clear from context. The matrix of edge weights $U \in \mathbb{R}^{E \times E}$ is defined as $U := \mathbf{diag}(u)$. Orienting the edges of the graph arbitrarily, the vertex-edge incidence matrix $B \in \mathbb{R}^{V \times E}$ is defined as $B_{s,(u,v)} := -1$ if $s = u$, 1 if $s = v$ and 0 otherwise.

Divergences. In the analysis of mirror descent variants, a first-order method flexible to geometric constraints on its arguments, we require the concept of a Bregman divergence with respect to a regularizer r. For a convex function r, we define the (nonnegative) Bregman divergence to be

$$V_x^r(y) = r(y) - r(x) - \nabla r(x)^\top (y - x).$$

We drop the r for convenience when it is clear from context. The Bregman divergence satisfies the well-known equality

$$\langle \nabla V_x^r(y), u - y \rangle = V_x^r(u) - V_x^r(y) - V_y^r(u). \tag{4.1}$$

4.2.2 Overview of our algorithms

Here, we give an overview of the main ideas used in our algorithms for approximately solving ℓ_∞ regression problems. The main ideological contribution of this work is that it uses a new variation of coordinate descent which uses the novel concept of *local coordinate smoothness* in order to get tighter guarantees for accelerated algorithms.

ℓ_∞ regression algorithm

The first piece of our algorithm is developed in Section 4.3.2, where we show how to use a *primal-dual proximal point method* inspired by the "conceptual mirror-prox" algorithm of [415] to reduce the task of designing an accelerated scheme for the ℓ_∞ regression problem to designing an unaccelerated procedure for minimizing a regularized approximation of the regression objective. Next, we show in Section 4.3.3 how to improve the complexity of the standard coordinate descent algorithm for an appropriately regularized ℓ_∞-smooth approximation to the regression problem by using the concept of local coordinate smoothnesses, which we introduce. To analyze its convergence, we develop a novel analysis of coordinate descent under dynamic sampling probabilities subject to a box constraint. Finally, in order to implement the steps of the algorithm, it is necessary to efficiently compute overestimates to the local coordinate smoothnesses, and furthermore sample coordinates proportional to these overestimates; this procedure is given in Lemma 51.

Acceleration via proximal point reduction. In Section 4.3.2, we show how we can reduce minimizing the original ℓ_∞ objective to efficiently finding high-precision minimizers to a sequence of regularized approximations, via a proximal scheme of [415], which we refer to as the primal-dual proximal point method, or proximal point method for short.[1] This reduction constructs a sequence of iterates by calling a high-precision minimization oracle for each regularized approximation, where the regularization amount is parameterized by a scalar quantity $\alpha > 0$. A larger α will result in simpler subproblems, but will require more calls to the oracle; trading off these complexities via the parameter α results in our accelerated runtime. More formally, note that we may rewrite the original regression problem by introducing a dual variable (after appropriately doubling the constraints to account for signs; see discussion in Section 4.3.1)

$$\min_{x\in[-1,1]^m} \|Ax-b\|_\infty = \min_{x\in[-1,1]^m} \max_{p\in\Delta^n} p^\top(Ax-b).$$

The proximal point method with parameter α constructs a sequence of points $\{z_t\}$ as follows: from an iterate $z_t = (x_t, p_t)$, define the next iterate $z_{t+1} = (x_{t+1}, p_{t+1})$ to be the solution to a proximal subproblem (here and throughout, $s := \|x^*\|_2^2$ where x^* is the optimizer of the box-constrained ℓ_∞

[1] The proximal point method in this paper is slightly different than the "conceptual mirror-prox" algorithm of [415]. In [415], each iteration takes two steps, the first of which solves a regularized proximal problem to sufficiently high accuracy, and the second of which is an extragradient adjustment step. We bypass the need for this adjustment step via more stringent requirements on the accuracy level of the solution of the proximal problem.

regression problem)

$$z_{t+1} = \text{argmin}_{x\in[-1,1]^m} \text{argmax}_{p\in\Delta^n} \; p^\top(Ax-b) + \frac{\alpha}{2s}\|x-x_t\|_2^2 - \alpha\sum_i p_i \log\frac{p_i}{[p_t]_i}. \tag{4.2}$$

To explain further, the most prevalent first-order method approach to convex optimization, and its primal-dual generalization (for example found in mirror descent and gradient descent) for solving a problem of the form $\min_{x\in[-1,1]^m}\max_{p\in\Delta^n} p^\top(Ax-b)$ with gradient operator $g(x,p)$, is to repeatedly construct regularized linearizations of the form, for some regularizer function r,

$$z_{t+1} = \text{argmin}_z \; \langle g(z_t), z\rangle + V_{z_t}^r(z).$$

The proximal method instead sets the next iterate z_{t+1} to be the result of a proximal problem, without the linearization; we set the regularizer $r(x,p)$ to be $\frac{1}{2s}\|x\|_2^2 + \sum_i p_i\log p_i$. Overall, if the regularizer r has range bounded by Θ, then the proximal point method converges in roughly $\alpha\Theta/\epsilon$ iterations to an ϵ-approximate saddle point, which suffices for our purposes.

We give the convergence analysis of the proximal point method under approximate solutions to the subproblems defining the iterates $\{z_t\}$ in Section 4.3.2. Therefore, the main algorithmic workhorse can be reduced to computing high-accuracy saddle points to problems of the form

$$\text{argmin}_{x\in[-1,1]^m} \; \alpha\log\sum_{i\in[n]}\exp\left(\frac{1}{\alpha}[Ax-b_t]_i\right) + \frac{\alpha}{2s}\|x-x_t\|_2^2. \tag{4.3}$$

Note that the problem (6.21) is the same as (4.2), where we maximized over p explicitly; the vector b_t is obtained via a linear shift of the vector b (details can be found in Section 4.3.2). Our remaining algorithmic development deals with this subproblem; combining a fast iterative method for this subproblem with the optimal choice of α yields the runtime for regression. To obtain our more fine-grained runtimes in Section 4.4, we also generalize to diagonally-reweighted ℓ_2^2 regularizers.

Local coordinate smoothness. In this work, we introduce the concept of *local coordinate smoothness* at a point x. This generalizes the concept of global coordinate smoothness to a particular point. This definition is crucial to the analysis throughout the rest of the paper.

Definition 9 (Local coordinate smoothness)**.** *Twice-differentiable function f is $L_j(x)$ locally coordinate smooth in coordinate j at x, if for all $|c| \le |\nabla_j f(x)/L_j(x)|$, $\nabla_{jj}^2 f(x+ce_j) \le L_j(x)$.*

We state a useful equivalent characterization to Definition 9; the proof is standard and follows by integration (once and twice respectively).

Lemma 42. *For twice-differentiable f, f is $L_j(x)$ locally coordinate smooth if and only if $|\nabla_j f(y) - \nabla_j f(y')| \le L_j(x)|y-y'|$ for all y, y' between $x \pm \nabla_j f(x)/L_j(x)e_j$. If f is $L_j(x)$ locally coordinate smooth then for all y between $x \pm \nabla_j f(x)/L_j(x)e_j$, $f(y) \le f(x) + \nabla f_j(x)(y_j - x_j) + \frac{L_j(x)}{2}|y_j - x_j|^2$.*

Note that this says that a coordinate descent step using local smoothnesses at a point exhibits roughly the same behavior as a single step of coordinate descent with global smoothnesses. In particular, for the point which the coordinate descent algorithm would step to, the function values exhibit the same quadratic upper bound along the coordinate. For a more motivating discussion of this definition, we refer the reader to an analysis of coordinate descent presented in Appendix C.1.5. We will drop the x from the notation $L_j(x)$ when the point we are discussing is clear, i.e. a particular iterate of one of our algorithms.

Bounding the progress of coordinate descent in ℓ_∞-smooth functions. Here, we sketch the main idea underlying our improved runtime for the problem (6.21), whose first component is ℓ_∞-smooth. Why is it possible to hope to improve gradient methods in the ℓ_∞ norm via coordinate descent? One immediate reason is that smoothness in this norm is a strong assumption on the sum S of the local coordinate smoothness values of f.

As we recall in Appendix C.1, gradient descent for an ℓ_∞-smooth function initialized at $x^0 \in \mathbb{R}^m$ takes roughly $\frac{L\|x^0-x^*\|_\infty^2}{\epsilon}$ iterations to converge to a solution which has ϵ additive error, whereas coordinate descent with appropriate sampling probabilities $\frac{L_j}{S}$, for $S = \sum_j L_j$, takes $\frac{S\|x^0-x^*\|_2^2}{\epsilon}$ iterations to converge to the same quality of solution.

When the norm in the gradient descent method is $\|\cdot\|_\infty$, we have $\|x^0 - x^*\|_2^2 \leq m\|x^0 - x^*\|_\infty^2$, but the iterates can be m times cheaper because they do not require a full gradient computation. So, if we can demonstrate $S \leq L$, we can hope to match and improve the runtime. To be more concrete, we will demonstrate the following fact.

Lemma 43. *Suppose for some point x, $f : \mathbb{R}^m \to \mathbb{R}$ is convex and L-smooth with respect to $\|\cdot\|_\infty$, $\Lambda_j(x) = \nabla_{jj}^2 f(x)$, and $S = \sum_j \Lambda_j(x)$. Then $S \leq L$.*

Proof. Fix x, and define $M := \nabla^2 f(x)$ and $S := \mathrm{Tr}(M)$. Consider drawing y uniformly at random from $\{-1, 1\}^m$. By the smoothness assumption, we have $y^\top My \leq L\|y\|_\infty^2 = L$. Also, note that

$$\mathbb{E}[y^\top My] = \mathbb{E}\left[\sum_{i,j} M_{ij}y_iy_j\right] = \mathrm{Tr}(M) = S$$

Thus, by the probabilistic method, there exists some y such that $S \leq y^\top My \leq L$, as desired. □

While this gives a bound on the number of iterations required by a coordinate descent algorithm, it requires being able to compute and sample by the $L_j(x)$; as we take coordinate descent steps, it is not clear how the local coordinate smoothnesses $L_j(x^k)$ will change, and how to update and compute them. Naively, at each iteration, we could recompute the local smoothnesses, but this requires as much work as a full gradient computation if not more. Furthermore, we need to implement sampling the coordinates in an appropriate way, and show how the algorithm behaves under acceleration. However, a key idea in our work is that if we can take steps within regions where the smoothness values do not change by much, we can still make iterates computationally cheap, which we will show.

Box-constrained coordinate descent under dynamic sampling. One technical difficulty that arises in the analysis of coordinate descent methods under local coordinate smoothnesses is the fact that the sampling distribution changes from iteration to iteration. In prior analyses of coordinate descent subject to a separable convex (i.e. box) constraint [222, 449], a key technical fact of the iterates was the fact that they could be written as a convex combination of prior iterates. Under dynamic sampling distributions, this may no longer be the case. In this work, we give a new analysis of coordinate descent under a box constraint, and show that the progress of each iteration can be directly analyzed by using the geometry of the box constraint. We develop this analysis in Section 4.3.3, and combining it with our local coordinate smoothness analysis yields the faster oracle for minimizing problem (6.21).

Implementation of local smoothness estimates. One useful property of coordinate descent is that as long as we implement the algorithm with overestimates to the local smoothness values, the convergence rate scales with the sum of the overestimates. Our full algorithm for solving (6.21) proceeds by showing how to compute and sample proportional to slight overestimates to the local smoothnesses, for regression problems in a column-sparse matrix. We do so by first proving that the smooth approximation to ℓ_∞ regression admits local smoothnesses which can be bounded in a structured way, in Section 4.3.3. Further, using a lightweight data structure, we are able to maintain these overestimates and sample by them in nearly-constant time, yielding a very efficient implementation, which we show in Section 4.3.5.

Maximum flow algorithm

In Section 4.4, we study the maximum flow problem as an example of a problem which can be reduced to ℓ_∞ regression in a column-sparse matrix. We first describe a reduction from approximate maximum flow to structured instances of ℓ_∞ regression, already-present in the literature [482, 318, 441]. We first show that a direct application of our accelerated ℓ_∞ regression algorithm yields the fastest currently known approximate maximum flow algorithm, roughly giving a runtime of $\tilde{O}(m+(n+\sqrt{ms})/\epsilon)$. We also show that a slight modification of our accelerated regression algorithm, where the norm we measure smoothness and strong-convexity of the box-constrained variable is weighted by columns of the matrix, yields a runtime of $\tilde{O}(m+\sqrt{mn}/\epsilon)$, generically improving upon the runtime of [483] for slightly-dense graphs.

Finally, in Section 4.5, we show that by opening up the algorithm further into a fully primal-dual method, we can use a novel analysis of a variance-reduced mirror prox method based on local coordinate smoothness estimates in order to obtain an improved runtime of $\tilde{O}(m+(n+\sqrt{ns})/\epsilon)$. Our randomized mirror prox method requires the development of a somewhat more-complicated data structure, based on efficient polynomial approximations to the exponential, in order to approximately query and sample from a simplex variable under dense updates.

4.3 ℓ_∞ regression subject to a box constraint

We now show how to turn the framework presented in the previous section into improved algorithms for the problem of box-constrained regression in the ℓ_∞ norm. Recall that our goal is to compute an ϵ-approximate minimizer of the constrained ℓ_∞ regression problem with a $O(1/\epsilon)$ method complexity (see Definition 8).

In the style of previous approaches to solving ℓ_∞ regression, because $\|x\|_\infty$ is not a smooth function, we choose to minimize a suitable smooth approximation instead. Intuitively, the $O(1/\epsilon)$ rate comes from accelerating gradient descent for a function which is $O(1/\epsilon)$-smooth. One would then expect the function error of the T^{th} iterate with respect to OPT is proportional to $(1/\epsilon)/T^2$, so if we wish for an ϵ-approximate minimizer, it suffices to pick $T = O(1/\epsilon)$. Because our method is not a typical accelerated method, and is instead based on reducing the proximal point method to solving a series of subproblems (6.21), the runtime analysis proceeds somewhat differently. We will show (roughly speaking) how to solve a subproblem of type (6.21) in

$$\tilde{O}\left(m + \frac{\min(m,n)}{\alpha} + \frac{s}{\alpha^2}\right)$$

iterations, where each iteration can be implemented in time $\tilde{O}(c)$, where c is the maximum number of nonzero entries in any column of A. Because each problem (6.21) results from a regularization based on a regularizer r of nearly-constant range, it suffices to solve $\tilde{O}(\alpha/\epsilon)$ such problems to yield an ϵ-approximate solution. Finally, each reduction to the subproblem is complemented by an extragradient step, which takes time $O(\text{nnz}(A))$. The accelerated runtime is then roughly

$$\tilde{O}\left(\left(\text{nnz}(A) + \frac{\min(m,n)}{\alpha} + \frac{s}{\alpha^2}\right)\frac{\alpha}{\epsilon}\right) = \tilde{O}\left(\text{nnz}(A) + \frac{\min(m,n) + \sqrt{\text{nnz}(A)s}}{\epsilon}\right),$$

where the choice of α was to appropriately balance the terms.

4.3.1 Constructing the smooth approximation to regression

In this section, we define the smooth approximation for ℓ_∞ regression we use through the paper and provide some technical facts about this approximation. Note that these approximations are standard in the literature. First, we define the smax function which is used throughout. This function is smooth in the ℓ_∞ norm, which can be seen because it is the result of the following conjugate problem

$$\max_{p \in \Delta^n} \langle p, x \rangle - \alpha \sum_i p_i \log p_i;$$

because the function $r(p) = \sum_i p_i \log p_i$ is 1-strongly convex in the ℓ_1 norm, its dual, the softmax function, is smooth in the ℓ_∞ norm.

Definition 10 (Softmax)**.** *For all real valued vectors x we let* $\mathrm{smax}_\alpha(x) := \alpha \log(\sum_j \exp(\frac{x_j}{\alpha}))$.

Fact 5 (Softmax additive error)**.** $\forall x \in \mathbb{R}^m$, $\max_{j\in[m]} x_j \leq \mathrm{smax}_\alpha(x) \leq \alpha \log m + \max_{j\in[m]} x_j$.

Proof. It follows from monotonicity of log and positivity of exp: letting j^* be the maximal index of x, $\mathrm{smax}_\alpha(x) \geq \alpha \log(\exp(x_{j^*}/\alpha)) = x_{j^*}$, and $\mathrm{smax}_\alpha(x) \leq \alpha \log(m \exp(x_{j^*}/\alpha)) = \alpha \log m + x_{j^*}$. □

Note that these properties are about the quality of approximation smax provides on the maximum element of a vector, instead of its ℓ_∞ norm. To apply this to an ℓ_∞ objective, we used the standard reduction of applying it to the regression problem in twice the original dimension, defined with a proxy matrix $A' = \begin{pmatrix} A \\ -A \end{pmatrix}$ and a proxy vector $b' = \begin{pmatrix} b \\ -b \end{pmatrix}$. For notational convenience, we will focus on minimizing $f(x)$ defined above, but with $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^n$ in the original dimensionalities, which preserves all dependencies on the dimension and structural sparsity assumptions used later in this work up to a constant. Next, we state some technical properties of our approximation. We drop the α from many definitions because the α we choose for all our methods is fixed.

Definition 11. *For $x \in \mathbb{R}^m$ let $p(x) \in \mathbb{R}^m$ be defined as $p_j(x) := \frac{\exp(x_j/\alpha)}{\sum_{j'} \exp(x_{j'}/\alpha)}$.*

Note that for any x the above $p_j(x)$ form a probability distribution. Moreover, they are defined in this way because they directly are used in the calculation of the gradient and Hessian of smax. The following facts can be verified by direct calculation.

Fact 6 (Softmax calculus)**.** $\nabla \mathrm{smax}_\alpha(x) = p(x)$, $0 \preceq \nabla^2 \mathrm{smax}_\alpha(x) \preceq \alpha^{-1}\mathbf{diag}(p(x))$.

4.3.2 Acceleration via proximal point method

In this section, we give an analysis of a proximal point method inspired by [415], tailored to our purposes. The method reduces the problem of finding an ϵ-approximate saddle point to a minimax convex-concave objective to iteratively solving a proximal subproblem to sufficiently high accuracy. Consider a saddle point problem of the form

$$\min_{x\in\mathcal{X}} \max_{p\in\mathcal{P}} f(x,p),$$

where f is convex in its restriction to the first argument and concave in its restricton to the second. Define the *duality gap* of a pair (x,p) to be

$$\max_{p'\in\mathcal{P}} f(x,p') - \min_{x'\in\mathcal{X}} f(x',p).$$

Note that when we define the associated gradient operator

$$g(x,p) := \left(\nabla_x f(x,p), -\nabla_p f(x,p)\right),$$

convexity-concavity shows we may upper bound the duality gap with respect to some pair (x', p') by the *regret* $\langle g(x,p), (x,p) - (x',p')\rangle$, in the sense of

$$f(x,p') - f(x',p) \leq \langle \nabla_x f(x,p), x - x'\rangle - \langle \nabla_p f(x,p), p - p'\rangle = \langle g(x,p), (x,p) - (x',p')\rangle .$$

The proximal point algorithm, with a possibly randomized prox oracle, defines a sequence $\{z_t\}$, where each iterate is the result of calling a proximal oracle on the previous iterate. Formally, the method is defined as follows.

Definition 12 (Primal-dual proximal point method)**.** *Initalize some $z_0 = (x_0, p_0)$, and let $q(x)$ and $r(p)$ be convex distance generating functions; let $V_z(w)$ be the Bregman divergence on the joint space with respect to their sum, i.e. for $z = (x,p)$ and $z' = (x',p')$,*

$$V_z(z') := V_x^q(x') + V_p^r(p').$$

We define the primal-dual proximal point method *to be the iteration of the following procedure: on iteration t, from the point $z_t = (x_t, p_t)$, let $z_{t+1} = (x_{t+1}, p_{t+1})$ be any point such that*

$$\max_{u \in \mathcal{X}\times\mathcal{P}} \left\{ \langle g(z_{t+1}), z_{t+1} - u\rangle - \alpha V_{z_t}(u) + \alpha V_{z_{t+1}}(u) \right\} \leq \epsilon.$$

We remark that this definition of z_{t+1} is motivated by the fact that the (exact) solution of

$$\min_{x\in\mathcal{X}} \max_{p\in\mathcal{P}} f(x,p) + \alpha V_{x_t}^q(x) - \alpha V_{p_t}^r(p) \tag{4.4}$$

has this property with $\epsilon = 0$; the conceptual prox method implies that any efficient algorithm for finding a high-precision saddle point to the prox problem suffices. Our algorithm for computing iterates will ultimately be randomized; we will union bound the probability that the iterate produced does not have the necessary property over all iterations by an inverse polynomial in n.

Lemma 44. *The iterates resulting from running the primal-dual proximal point method for T iterations satisfy, for any $u \in \mathcal{X}\times\mathcal{P}$,*

$$\frac{1}{T}\sum_{t\in[T]} \langle g(z_t), z_t - u\rangle \leq \frac{\alpha V_{z_0}(u)}{T} + \epsilon.$$

Proof. Consider some particular iterate t. By the definition of z_{t+1}, we have for all u,

$$\langle g(z_{t+1}), z_{t+1} - u\rangle \leq \alpha \left(V_{z_t}(u) - V_{z_{t+1}}(u)\right) + \epsilon.$$

Summing over all iterations, taking an average, and using nonnegativity of V yields the conclusion. □

We now specialize the required oracle for computing the $\{z_t\}$ to our particular saddle-point problem (in the case of ℓ_∞ regression). In our setting (after the constraints A have been appropriately doubled to account for sign), we wish to solve

$$\min_{x\in[-1,1]^m} \|Ax-b\|_\infty = \min_{x\in[-1,1]^m} \max_{p\in\Delta^n} p^\top (Ax-b).$$

The associated gradient operator for a point (x,p) is

$$g(x,p) = \left(A^\top p, b - Ax\right). \tag{4.5}$$

For the rest of this section, whenever we write $g(x,p)$ and the associated A, b in the regression problem are clear from context, we mean (4.5). We note that to solve the original (primal-only) regression problem, it suffices to obtain duality gap in the primal-dual regression problem with respect to (x^*, p') for any p', where $x^* = \operatorname{argmin}_{x\in[-1,1]^m} \|Ax-b\|_\infty$, as quantified in the following.

Lemma 45. *Let $z = (x,p) \in [-1,1]^m \times \Delta^n$ be a pair such that for all $u = (x^*, p')$, where $x^* \in [-1,1]^m$ is fixed and $p' \in \Delta^n$ is arbitrary,*

$$\langle g(z), z-u\rangle \le \epsilon.$$

Then, we have

$$\|Ax-b\|_\infty - \|Ax^*-b\|_\infty \le \epsilon.$$

Proof. Choose p' so that $p'^\top(Ax-b) = \|Ax-b\|_\infty$. Then,

$$\langle g(z), z-u\rangle = p^\top\left((Ax-b)-(Ax^*-b)\right) + (b-Ax)^\top(p-p') \ge \|Ax-b\|_\infty - \|Ax^*-b\|_\infty.$$

The only inequality follows from $p^\top(Ax^*-b) \le \|Ax^*-b\|_\infty$ for any $p\in\Delta^n$. □

In our definition of the proximal point method (Definition 12), we choose $q(x) = \frac{1}{2s}\|x\|_2^2$, and $r(p) = \sum_{i\in[n]} p_i \log p_i$, where $s := \|x^*\|_2^2$. It is simple to compute that from these definitions,

$$V_x^q(x') = \frac{1}{2s}\|x-x'\|_2^2,\ V_p^r(p') = \sum_{i\in[n]} p_i' \log\frac{p_i'}{p_i}.$$

Moreover, it is well-known that when $p\in\Delta^n$ is the uniform distribution $\frac{1}{n}\mathbf{1}$, the range of $V_p^r(p')$ is bounded by $\log n$. Therefore, Lemma 44 and Lemma 45 imply that we only need to take $\tilde{O}(\alpha/\epsilon)$ iterations of the proximal point method to obtain an ϵ-approximate minimizer to the regression problem. We complete the analysis of this framework by showing that in order to return a sequence $\{z_t\}$ with the necessary properties, it suffices to approximately compute the saddle point to problems of the form (4.4).

Lemma 46. *From a point $z = (x, p)$, let $\bar{z} = (\bar{x}, \bar{p})$ be the solution to the problem*

$$\operatorname{argmin}_{x' \in [-1,1]^m} \operatorname{argmax}_{p' \in \Delta^n} p'^\top (Ax' - b) + \frac{\alpha}{2s} \|x - x'\|_2^2 - \alpha \sum_{i \in [n]} p'_i \log \frac{p'_i}{p_i}.$$

Then, for $\epsilon < 1$, any x' with

$$\|x' - \bar{x}\|_\infty \leq \min\left(\frac{\epsilon}{16 \|A\|_\infty}, \frac{\epsilon s}{8\alpha m}, \frac{\epsilon \alpha}{64 \|A\|_\infty^2} \right),$$

and setting $p' \in \Delta^n$ to be

$$p' \propto \exp\left(\frac{1}{\alpha} (Ax' - b + \alpha \log p) \right),$$

letting $z' = (x', p')$, for all $u \in [-1, 1]^m \times \Delta^n$,

$$\langle g(z'), z' - u \rangle - \alpha V_z(u) + \alpha V_{z'}(u) \leq \epsilon.$$

Proof. By the optimality conditions of the definition of $\bar{z}$, we see that for all $u \in [-1, 1]^m \times \Delta^n$,

$$\langle g(\bar{z}), \bar{z} - u \rangle \leq \alpha(V_z(u) - V_{\bar{z}}(u) - V_z(\bar{z})) \leq \alpha(V_z(u) - V_{\bar{z}}(u)).$$

Therefore, it suffices to show that

$$\langle g(z') - g(\bar{z}), \bar{z} - u \rangle + \langle g(z'), z' - \bar{z} \rangle + \alpha(V_{\bar{z}}(u) - V_{z'}(u)) \leq \epsilon. \tag{4.6}$$

We first derive a simple bound on $\|\bar{p} - p'\|_1$. Note that by the definition of $\bar{p}$ as the optimal response to $\bar{x}$, we have that $A\bar{x} - b + \alpha \log(p/\bar{p})$ is a multiple of the all-ones vector, so

$$\bar{p} \propto \exp\left(\frac{1}{\alpha} (A\bar{x} - b + \alpha \log p) \right).$$

Therefore, the multiplicative ratio between each entry of p' and $\bar{p}$ is bounded by

$$\exp\left(\frac{2}{\alpha} \|A\|_\infty \|\bar{x} - x'\|_\infty \right) \leq \exp\left(\frac{\epsilon}{32 \|A\|_\infty} \right) \leq 1 + \frac{\epsilon}{16 \|A\|_\infty}. \tag{4.7}$$

This immediately implies that $\|\bar{p} - p'\|_1 \leq \epsilon \|p'\|_1 / (16 \|A\|_\infty) = \epsilon / (16 \|A\|_\infty)$. Finally, we conclude by noting that by ℓ_1-ℓ_∞ Hölder, and $\|x' - x\|_\infty \leq \epsilon / (16 \|A\|_\infty)$,

$$\langle g(\bar{z}) - g(z'), \bar{z} - u \rangle \leq 2 \|A\|_\infty \|\bar{x} - x'\|_\infty + 2 \|A\|_\infty \|\bar{p} - p'\|_1 \leq \frac{\epsilon}{4},$$
$$\langle g(z'), z' - \bar{z} \rangle \leq \|A\|_\infty \|\bar{x} - x'\|_1 + \|A\|_\infty \|\bar{p} - p'\|_1 \leq \frac{\epsilon}{4}.$$

Moreover, by using the definitions of Bregman divergences and $\|x\|_1 \leq m$ for $x \in [-1,1]^m$, and noting that similarly to the derivation of (4.7), p'/p is entrywise bounded by $\exp(\epsilon/4\alpha)$ via $\|x' - x\|_\infty \leq \epsilon/(16\|A\|_\infty)$,

$$\begin{aligned}\alpha(V_{\bar{x}}^q(u_x) - V_{x'}^q(u_x)) &= \frac{\alpha}{2s}\|\bar{x} - u_x\|_2^2 - \frac{\alpha}{2s}\|x' - u_x\|_2^2 = \frac{\alpha}{s}\langle u_x, x' - \bar{x}\rangle + \frac{\alpha}{2s}\langle x' + \bar{x}, x' - \bar{x}\rangle \\ &\leq \frac{\alpha}{s}\left(\|u_x\|_1 + \frac{1}{2}\|x' + \bar{x}\|_1\right)\|x' - \bar{x}\|_\infty \leq \frac{2\alpha m}{s}\|x' - \bar{x}\|_\infty \leq \frac{\epsilon}{4}, \\ \alpha(V_{\bar{p}}^r(u_p) - V_{p'}^r(u_p)) &= \alpha\sum_{i\in[n]}[u_p]_i \log\frac{p_i'}{\bar{p}_i} \leq \alpha\max_{i\in[n]}\log\frac{p_i'}{\bar{p}_i} \leq \frac{\epsilon}{4}.\end{aligned}$$

Finally, (4.6) follows by combining the above bounds. □

Finally, note that for $\bar{z} = (\bar{x}, \bar{p})$ the solution to the problem

$$\text{argmin}_{x'\in[-1,1]^m}\text{argmax}_{p'\in\Delta^n} p'^\top(Ax' - b) + \frac{\alpha}{2s}\|x - x'\|_2^2 - \alpha\sum_{i\in[n]} p_i'\log\frac{p_i'}{p_i},$$

we can equivalently write that $\bar{x}$ is the solution to the problem

$$\text{argmin}_{x\in[-1,1]^m}\ \alpha\log\sum_{i\in[n]}\exp\left(\frac{1}{\alpha}\left[Ax - \tilde{b}\right]_i\right) + \frac{\alpha}{2s}\|x' - x\|_2^2,\ \tilde{b} := b - \alpha\log p.$$

We will show in the following section how to efficiently compute an approximate minimizer to this problem with high probability.

4.3.3 Constructing the subproblem oracle

In this section, we develop a new analysis of (unaccelerated) coordinate descent under local coordinate smoothness estimates and a box constraint, and show how to use it to compute a high-accuracy solution to the subproblems required by our proximal point method. More specifically, we develop an efficient iterative method for solving the problem (abusing some notation for simplicity of this self-contained section)

$$\text{argmin}_{x\in[-1,1]^m}\ \alpha\log\sum_{i\in[n]}\exp\left(\frac{1}{\alpha}\left[Ax - b\right]_i\right) + \frac{\alpha}{2s}\|x - \bar{x}\|_2^2. \tag{4.8}$$

Box-constrained coordinate descent under dynamic sampling

In this section, we first develop a general coordinate descent analysis under a box constraint, amenable to dynamic sampling probabilities. Let $\mathcal{X}$ be an arbitrary box, e.g. product of one-dimensional intervals, and let f be an ℓ_2 μ-strongly convex function. Suppose at each point x, we

have local coordinate smoothness estimates $\{L_j(x)\}$ such that for

$$x' = \text{argmin}_{x' \in \mathcal{X}} \left\{ f(x) + \langle \nabla_j f(x), x' - x \rangle + \frac{L_j(x)}{2} \|x' - x\|_2^2 \right\},$$

we have that the upper bound (recalling Definition 9) holds, e.g.

$$f(x') \leq f(x) + \langle \nabla_j f(x), x' - x \rangle + \frac{L_j(x)}{2} \|x' - x\|_2^2 .$$

Further, define

$$S(x) = \sum_{j \in [m]} L_j(x),$$

and assume that there is a global upper bound S on $S(x)$. Consider the following "local smoothness" variant of the standard coordinate descent algorithm.

Definition 13 (Locally smooth coordinate descent)**.** *Given a function f with local coordinate smoothnesses $\{L_j(x)\}$ at each point x, define the local smoothness coordinate descent algorithm as iteratively performing the following (resetting $x' \leftarrow x$) every iteration:*

1. *Sample $j \propto L_j(x)$.*
2. *Update $x' \leftarrow \text{argmin}_{x' \in \mathcal{X}} \left\{ f(x) + \langle \nabla_j f(x), x' - x \rangle + \frac{L_j(x)}{2} \|x' - x\|_2^2 \right\}$.*

We will now prove a bound on its multiplicative progress in a single iteration.

Lemma 47.

$$\mathbb{E}[f(x')] - f(x^*) \leq \left(1 - \frac{\mu}{2S}\right) (f(x) - f(x^*)).$$

Proof. First, define

$$\text{Prog}^\downarrow := f(x) - \min_{x^\downarrow \in \mathcal{X}} \left\{ f(x) + \left\langle \nabla f(x), x^\downarrow - x \right\rangle + \frac{\mu}{2} \left\| x^\downarrow - x \right\|_2^2 \right\},$$
$$x^\downarrow := \text{argmin}_{x^\downarrow \in \mathcal{X}} \left\{ f(x) + \left\langle \nabla f(x), x^\downarrow - x \right\rangle + \frac{\mu}{2} \left\| x^\downarrow - x \right\|_2^2 \right\}.$$

We have by strong convexity that $f(x) - f(x^*) \leq \text{Prog}^\downarrow$. We also define $g^\downarrow := x - x^\downarrow$, and note that $g^\downarrow$ agrees with $\nabla f(x)$ in the sign of each coordinate. Further, by separability of the box constraint,

$$0 \leq |g_j^\downarrow| \leq \frac{1}{\mu} |\nabla_j f(x)|, \ \forall j \in [m]. \tag{4.9}$$

We can explicitly write that

$$\text{Prog}^\downarrow = \sum_{j \in [m]} \text{Prog}_j^\downarrow, \text{ where } \text{Prog}_j^\downarrow := g_j^\downarrow \left(\nabla_j f(x) - \frac{\mu}{2} g_j^\downarrow \right).$$

Similarly, we define for each $j \in [m]$,

$$\text{Prog}_j^\uparrow := f(x) - \min_{x' \in \mathcal{X}} \left\{ f(x) + \langle \nabla_j f(x), x' - x \rangle + \frac{L_j(x)}{2} \|x' - x\|_2^2 \right\}.$$

We let $g^\uparrow$ be the vector such that $g_j^\uparrow$ agrees with $[x - x']_j$ if coordinate j was sampled. In particular, $g^\uparrow$ agrees with $\nabla f(x)$ in the sign of each coordinate and by separability $\forall j \in [m]$,

$$0 \leq |g_j^\uparrow| \leq \frac{1}{L_j(x)} |\nabla_j f(x)|. \tag{4.10}$$

We can explicitly write

$$\text{Prog}_j^\uparrow = g_j^\uparrow \left(\nabla_j f(x) - \frac{L_j(x)}{2} g_j^\uparrow \right).$$

First, we claim that for each $j \in [m]$,

$$\text{Prog}_j^\uparrow \geq \frac{\mu}{2L_j(x)} \text{Prog}_j^\downarrow. \tag{4.11}$$

Note that if coordinate j was sampled, and x'_j is on the boundary of $\mathcal{X}$, then $g_j^\uparrow = g_j^\downarrow$, since the minimization problem defining $g^\downarrow$ involves a larger step size. Conversely, if $[x^\downarrow]_j$ is not on the boundary of $\mathcal{X}$, then neither is x'_j, and the upper bounds of (4.9), (4.10) are tight. In both these cases and the third where $[x^\downarrow]_j$ is on the boundary and x'_j is not, the following inequality holds:

$$|g_j^\uparrow| \geq \frac{\mu}{L_j(x)} |g_j^\downarrow|.$$

We further note that

$$\left| \nabla_j f(x) - \frac{L_j(x)}{2} g_j^\uparrow \right| \geq \frac{1}{2} |\nabla_j f(x)| \geq \frac{1}{2} \left| \nabla_j f(x) - \frac{\mu}{2} g_j^\downarrow \right|.$$

Combining these two facts with the definitions of $\text{Prog}_j^\uparrow$, $\text{Prog}_j^\downarrow$ shows (4.11). Now, we have

$$\begin{aligned}
\mathbb{E}[f(x')] &\leq f(x) - \mathbb{E}[\text{Prog}_j^\uparrow] \\
&= f(x) - \sum_{j \in [m]} \frac{L_j(x)}{2S(x)} \text{Prog}_j^\uparrow \\
&\leq f(x) - \sum_{j \in [m]} \frac{\mu}{2S(x)} \text{Prog}_j^\downarrow \\
&= f(x) - \frac{\mu}{2S} \text{Prog}^\downarrow.
\end{aligned}$$

Subtracting $f(x^*)$ from both sides and using the lower bound on $\text{Prog}^\downarrow$ gives the result. □

By iteratively applying Lemma 47, and Markov's inequality, we have the following corollary.

Corollary 7. *Box-constrained locally smooth coordinate descent initialized at x_0, applied to an ℓ_2 μ-strongly convex function f converges to an ϵ-approximate minimizer with probability at least $1-\delta$ in*

$$O\left(\frac{S}{\mu}\log\left(\frac{f(x_0)-f(x^*)}{\epsilon\delta}\right)\right) \text{ iterations.}$$

We also remark that this analysis generalizes easily to strong convexity in any diagonal norm given by a (nonnegative) diagonal matrix D. In particular, let f be μ-strongly-convex in the D norm, where $D = \mathbf{diag}\,((\,)\,d)$ is some (positive) diagonal matrix, and assume we have the local coordinate smoothness bounds $\{L_j(x)\}_{j\in[m]}$ (note that the smoothness bound is still in the ℓ_2 norm, i.e. independent of the strong convexity measurement matrix). We briefly discuss how to modify the guarantee of Lemma 47. The algorithm is given as follows.

Definition 14 (Local smoothness coordinate descent in a diagonal norm)**.** *Given a function f with local coordinate smoothnesses $\{L_j(x)\}$ at each point x, define the local smoothness coordinate descent algorithm in the D norm as iteratively performing the following (resetting $x' \leftarrow x$) every iteration:*

1. *Sample $j \propto \kappa_j(x) := \frac{L_j(x)}{d_j}$.*
2. *Update $x' \leftarrow \text{argmin}_{x'\in\mathcal{X}}\left\{f(x) + \langle\nabla_j f(x), x'-x\rangle + \frac{L_j(x)}{2}\|x'-x\|_2^2\right\}$.*

We also define $S(x) = \sum_{j\in[m]}\kappa_j(x)$, and let S be a global upper bound. We modify the definitions

$$\begin{aligned}
\text{Prog}^\downarrow &:= f(x) - \min_{x_*\in\mathcal{X}}\left\{f(x) + \langle\nabla f(x), x_*-x\rangle + \frac{\mu}{2}\|x_*-x\|_D^2\right\},\\
x_* &:= \text{argmin}_{x_*\in\mathcal{X}}\left\{f(x) + \langle\nabla f(x), x_*-x\rangle + \frac{\mu}{2}\|x_*-x\|_D^2\right\},\\
g^\downarrow &:= x - x_*.
\end{aligned}$$

We also clearly have by the same argument that

$$0 \le |g_j^\downarrow| \le \frac{1}{\mu d_j}|\nabla_j f(x)|.$$

Therefore, the same arguments allow us to conclude that for each $j\in[m]$,

$$\text{Prog}_j^\uparrow \ge \frac{\mu d_j}{2L_j(x)}\text{Prog}_j^\downarrow, \text{ where we recall } \text{Prog}_j^\uparrow = g_j^\uparrow\left(\nabla_j f(x) - \frac{L_j(x)}{2}g_j^\uparrow\right).$$

Finally, our given sampling probabilities imply that we have the desired

$$\mathbb{E}[f(x')] - f(x^*) \le \left(1-\frac{\mu}{2S}\right)(f(x)-f(x^*)).$$

This yields the following corollary.

Corollary 8. *Box-constrained local smoothness coordinate descent in the diagonal D norm initialized at x_0, applied to a μ-strongly convex function f in the D norm, converges to an ϵ-approximate minimizer with probability at least $1-\delta$ in*

$$O\left(\frac{S}{\mu}\log\left(\frac{f(x_0)-f(x^*)}{\epsilon\delta}\right)\right) \text{ iterations.}$$

Minimizing the regularized softmax objective

We now use the developments of the prior section to obtain the runtime of an efficient oracle for solving (4.8) to high precision; we restate the objective here:

$$h(x) := \alpha\log\sum_{i\in[n]}\exp\left(\frac{1}{\alpha}\left[Ax-b\right]_i\right)+\frac{\alpha}{2s}\left\|x-\bar{x}\right\|_2^2.$$

The complexity of minimizing this objective function using the box-constrained coordinate descent under local coordinate smoothnesses follows from estimates given in the following lemma.

Lemma 48 (Local coordinate smoothnesses of regularized softmax)**.** *At a point $x\in[-1,1]^m$, and for all $j\in[m]$, define*

$$L_j(x)=\frac{8}{\alpha}\left\|A_{:j}\right\|_\infty\left(\langle|A_{:j}|,p(x)\rangle+\frac{2\alpha}{s}\right)+\frac{\alpha}{s}.$$

Then, h is $L_j(x)$ locally-coordinate smooth at x for all $j\in[m]$.

Proof. Recalling Definition 9, we prove the following: for $y=x+\gamma e_j$, $\gamma\in\left[\pm\frac{1}{L_j(x)}|\nabla_j h(x)|\right]$,

$$\nabla^2_{jj}h(y)\le L_j(x). \tag{4.12}$$

Defining $p(x)\propto\exp((Ax-b)/\alpha)$, by Fact 6, $\nabla^2_{jj}h(y)\le\frac{1}{\alpha}\left\|A_{:j}\right\|^2_{p(y)}+\frac{\alpha}{s}$. Therefore, it clearly suffices to show that $p(y)\le 8p(x)$ entrywise. Note that as long as we show that entrywise

$$\exp\left(\frac{Ay-b}{\alpha}\right)\in\left[\frac{1}{e},e\right]\exp\left(\frac{Ax-b}{\alpha}\right),$$

we have the conclusion by $e^2<8$. Now, using the bound on γ, this is equivalent to showing for all i that $|\langle A_{i:},x-y\rangle|\le\alpha$. Recalling $\nabla_j h(x)=\langle A_{:j},p(x)\rangle+\frac{\alpha}{s}(x_j-\bar{x}_j)$, the following suffices:

$$|A_{ij}|\left|\frac{\nabla_j h(x)}{L_j(x)}\right|=\left|\frac{|A_{ij}|\left(\langle A_{:j},p(x)\rangle+\frac{\alpha}{s}(x_j-\bar{x}_j)\right)}{\frac{8}{\alpha}\left\|A_{:j}\right\|_\infty\left(\langle|A_{:j}|,p(x)\rangle+\frac{2\alpha}{s}\right)+\frac{\alpha}{s}}\right|\le\alpha.$$

The conclusion follows. □

Minimizing the diagonally regularized softmax objective

By a simple modification of the regularizer $q(x)$ used in the proximal point method, we show how to obtain improved smoothness parameters in the regime $n < m$, independent of the sparsity of the optimal solution. In particular, for $D = \mathbf{diag}\left(() \left\{\|A_{:j}\|_\infty\right\}\right)$, the diagonal matrix whose entries are the $\{\|A_{:j}\|_\infty\}$, consider running the mirror prox procedure with the regularizer $q(x) = \frac{1}{2n\|A\|_\infty} \|x\|_D^2$; the range of $q(x)$ over the box $[-1,1]^m$ is clearly at most a constant, since the sum of (absolute values of) entries of A is bounded by $\tilde{O}(n \|A\|_\infty)$. Therefore, it suffices to design an efficient iterative method for, in the vein of (4.8), solving subproblems

$$h_d(x) := \alpha \log \sum_{i \in [n]} \exp\left(\frac{1}{\alpha} [Ax - b]_i\right) + \frac{\alpha}{2n \|A\|_\infty} \|x - \bar{x}\|_D^2 . \tag{4.13}$$

In lieu of Lemma 48, we have the following local smoothness bounds on this subproblem.

Lemma 49 (Local coordinate smoothnesses of diagonally regularized softmax)**.** *Let d be the vector whose entries are $\|A_{:j}\|_\infty$ such that $D = \mathbf{diag}\left(() d\right)$. At a point $x \in [-1,1]^m$, and for all $j \in [m]$, define*

$$L_j(x) = \frac{8}{\alpha} \|A_{:j}\|_\infty \left(\langle |A_{:j}|, p(x) \rangle + \frac{2\alpha}{n \|A\|_\infty} \|A_{:j}\|_\infty\right) + \frac{\alpha}{n \|A\|_\infty} \|A_{:j}\|_\infty .$$

Then, h_d is $L_j(x)$ locally-coordinate smooth at x for all $j \in [m]$.

Proof. The proof is similar to that of Lemma 48. Recalling Definition 9, for $y = x + \gamma e_j$, $\gamma \in \left[\pm \frac{1}{L_j(x)} |\nabla_j h_d(x)|\right]$, we wish to show

$$\nabla^2_{jj} h_d(y) \leq L_j(x). \tag{4.14}$$

Defining $p(x) \propto \exp((Ax - b)/\alpha)$, by Fact 6, $\nabla^2_{jj} h_d(y) \leq \frac{1}{\alpha} \|A_{:j}\|^2_{p(y)} + \frac{\alpha}{n\|A\|_\infty} \|A_{:j}\|_\infty$. Therefore, it clearly suffices to show that $p(y) \leq 8p(x)$ entrywise. It suffices to show that for all i that $|\langle A_{i:}, x - y \rangle| \leq \alpha$, or recalling the definition of $\nabla_j h(x) = \langle A_{:j}, p(x) \rangle + \frac{\alpha}{n\|A\|_\infty} \|A_{:j}\|_\infty (x_j - \bar{x}_j)$,

$$|A_{ij}| \left|\frac{\nabla_j h(x)}{L_j(x)}\right| = \left|\frac{|A_{ij}| \left(\langle A_{:j}, p(x) \rangle + \frac{\alpha}{n\|A\|_\infty} \|A_{:j}\|_\infty (x_j - \bar{x}_j)\right)}{\frac{8}{\alpha} \|A_{:j}\|_\infty \left(\langle |A_{:j}|, p(x) \rangle + \frac{2\alpha}{n\|A\|_\infty} \|A_{:j}\|_\infty\right) + \frac{\alpha}{n\|A\|_\infty} \|A_{:j}\|_\infty}\right| \leq \alpha.$$

The conclusion follows. □

4.3.4 Putting it all together: accelerated ℓ_∞ regression

We now state our main runtime result for ℓ_∞ regression. We combine previous developments to bound the number of coordinate descent iterations needed under local coordinate smoothness estimates needed to find an ϵ-approximate minimizer to the box-constrained ℓ_∞ regression problem

(Definition 8). We remark that the theorem statement assumes access to query and sampling oracles for the local coordinate smoothnesses; we show how to design efficient oracles for column-sparse A in Section 4.3.5. The combination of the following two theorems formally show Theorem 15.

Theorem 19 (Coordinate acceleration for ℓ_∞ regression)**.** *The proximal point method (Definition 12) with regularizers $q(x) = \frac{1}{2s}\|x\|_2^2$ and $r(p) = \sum_{i\in[n]} p_i \log p_i$, with each iterate defined by the local smoothness coordinate descent method (Definition 13) applied to the appropriate subproblem, results (with high probability) in an ϵ-approximate minimizer to the box-constrained ℓ_∞ regression problem in time*

$$\tilde{O}\left(\left(\left(\frac{s\|A\|_\infty^2}{\alpha^2} + \frac{\min(m,n)\|A\|_\infty}{\alpha} + m\right)\cdot \mathcal{T}_{iter} + \mathrm{nnz}(A)\right)\cdot\frac{\alpha}{\epsilon}\right),$$

where $\mathcal{T}_{iter}$ is the cost of sampling proportional to $L_j(x)$ and computing the value of $L_j(x)$ for an iterate x of local smoothness coordinate descent. For $\alpha = \max(\epsilon, \sqrt{s/m}\|A\|_\infty)$ and $\mathcal{T}_{iter} = O(c\log n)$ from Section 4.3.5, where c is the column sparsity of A, the runtime is

$$\tilde{O}\left(mc + \frac{(\min(m,n) + \sqrt{ms})\, c\|A\|_\infty}{\epsilon}\right).$$

Proof. We first discuss the complexity of returning an iterate of the proximal point method. Lemma 46, and the discussion following, imply that for a function of form (4.8) with optimal argument x_t^*, it suffices to find any point x' with $\|x' - x_t^*\|_\infty$ bounded by an inverse polynomial in parameters $\|A\|_\infty, m, s, \epsilon^{-1}, \alpha^{-1}$ to implement the proximal point method. By Fact 5, the range of the function (where the linear term b_t is appropriately shifted)

$$h(x) = \alpha\log\sum_{i\in[n]}\exp\left(\frac{1}{\alpha}[Ax - b_t]_i\right) + \frac{\alpha}{2s}\|x - \bar{x}\|_2^2$$

is at most $\alpha\log n + 2\|A\|_\infty + \frac{\alpha m}{2s}$, where the second term comes from the range of $\|Ax - b\|_\infty$ over $[-1,1]^m$, and the third from a simple bound $\|x - \bar{x}\|_2^2 \le m$ for all $x \in [-1,1]^m$. Moreover, strong-convexity of $h(x)$ in the ℓ_2 norm, and optimality of x_t^*, yields

$$h(x') - h(x_t^*) \ge \langle \nabla h(x_t^*), x' - x_t^*\rangle + \frac{\alpha}{2s}\|x' - x_t^*\|_2^2 \ge \frac{\alpha}{2s}\|x' - x_t^*\|_2^2,$$

which implies

$$\|x' - x_t^*\|_\infty \le \|x' - x_t^*\|_2 \le \sqrt{\frac{2s\,(h(x') - h(x_t^*))}{\alpha}}. \tag{4.15}$$

Note that for any point x, we can define an upper bound on the sum of values $L_j(x)$ in Lemma 48,

$$S := \frac{8}{\alpha}\|A\|_\infty^2 + \frac{16\min(m,n)}{s}\|A\|_\infty + \frac{m\alpha}{s} \geq \sum_{j\in[m]} \frac{8}{\alpha}\|A_{:j}\|_\infty \left(\langle |A_{:j}|, p(x)\rangle + \frac{2\alpha}{s}\right) + \frac{\alpha}{s}. \quad (4.16)$$

Here, we used that the sum of entries in the matrix is at most $n\|A\|_\infty$, and the largest entry in any column is at most $\|A\|_\infty$. Then, by Corollary 7 with strong convexity parameter $\mu = \alpha/s$, we see that a sufficient x' may be found with high probability in time

$$\tilde{O}\left(\left(\frac{s\|A\|_\infty^2}{\alpha^2} + \frac{\min(m,n)\|A\|_\infty}{\alpha} + m\right)\cdot \mathcal{T}_{\text{iter}}\right).$$

Finally, due to our choice of the regularizers q and r in the proximal point method and Lemma 44, $\widetilde{O}(\alpha/\epsilon)$ iterations of proximal point suffice, for any $\alpha \geq \epsilon$. Combining these bounds yields the first runtime claim. To see the second, we simplified using $\text{nnz}(A) \leq mc$. □

Before we give our result for accelerating ℓ_∞ regression via a diagonal norm regularization, we state a technical result on the degree of accuracy required by solutions of the proximal point subproblems, the analog of Lemma 46 in the diagonal norm (throughout, $D := \mathbf{diag}(\{\|A_{:j}\|_\infty\})$).

Lemma 50. *From a point $z = (x,p)$, let $\bar{z} = (\bar{x},\bar{p})$ be the solution to the problem*

$$\text{argmin}_{x'\in[-1,1]^m}\text{argmax}_{p'\in\Delta^n} p'^\top(Ax'-b) + \frac{\alpha}{2n\|A\|_\infty}\|x-x'\|_D^2 - \alpha\sum_{i\in[n]} p_i' \log\frac{p_i'}{p_i}.$$

Then, for $\epsilon < 1$, any x' with

$$\|x'-\bar{x}\|_\infty \leq \min\left(\frac{\epsilon}{16\|A\|_\infty}, \frac{\epsilon n}{8\alpha m}, \frac{\epsilon\alpha}{64\|A\|_\infty^2}\right),$$

and setting $p' \in \Delta^n$ to be

$$p' \propto \exp\left(\frac{1}{\alpha}(Ax'-b+\alpha\log p)\right),$$

letting $z' = (x',p')$, for all $u \in [-1,1]^m \times \Delta^n$, and where divergences are with respect to $q(x) := \frac{1}{2n\|A\|_\infty}\|x\|_D^2$ and $r(p) = \sum_{i\in[n]} p_i\log p_i$,

$$\langle g(z'), z'-u\rangle - \alpha V_z(u) + \alpha V_{z'}(u) \leq \epsilon.$$

Proof. By the optimality conditions of the definition of $\bar{z}$, we see that for all $u \in [-1,1]^m \times \Delta^n$,

$$\langle g(\bar{z}), \bar{z}-u\rangle \leq \alpha(V_z(u) - V_{\bar{z}}(u) - V_z(\bar{z})) \leq \alpha(V_z(u) - V_{\bar{z}}(u)).$$

Therefore, it suffices to show that

$$\langle g(z') - g(\bar{z}), \bar{z} - u\rangle + \langle g(z'), z' - \bar{z}\rangle + \alpha(V_{\bar{z}}(u) - V_{z'}(u)) \le \epsilon. \tag{4.17}$$

By exactly the same logic as in Lemma 46, we have the multiplicative ratio between every entry of p' and $\bar{p}$ is bounded by $1 + \epsilon/(16\|A\|_\infty)$, and $\|\bar{p} - p'\|_1 \le \epsilon/(16\|A\|_\infty)$. Finally, we conclude by noting that by ℓ_1-ℓ_∞ Hölder, and $\|x' - x\|_\infty \le \epsilon/(16\|A\|_\infty)$,

$$\langle g(\bar{z}) - g(z'), \bar{z} - u\rangle \le 2\|A\|_\infty \|\bar{x} - x'\|_\infty + 2\|A\|_\infty \|\bar{p} - p'\|_1 \le \frac{\epsilon}{4},$$
$$\langle g(z'), z' - \bar{z}\rangle \le \|A\|_\infty \|\bar{x} - x'\|_1 + \|A\|_\infty \|\bar{p} - p'\|_1 \le \frac{\epsilon}{4}.$$

Moreover, by using the definitions of Bregman divergences, $\|x\|_1 \le m$ for $x \in [-1, 1]^m$, p'/p is entrywise bounded by $\exp(\epsilon/4\alpha)$, and $\|A\|_\infty$ is larger than every entry of D,

$$\begin{aligned}
\alpha(V^q_{\bar{x}}(u_x) - V^q_{x'}(u_x)) &= \frac{\alpha}{2n\|A\|_\infty}\|\bar{x} - u_x\|^2_D - \frac{\alpha}{2n\|A\|_\infty}\|x' - u_x\|^2_D \\
&= \frac{\alpha}{n\|A\|_\infty} u_x D(x' - \bar{x}) + \frac{\alpha}{2n\|A\|_\infty}(x' + \bar{x})D(x' - \bar{x}) \\
&\le \frac{\alpha}{n}\left(\|u_x\|_1 + \frac{1}{2}\|x' + \bar{x}\|_1\right)\|x' - \bar{x}\|_\infty \le \frac{2\alpha m}{n}\|x' - \bar{x}\|_\infty \le \frac{\epsilon}{4}, \\
\alpha(V^r_{\bar{p}}(u_p) - V^r_{p'}(u_p)) &= \alpha\sum_{i\in[n]} [u_p]_i \log\frac{p'_i}{\bar{p}_i} \le \alpha \max_{i\in[n]} \log\frac{p'_i}{\bar{p}_i} \le \frac{\epsilon}{4}.
\end{aligned}$$

Finally, (4.17) follows by combining the above bounds. □

Theorem 20 (Coordinate acceleration for ℓ_∞ regression in a diagonal norm)**.** *The proximal point method (Definition 12) with regularizers $q(x) := \frac{1}{2n\|A\|_\infty}\|x\|^2_D$ for $D = \boldsymbol{diag}(\{\|A_{:j}\|_\infty\})$ and $r(p) = \sum_{i\in[n]} p_i \log p_i$, with each iterate defined by the local smoothness coordinate descent method in the D norm (Definition 14) applied to the appropriate subproblem, results (with high probability) in an ϵ-approximate minimizer to the box-constrained ℓ_∞ regression problem in time*

$$\tilde{O}\left(\left(\left(\frac{n\|A\|^2_\infty}{\alpha^2} + \frac{n\|A\|_\infty}{\alpha} + m\right)\cdot \mathcal{T}_{iter} + \mathrm{nnz}(A)\right)\cdot\frac{\alpha}{\epsilon}\right),$$

where $\mathcal{T}_{iter}$ is the cost of sampling proportional to $L_j(x)/d_j$ and computing the value of $L_j(x)$ for an iterate x of local smoothness coordinate descent. For $\alpha = \max(\epsilon, \sqrt{n/m}\,\|A\|_\infty)$ and $\mathcal{T}_{iter} = O(c\log n)$ from Section 4.3.5, where c is the column sparsity of A, the runtime is

$$\tilde{O}\left(mc + \frac{(n + \sqrt{mn})\,c\,\|A\|_\infty}{\epsilon}\right).$$

Proof. We first note that without loss of generality, every entry of D is at least ϵ/m; indeed, adding

ϵ/m to an arbitrary nonzero entry of each column only perturbs the value of $\|Ax-b\|_\infty$ over $[-1,1]^m$ by an additive ϵ. Thus, in lieu of (4.15) in the proof of Theorem 19, it suffices to solve to a degree of accuracy polynomially larger in problem parameters, where we use that the objective is $1/(n\|A\|_\infty)$-strongly convex in the D norm, and the D norm is at most ϵ/m times smaller than the ℓ_2 norm (Lemma 50 bounds the accuracy we require in our subproblem solutions in ℓ_∞).

We next bound the sum of local smoothnesses (relative to d_j) induced by Lemma 49, and the resulting complexity of solving the subproblems induced by the proximal point method with a diagonal regularizer; the remainder of the proof follows identically from Theorem 19. Note that

$$\sum_{j\in[m]}\frac{L_j(x)}{d_j}=\sum_{j\in[m]}\frac{8}{\alpha}\left(\langle|A_{:j}|,p(x)\rangle+\frac{2\alpha}{n\|A\|_\infty}\|A_{:j}\|_\infty\right)+\frac{\alpha}{n\|A\|_\infty}=O\left(\frac{\|A\|_\infty}{\alpha}+1+\frac{m\alpha}{n\|A\|_\infty}\right).$$

The strong convexity parameter of the induced subproblems in the diagonal norm, of the form (4.13), is $\alpha/(n\|A\|_\infty)$. Thus, applying Corollary 8 implies each iterate of the mirror prox method can be found with high probability in time

$$\tilde{O}\left(\left(\frac{n\|A\|_\infty^2}{\alpha^2}+\frac{n\|A\|_\infty}{\alpha}+m\right)\cdot\mathcal{T}_{\text{iter}}\right).$$

Finally, due to the choice of regularizers, the domain size is still $\tilde{O}(1)$, so $\tilde{O}(\alpha/\epsilon)$ iterations of proximal point suffice, giving the first claim; the second claim follows by choice of α. □

4.3.5 Cheap iterations for ℓ_∞ regression in column-sparse matrices

In this section, we show how to attain cheap iterations for A whose columns have bounded sparsity. In particular, suppose A is c-column-sparse. We show how to, for the local coordinate smoothness estimates

$$L_j(x)=\frac{8}{\alpha}\|A_{:j}\|_\infty\left(\langle|A_{:j}|,p(x)\rangle+\frac{2\alpha}{s}\right)+\frac{\alpha}{s} \tag{4.18}$$

defined in Lemma 48, implement maintenance of the $L_j(x)$ and sampling by the quantities $L_j(x)$ for each iteration of the local smoothness coordinate descent procedure applied to the problem (4.8), in time $\mathcal{T}_{\text{iter}}=O(c\log n)$. This shows that the runtime of the efficient implementation of our algorithm is, up to a $\tilde{O}(c)$ multiplicative factor, the same as the iteration count; in particular, for $c=\tilde{O}(1)$, we are able to implement each step in $\tilde{O}(1)$ time, without affecting the number of iterations by more than a $\tilde{O}(1)$ factor. More formally, in this section we show the following.

Lemma 51 (Efficient implementation of iterates)**.** *Suppose we implement local smoothness coordinate descent (Definition 13) for the problem* (4.8) *for some c-column-sparse A. Then, with* $\text{nnz}(A)$ *precomputation cost, throughout the lifetime of the algorithm for local coordinate smoothness estimates $L_j(x)$* (4.18) *where x is an iterate, it is possible to (1) maintain the sum $\sum_{j\in[m]}L_j(x)$, (2)*

compute for any j the value $L_j(x)$, and (3) sample from the distribution $\{p_j \propto L_j(x)\}$ in time $O(c \log n)$ per iteration.

Proof. We will describe the $L_j(x)$ maintenance and sampling procedures separately.

Maintaining smoothness overestimates.

We first show how to (implicitly) maintain the quantities

$$p_i(x) = \frac{\exp(\frac{1}{\alpha}[Ax-b]_i)}{\sum_{i' \in [n]} \exp(\frac{1}{\alpha}[Ax-b]_{i'})}$$

in $O(c)$ time per iteration. In particular, because each iteration of (local smoothness) coordinate descent, starting at x and stepping to x', only affects a single coordinate, and by column-sparsity this only affects at most c of the values $\exp(\frac{1}{\alpha}[Ax-b]_i)$, we can maintain their sum in $O(c)$ time, and also maintain the vector $\exp(\frac{1}{\alpha}(Ax-b))$.

Next, we discuss how to maintain $\sum_{j \in [m]} L_j(x)$ and query any $L_j(x)$ in $O(c)$ time per iteration. In $O(\text{nnz}(A))$ time we precompute and store all values

$$\frac{16 \left\|A_{:j}\right\|_\infty}{s} + \frac{\alpha}{s},$$

and there are at most c entries in $A_{:j}$, so querying $L_j(x)$ can be performed in $O(c)$ time, because we can compute any entry of $p(x)$ using the stored $\exp(\frac{1}{\alpha}(Ax-b))$ and its maintained sum. Moreover, in computing the sum

$$\sum_{j \in [m]} L_j(x) = \left(\sum_{j \in [m]} \frac{16 \left\|A_{:j}\right\|_\infty}{s} + \frac{m\alpha}{s} \right) + \left(\frac{8}{\alpha} \sum_{i \in [n]} p_i(x) \sum_{j \in [m]} |A_{ij}| \left\|A_{:j}\right\|_\infty \right),$$

all quantities other than the $p_i(x)$ can be precomputed; the second summand can be computed with respect to the unnormalized vector $\exp(\frac{1}{\alpha}(Ax-b))$, and then scaled uniformly using its sum.

Sampling from the distribution.

In this part of the proof, we describe how to implement sampling from the distribution proportional to $L_j(x)$. First, in the prior discussion note that we maintain the sum of the $L_j(x)$ by computing the values of the two summands

$$\sum_{j \in [m]} \frac{16 \left\|A_{:j}\right\|_\infty}{s} + \frac{m\alpha}{s}, \; \frac{8}{\alpha} \sum_{i \in [n]} p_i(x) \sum_{j \in [m]} |A_{ij}| \left\|A_{:j}\right\|_\infty.$$

We first flip an appropriately biased coin to choose a summand. If the first is selected, then we sample a coordinate $j \in [m]$ with probability proportional to

$$\frac{16 \left\|A_{:j}\right\|_\infty}{s} + \frac{\alpha}{s};$$

this can be done in constant time via precomputation [528].

To sample from the second summand, it clearly suffices to first sample the rows of A by a distribution proportional to $p(x)$, and then sample the indices of that row proportional to $|A_{ij}| \|A_{:j}\|_\infty$, the latter of which takes constant time via precomputation [528]. To sample the rows, we use the well-known strategy that it suffices to maintain an augmented binary search tree data structure whose leaves dynamically maintain the set of $\exp(\frac{1}{\alpha}[Ax-b]_i)$ for the current iterate x. As previously argued, each iteration changes only c of these values, so maintaining the augmented binary search tree takes $O(c \log n)$ per iteration.

□

4.4 Accelerating maximum flow

The primary goal of this section is to show how to use the development of Section 4.3, tailored to the regression problem associated with maximum flow, and give tighter analyses on its runtime guarantees to demonstrate how it yields faster algorithms. The reduction to ℓ_∞ regression is the same as introduced in [482], and is included for completeness.

4.4.1 Maximum flow preliminaries

The maximum flow problem is defined as follows: given a graph, and two of its vertices s and t labeled as source and sink, find a flow $f \in \mathbb{R}^m$ which satisfies the capacity constraints such that the discrete divergence at the sink, $(Bf)_t$, is as large as possible, and $(Bf)_s = -(Bf)_t$, $(Bf)_v = 0$ for $v \neq s, t$.

Following the framework of [482], we consider instead the equivalent problem of finding a minimum congestion flow; intuitively, if we route 1 unit of flow from s to t and congest edges as little as possible, we can find the maximum flow by just taking the multiple of the minimum congestion flow which just saturates edges. The congestion incurred by a flow f is $\left\|U^{-1}f\right\|_\infty$ where U is the diagonal matrix of edge capacities, and we say f routes demands d if $Bf = d$. The problem of finding a minimum congestion flow for a given demand vector, and its dual, the maximum congested cut, can be formulated as follows:

$$\begin{aligned} &\min_f. \quad \|U^{-1}f\|_\infty \quad \text{s.t.} \quad Bf = d, f \geq 0. \\ &\max_v. \quad d^\top v \qquad\quad \text{s.t.} \quad \|UB^\top v\|_1 \leq 1. \end{aligned} \tag{4.19}$$

Let $d_S := \sum_{u \in S} d_u$ and $c(S,T)$ denote the total weight of edges from S to T. It is well-known that for the second problem, one of the threshold cuts with respect to v achieves $d_S / c(S, V-S) \geq d^\top v$. Whenever the flow problem is clear from context, we will refer to any optimal flow by f^{OPT}.

4.4.2 From maximum flow to constrained ℓ_∞ regression

First, we show how to transform the maximum flow problem into a constrained regression problem. The key tool used here is the concept of a good *congestion approximator* [482], and associated properties.

Definition 15 (Congestion approximator)**.** *An α-congestion approximator for G is a matrix R such that for any demand vector d, $\|Rd\|_\infty \leq \mathrm{OPT}_d \leq \alpha \|Rd\|_\infty$.*

For undirected graphs, it is known that $\tilde{O}(1)$-congestion approximators can be computed in nearly linear time [376, 482, 318, 441]. Further, the certain variants of these congestion approximator have additional nice properties. We use the following construction from [441].

Theorem 21 (Summary of results in [441])**.** *There is an algorithm which given an m-edge n-vertex undirected graph runs in time $\tilde{O}(m)$ and with high probability produces an α-congestion approximator R, for $\alpha = \tilde{O}(1)$. Furthermore, the matrix $A := 2\alpha RBU$ has the following properties: (1) each column of A has at most $\tilde{O}(1)$ nonzero entries, (2) $\|A\|_\infty = \tilde{O}(1)$, (3) A has $O(n)$ rows, and (4) A can be computed in time $\tilde{O}(m)$.*

The above theorem is the result of a construction in [441]. Properties 2, 3, and 4 are direct results of the construction given in the paper (where 3 follows from the fact that the congestion approximator comes from routing on a graph which is a tree). Property 1 results from the way in which the tree is constructed, such that the depth of the congestion-approximating tree is $\tilde{O}(1)$, so each edge in the original graph G is only routed onto a polylogarithmic number of edges.

Our analysis of reducing the flow problem to the regression problem follows that of [482]. In particular, the reduction is given as follows.

Lemma 52. *Let G be an undirected graph and d be a demand vector. Assume we are given an α-congestion approximator R, and the associated matrix $A = 2\alpha RBU$. Furthermore, let $2\alpha Rd := b$. In order to multiplicatively approximately solve the maximum flow problem given by Equation (4.19), it suffices to solve an associated box-constrained regression problem $\|Ax - b\|_\infty$ over $x \in [-1, 1]^m$ a nearly-constant number of times to an ϵ-additive approximation, and pay an additional $\tilde{O}(m)$ cost, which under the change of variables $x := U^{-1}f$ recovers a corresponding flow. We call the full algorithm* Flow-To-Regress*.*

In particular, we are able to use R from the statement of Theorem 21. For completeness, we will prove Lemma 52 in the appendices, but on a first read one may skip the proof and use the reduction statement as a black box result for the remaining analysis.

4.4.3 Runtimes for accelerated maximum flow

Here, we provide a full description of how to implement relevant machinery for applying the tools from Section 4.3 for accelerating the minimization of a constrained ℓ_∞ function to the regression

problem given in Lemma 52. Due to the arguments presented in Appendix C.2, it suffices to bound the runtime of approximately solving the initial regression problem.

Definition 16 (Flow regression problem)**.** *The maximum flow regression problem asks to ϵ-approximately minimize the function $\|Ax - b\|_\infty$ subject to $x \in [-1, 1]^m$, $\|b\|_\infty \leq 1$, and for $\tilde{O}(1)$-column-sparse A with $\|A\|_\infty = \tilde{O}(1)$.*

Lemma 52 implies that the cost of finding an ϵ-approximate maximum flow is (up to logarithmic factors) the same as solving the flow regression problem once.

Applications of Section 4.3

We first show how to use the methods of Section 4.3 to obtain an improved maximum flow algorithm. First, note that by applying Theorem 19 directly, combining with the properties given in Theorem 21 of the regression matrix A, we immediately obtain a runtime of $\tilde{O}\left(m + (n + \sqrt{ms})/\epsilon\right)$ for the maximum flow problem, where the additive factor $\tilde{O}(m)$ comes from the preprocessing required in Section 4.3.5, as well as the cost of computing the matrix A. Here, we used $\min(m, n) = n$ in the case of the flow regression matrix. We further can apply Theorem 20 to obtain a runtime of $\tilde{O}\left(m + \sqrt{mn}/\epsilon\right)$, where the dominant term is $\sqrt{mn}$ as $m = \Omega(n)$. Taking the better of these runtimes implies the following.

Theorem 22. *There is an algorithm that takes time $\tilde{O}(m + (n + \sqrt{m \min(n, s)})/\epsilon)$ to find, with high probability, an ϵ-approximate maximum flow, where s is the squared ℓ_2 norm of the congestion vector of any optimal flow.*

Tighter runtime dependence

We develop an algorithm with an improved runtime for the flow regression problem in Section 4.5, based on directly applying a randomized mirror prox method to the primal-dual regression objective. Its runtime guarantee is stated here, and its full details are given in Section 4.5.

Theorem 23. *There is an algorithm, initialized at x_0, for finding an ϵ-approximate minimizer to the flow regression problem (Definition 16), with high probability, in time $\tilde{O}(m + \max(n, \sqrt{ns})/\epsilon)$, where $s = \|x_0 - x^*\|_2^2$.*

By combining this improved algorithm with the reduction procedure of Lemma 52, we obtain our fastest algorithm for maximum flow, generically improving upon Theorem 22.

Theorem 24. *There is an algorithm that takes time $\tilde{O}(m + \max(n, \sqrt{ns})/\epsilon)$ to find, with high probability, an ϵ-approximate maximum flow, where s is the squared ℓ_2 norm of the congestion vector of any optimal flow.*

4.4.4 Exact maximum flows in uncapacitated graphs

Here, we describe several corollaries of our approach, for rounding to an exact maximum flow for several types of uncapacitated graphs. In an uncapacitated graph, $s = \|f^*\|_2^2 \leq Fn$ where F is the maximum flow value, because the maximum flow is a 0-1 flow, and thus can be decomposed into F $s-t$ paths with length at most n. We assume here that all the graphs are simple, and thus $m \leq n^2$; it is not difficult to generalize these results to non-simple graphs. As preliminaries, we state the following standard techniques for rounding to exact maximum flows.

Lemma 53 (Theorem 5 in [343])**.** *There is a randomized algorithm that runs in expected time $\tilde{O}(m)$ which takes a fractional flow of value F on an uncapacitated graph, and returns an integral flow of value $\lfloor F \rfloor$.*

We will thus always assume that we have applied the rounding to an integral flow as a preprocessing step, as it will not affect our asymptotic runtime.

Lemma 54 (Augmenting paths)**.** *There is an algorithm that runs in time $O(m)$ which takes a non-maximal integral flow of value F on an uncapacitated graph, and returns an integral flow of value $F+1$.*

Suppose we have a flow with value $(1-\epsilon)F$, where the maximum flow value is F. The two lemmas for rounding and augmenting a flow therefore imply that the additional runtime required to attain an exact maximum flow is $O(\epsilon F m)$.

Undirected uncapacitated graphs

We state several corollaries of Theorem 24 which apply to finding exact maximum flows in various types of undirected uncapacitated graphs. All of these results only hold with high probability.

Corollary 9 (Undirected graphs)**.** *There is an algorithm which finds a maximum flow in an undirected, uncapacitated graph in time $\tilde{O}(m^{5/4}n^{1/4})$.*

Proof. We run the algorithm from Theorem 24 for $\epsilon = n^{1/4}/m^{3/4}$, and then run augmenting paths for $O(\epsilon m^2)$ iterations. Note that the maximum flow value and sparsity are bounded by m, and thus this will yield a maximum flow. Furthermore the runtime of the approximate algorithm is bounded by $m + \sqrt{nm}/\epsilon$. Putting together these two runtimes yields the result. □

Corollary 10 (Undirected graphs with small maximum flow value)**.** *There is an algorithm which finds a maximum flow in an undirected, uncapacitated graph with maximum flow value F in time $\tilde{O}(m + \min(\sqrt{mn}F^{3/4}, m^{3/4}n^{1/4}\sqrt{F}))$.*

Proof. The analysis here is the same as in Corollary 9, but instead we note that the bound on s is $\min(m, Fn)$, where the latter factor results from combining F paths of length at most n. If

the better bound is Fn, our runtime is bounded by $\tilde{O}(m + (n + \sqrt{n^2F})/\epsilon + \epsilon Fm)$, and choosing $\epsilon = n^{1/2}/(F^{1/4}m^{1/2})$ yields the result. If the better bound is m, our runtime is bounded by $\tilde{O}(m + \sqrt{nm}/\epsilon + \epsilon Fm)$, and choosing $\epsilon = n^{1/4}/(\sqrt{F}m^{1/4})$ yields the result. □

Corollary 11 (Undirected graphs with sparse optimal flow)**.** *There is an algorithm which finds a maximum flow in an undirected, uncapacitated graph with a maximum flow that uses at most s edges in time $\tilde{O}(m + \sqrt{ms}n^{1/4}\max(n, s)^{1/4})$.*

Proof. The analysis here is the same as in Corollary 9, but instead we note that the bound on the maximum flow value is also s. Thus, our runtime is bounded by $\tilde{O}(m + \max(n, \sqrt{ns})/\epsilon + \epsilon sm)$. If $n \leq s$, choosing $\epsilon = n^{1/4}/(s^{1/4}m^{1/2})$ yields the result; otherwise, we choose $\epsilon = \sqrt{n/ms}$. □

Directed graphs

We follow the standard reduction of finding a maximum flow in a directed graph to finding a maximum flow in an undirected graph described in, for example, [363]. In short, an undirected graph with maximum flow value $O(m)$ is created, such that we can initialize the algorithm in Theorem 24 at a flow which is off from the true maximum flow by s in ℓ_2^2 distance. We give this reduction in Appendix C.2.2, and refer the reader to [363] for a more detailed exposition.

Thus, after applying this reduction, the only difference in the runtimes given by the previous section are that the rounding algorithm will always take time $O(\epsilon m^2)$ instead of $O(\epsilon Fm)$. This immediately yields the following runtimes for exact maximum flows in directed graphs.

Corollary 12 (Directed graphs)**.** *There is an algorithm which finds a maximum flow in a directed, uncapacitated graph in time $\tilde{O}(m^{5/4}n^{1/4})$.*

Corollary 13 (Directed graphs with a sparse optimal flow)**.** *There is an algorithm which finds a maximum flow in a directed, uncapacitated graph in time $\tilde{O}(mn^{1/4}\max(n, s)^{1/4})$.*

4.5 Improved flow runtimes via primal-dual coordinate regression

In this section, we prove Theorem 23 by giving the algorithm and analyzing its runtime. Throughout, as in the statement of the flow regression problem (Definition 16), $A \in \mathbb{R}^{n\times m}$ has $\tilde{O}(1)$-sparse columns, $\|A\|_\infty \leq 1$, and $\|b\|_\infty \leq 1$, where we drop logarithmic factors in $\|A\|_\infty$ for simplicity. We describe how to obtain a point $\hat{x}$ with $\|\hat{x}\|_\infty \leq 1$, and

$$\|A\hat{x} - b\|_\infty - \epsilon \leq \text{OPT} := \|Ax^* - b\|_\infty, \text{ where } x^* := \text{argmin}_{x|\|x\|_\infty \leq 1} \|Ax - b\|_\infty.$$

The runtime we will prove for the algorithm (initialized at the origin) is, as in Theorem 24,

$$\tilde{O}\left(m + \frac{n + \sqrt{ns}}{\epsilon}\right), \text{ where } s := \|x^*\|_2^2.$$

Note if we wish to supply the algorithm with an initial point which is not the origin, as is the case for our results on maximum flow in directed graphs, it suffices to modify the definition of b appropriately and shift by the initial point (see Appendix C.1.2 for a more formal treatment).

4.5.1 Overview

We first give an outline of our algorithm. The main motivation for the form it takes is to obtain the "best of both worlds" runtime of the form $\sqrt{ns}/\epsilon$. In terms of the dependence of Theorem 19 on s, i.e. the sparsity of the optimal point, a standard (unweighted) Euclidean regularizer is necessary for the primal point $x \in [-1, 1]^m$. In terms of the dependence of Theorem 20 trading off an n factor for an m, we require more fine-grained estimates on local coordinate smoothnesses based on dual information and properties of the matrix. We obtain both of these improvements in our final runtime via a fully primal-dual coordinate regression algorithm.

Throughout, all divergences on x space are with respect to $q(x) = \frac{1}{2s}\|x\|_2^2$, on y space[2] are with respect to $r(y) = \sum_i y_i \log y_i$, and on the product space are with respect to the direct sum (we drop superscripts in definitions of Bregman divergences in this section, as the regularizer will be fixed).

Regularized subproblem. The first step of our method is to define the following function, a regularized variant of the primal-dual formulation of the box-constrained $\|Ax - b\|_\infty$ objective:

$$h(x, y) := y^\top (Ax - b) + \frac{\epsilon}{2} q(x) - \frac{\epsilon}{4 \log n} r(y). \tag{4.20}$$

Throughout, we refer to the saddle point to the regularized objective h by $\tilde{x} \in [-1, 1]^m, \tilde{y} \in \Delta^n$. The motivation for considering the regularized problem is related to technical issues which arise when generalizing Lemma 44 to interact with a randomized algorithm; as we will see, returning the average iterate is computationally expensive for our coordinate method. We bypass this by providing a last-iterate guarantee via regularization, by arguing we can repeatedly return a point in each phase halving the distance to the saddle point.

The following lemma shows that to solve the box-constrained ℓ_∞ regression problem, it suffices to solve the regularized problem

$$\min_{x \in [-1,1]^m} \max_{y \in \Delta^n} h(x, y)$$

to high accuracy. We also show that the regularized optimizer's ℓ_2 sparsity is not too large.

Lemma 55. *$\|\tilde{x}\|_2^2 \le 2s$, and $\|A\tilde{x} - b\|_\infty \le \mathrm{OPT} + \frac{\epsilon}{2}$.*

[2] In this section, we use y rather than p to denote dual points, as they evolve separately; in our previous algorithms, p was typically a probability distribution induced by a primal point x.

Proof. Recall that the definition of $\mathrm{smax}_\alpha(x)$ implies

$$\mathrm{smax}_\alpha(x) = \max_{y \in \Delta^n} y^\top (Ax - b) - \alpha r(y).$$

By Fact 5,

$$\|Ax - b\|_\infty \leq \mathrm{smax}_{\epsilon/4\log n}(x) \leq \|Ax - b\|_\infty + \frac{\epsilon}{4}.$$

Correspondingly, we have the following chain of inequalities:

$$h(\tilde{x}, \tilde{y}) \leq h(x^*, \tilde{y}) \leq \mathrm{smax}_{\epsilon/4\log n}(x^*) + \frac{\epsilon}{2} q(x^*) \leq \mathrm{OPT} + \frac{\epsilon}{4} + \frac{\epsilon}{4} = \mathrm{OPT} + \frac{\epsilon}{2}.$$

The first inequality follows from minimality of $\tilde{x}$ with respect to $\tilde{y}$, the second from considering the terms in h corresponding to y, and the last by the definition of OPT and s. Now, we also have

$$h(\tilde{x}, \tilde{y}) = \mathrm{smax}_{\epsilon/4\log n}(\tilde{x}) + \frac{\epsilon}{2} q(\tilde{x}) \geq \|A\tilde{x} - b\|_\infty + \frac{\epsilon}{4s} \|\tilde{x}\|_2^2 .$$

Putting these together and using $\|A\tilde{x} - b\|_\infty \geq \mathrm{OPT}$ by definition,

$$\mathrm{OPT} + \frac{\epsilon}{4s} \|\tilde{x}\|_2^2 \leq \mathrm{OPT} + \frac{\epsilon}{2} \Rightarrow \|\tilde{x}\|_2^2 \leq 2s.$$

Similarly, the other conclusion follows by nonnegativity of $\frac{\epsilon}{4s} \|\tilde{x}\|_2^2$. □

Consequently, an algorithm which is capable of obtaining a high-accuracy saddle point to h suffices for minimizing the original objective.

Randomized mirror prox method. We now describe one phase of our algorithm, which takes an initial point $z_{k-1,0} = (x_{k-1,0}, y_{k-1,0})$, and returns a point $z_{k,0} = (x_{k,0}, y_{k,0})$ with

$$\mathbb{E}[V_{z_{k,0}}(\tilde{x}, \tilde{y})] \leq \frac{1}{2} V_{z_{k-1,0}}(\tilde{x}, \tilde{y}). \tag{4.21}$$

Here, the expectation is over randomness used in the k^{th} phase, i.e. the randomness used to define the point $z_{k,0}$. Combining this recursive guarantee via iterating expectations with the following initial bound (which uses Lemma 55) gives us a logarithmic bound on the number of phases.

Lemma 56. *Let $x_{0,0}$ be the all-zeroes vector and $y_{0,0} = \frac{1}{n}\mathbf{1}$. Then, $V_{x_{0,0}, y_{0,0}}(\tilde{x}, \tilde{y}) \leq \Theta_0 := 1 + \log n$.*

In order to obtain the guarantee (4.21), our starting point is Nemirovski's *mirror prox* method [415], which can be viewed as a fixed-point iteration approximating the proximal point method (Definition 12). Note that optimality conditions imply that iterating (4.4) in the proximal point method produces a sequence of iterates satisfying

$$z_{t+1} \leftarrow \mathrm{argmin}_z \{\langle g(z_{t+1}), z\rangle + \alpha V_{z_t}(z).\}$$

However, this method is not implementable, as z_{t+1} uses its own gradient operator in its definition. Nemirovski's mirror prox approximates this process via a fixed-point iteration, by defining a two-step sequence

$$w_t \leftarrow \text{argmin}_w \left\{ \frac{1}{\kappa} \langle g(z_t), w \rangle + V_{z_t}(w) \right\}, \; z_{t+1} \leftarrow \text{argmin}_z \left\{ \frac{1}{\kappa} \langle g(w_t), z \rangle + V_{z_t}(z) \right\}. \tag{4.22}$$

Here, the parameter κ must be chosen to meet certain criteria so that the fixed-point iteration provably converges to a sufficient quality, and also governs the iteration count. Typically, κ depends on the strong convexity of the regularizers q and r, which leads to a dimension dependence in the runtime in the case of ℓ_∞ regression. [483] bypassed this by identifying a weaker criteria for the sequence (4.22) to converge. We obtain further improvements via a randomized variation of (4.22).

Note that the gradient operator of the problem (4.20) is:

$$g(x, y) := \left(A^\top y + \frac{\epsilon}{2s} x, b - Ax + \frac{\epsilon}{4 \log n} \log y \right).$$

A natural attempt at an unbiased estimator for g, inspired by the algorithm of Section 4.3, is (for some sampling probabilities $\{p_j\}$) to randomly sample a coordinate of the primal block of g, i.e.

$$g_j(x, y) := \left(\frac{1}{p_j} \left(A_{:j}^\top y + \frac{\epsilon}{2s} x_j \right) e_j, b - Ax + \frac{\epsilon}{4 \log n} \log y \right), \tag{4.23}$$

We would then define a step by: sample $j \sim \{p_j\}$, then iterate

$$w_t \leftarrow \text{argmin}_w \left\{ \frac{1}{\kappa} \langle g_j(z_t), w \rangle + V_{z_t}(w) \right\}, \; z_{t+1} \leftarrow \text{argmin}_z \left\{ \frac{1}{\kappa} \langle g_j(w_t), z \rangle + V_{z_t}(z) \right\}.$$

However, in order to obtain our tight runtimes by leveraging a primal-dual analog of local coordinate smoothnesses, we require "sharing randomness" between these iterates, i.e. using the same coordinate j in both steps. Note that in doing so, it no longer makes sense to say that $g_j(w_t)$ is an unbiased estimator for $g(w_t)$, as the choice of j was used in the definition of w_t. We bypass this by defining an "aggregate point" $\bar{w}_t$ which $g_j(w_t)$ *is* unbiased for, over the randomness of w_t. We remark that this design strategy is a somewhat bespoke application of developments in Section 2.6, tailored to our purposes.

We then use a tight characterization of the convergence of our randomized method via local coordinate smoothnesses to argue about the quality of the average iterates $\bar{w}_t$, and show that randomly sampling one over $\widetilde{O}(m + (n + \sqrt{ns})/\epsilon)$ iterations halves the divergence to $(\tilde{x}, \tilde{y})$ in expectation. For this last step, we use the strong monotonicity[3] of the objective h to convert regret bounds into divergence bounds. Our complete algorithm concludes by repeating this procedure for $\widetilde{O}(1)$ phases.

Roadmap. Section 4.5.2 states the algorithm, a randomized variation of mirror prox which uses

[3]Strong monotonicity is a primal-dual analog of strong convexity.

the local coordinate smoothness ideas developed in Section 4.3 in its analysis. It first develops a one-phase analysis, which leverages strong monotonicity of the objective h in order to halve the distance to the true saddle point $(\tilde{x}, \tilde{y})$ in $\widetilde{O}(m + \max(n, \sqrt{ns})/\epsilon)$ iterations constituting a phase. It then uses the output of each phase as the starting point for the next phase, culminating in a high-accuracy saddle point in a logarithmic number of phases.

A key technical hurdle is that the iterates of the algorithm no longer have the sparse update structure used in the data structure development of Section 4.3.5. In Section 4.5.3, we show how to carefully use the structure of the updates to design a data structure based around Taylor approximation to perform iterations in batches, using nearly-constant amoritized time per iteration. We remark that the data structure we develop in this section is in some sense generalized by the data structure used for ℓ_1-ℓ_1 games in Chapter 3.

4.5.2 Algorithm

Throughout, we index phases of the algorithm by $k \in [K]$, and iterates within a phase by $t \in [T]$. As discussed in the overview, we will choose $T = \widetilde{O}(m + (n + \sqrt{ns})/\epsilon)$, and $K = \widetilde{O}(1)$.

Section 4.5.2 defines local coordinate smoothness quantities which will factor into the algorithm. Section 4.5.2 gives an analysis of a single phase of the algorithm, which outputs a point with expected divergence halved from the phase input. At the end of this section, we give a complete implementation of the phase, where we highlight issues with inexact implementation (which will be treated formally in Section 4.5.3). Section 4.5.2 leverages this single-phase method to give the complete algorithm, and proves the final runtime guarantee.

Preliminaries

We first define some parameters used in the algorithm. For any $y \in \Delta^n$ we define for all j,

$$L_j(y) := s \left\|A_{:j}\right\|_\infty \left|A_{:j}\right|^\top y + \epsilon \left\|A_{:j}\right\|_\infty,$$
$$\tilde{L}_j(y) := \left(\sqrt{s \left\|A_{:j}\right\|_\infty} \sum_{i \in [n]} \sqrt{|A_{ij}| y_i} + \sqrt{\epsilon \left\|A_{:j}\right\|_\infty}\right)^2.$$

where $|A_{:j}|$ is element-wise. These quantities will serve the role of local coordinate smoothness estimates in our algorithm and analysis. It is immediate that for all j,

$$\tilde{O}(1) L_j(y) \geq \tilde{L}_j(y) \geq L_j(y), \tag{4.24}$$

where the $\tilde{O}(1)$ factor is due to Cauchy-Schwarz and that each $A_{:j}$ has $\tilde{O}(1)$ non-zero entries.

Lemma 57. *For any y, $\sum_j \sqrt{\tilde{L}_j(y)} \leq C\sqrt{ns} + \sqrt{mn\epsilon}$, for some $C = \tilde{O}(1)$.*

Proof. Let all columns of A have at most $c = \tilde{O}(1)$ nonzero entries. Then,

$$\begin{aligned}\left(\sum_{j\in[m]}\sum_{i\in[n]}\sqrt{\|A_{:j}\|_\infty y_i|A_{ij}|}\right)^2 &= \left(\sum_{j\in[m]}\sqrt{\|A_{:j}\|_\infty}\cdot\left[\sqrt{c}\cdot\sqrt{\sum_{i\in[n]}y_i|A_{ij}|}\right]\right)^2 \\ &\le c\left(\sum_{j\in[m]}\|A_{:j}\|_\infty\right)\left(\sum_{j\in[m]}\sum_{i\in[n]}y_i|A_{ij}|\right) \\ &\le nc\left(\sum_{i\in[n]}y_i\|A_i\|_1\right)\le nc.\end{aligned} \tag{4.25}$$

Here, the first line follows from Cauchy-Schwarz and using the fact that the sum $\sum_{i\in[n]}\sqrt{y_i|A_{ij}|}$ is c-sparse, the second line follows from Cauchy-Schwarz again, and the third line follows from the assumption $\|A\|_\infty \le 1$. Thus,

$$\begin{aligned}\sum_{j\in[m]}\sqrt{\tilde{L}_j(y)} &= \sqrt{s}\sum_{j\in[m]}\sum_{i\in[n]}\sqrt{\|A_{:j}\|_\infty|A_{ij}|y_i}+\sum_{j\in[m]}\sqrt{\epsilon\|A_{:j}\|_\infty} \\ &\le \sqrt{nsc}+\sqrt{\epsilon}\sqrt{m\sum_{j\in[m]}\|A_{:j}\|_\infty}\le\sqrt{nsc}+\sqrt{mn\epsilon}.\end{aligned}$$

It suffices to choose $C=\sqrt{c}$, where we used the column sparsity assumption. □

Finally, we define the following sampling distribution at any point y:

$$\begin{aligned}p_j(y) = \frac{C\sqrt{ns}}{C\sqrt{ns}+\sqrt{mn\epsilon}}\cdot\frac{\sqrt{s\|A_{:j}\|_\infty}\sum_i\sqrt{|A_{ij}|y_i}}{\sum_j\sqrt{s\|A_{:j}\|_\infty}\sum_i\sqrt{|A_{ij}|y_i}}+\frac{\sqrt{mn\epsilon}}{C\sqrt{ns}+\sqrt{mn\epsilon}}\cdot\frac{\sqrt{\epsilon\|A_{:j}\|_\infty}}{\sum_j\sqrt{\epsilon\|A_{:j}\|_\infty}} \\ \ge\frac{\sqrt{\tilde{L}_j(y)}}{C\sqrt{ns}+\sqrt{mn\epsilon}}.\end{aligned} \tag{4.26}$$

The last inequality follows from the bounds from Lemma 57,

$$\sum_j\sqrt{s\|A_{:j}\|_\infty}\sum_i\sqrt{|A_{ij}|y_i}\le C\sqrt{ns},\ \sum_j\sqrt{\epsilon\|A_{:j}\|_\infty}\le\sqrt{mn\epsilon}.$$

We also make the simplifying assumption that at any y, all the sampling probabilities $p_j(y)$ are at least $1/(2m)$. To see why this is a valid assumption, our algorithm ultimately has a runtime depending linearly on our bound on $\sum_{j\in[m]}\sqrt{\tilde{L}_j(y)}$. By treating each $\sqrt{\tilde{L}_j(y)}$ as its sum with the average square root coordinate smoothness, this only doubles the overall sum (and therefore the bound in Lemma 57), but enforces the lower bound on the sampling probabilities. This can be always be implemented by uniform sampling with half probability.

Single phase analysis

In this section we give an analysis of the k^{th} phase. We drop subscript k from all iterates for simplicity, within the context of this section, until the very end. We define

$$\kappa := m\epsilon + 8\sqrt{mn\epsilon} + 8C\sqrt{ns} + 16n,$$

the parameter which will ultimately govern the iteration count of the phase. We briefly discuss where each summand comes from in the analysis.

1. The factor of $m\epsilon$ is used to account for terms of the form $\frac{\epsilon}{2p_j}[x_t]_j$ showing up in the randomized gradient estimator, which can be as large as $m\epsilon$, in Lemma 62. This is the key lemma used to bound the progress of a single iteration.

2. The factor of $8\sqrt{mn\epsilon}$ is used to ensure the stability of the simplex variable in a single iteration, due to the effects of terms of the form $\frac{\epsilon}{2p_j}[x_t]_j$, in Lemmas 59 and 60. It is never the leading-order term, due to the terms $m\epsilon$ and $16n$.

3. The factor of $8C\sqrt{ns}$ is used for both the error analysis and stability, due to effects of terms of the form $\frac{1}{p_j}A\Delta_t^{(j)}$, in Lemmas 59, 60, and 62.

4. The factor of $16n$ is used to guarantee that $\kappa = \Omega(n)$. This is necessary in bounding the movement due to a fixed, dense term in the gradient updates, in the runtime analysis of Section 4.5.3. In particular, it ensures we do not have to restart the data structure for simplex variable maintenance too frequently.

We now give one iteration of the phase, starting at a point $z_t = (x_t, y_t)$.

1. Sample $j \propto p_j(y_t)$
2. $x_{t+\frac{1}{2}}^{(j)} \leftarrow \text{argmin}_{x\in[-1,1]^m}\left\{\left\langle \frac{1}{\kappa p_j}(A_{:j}^\top y_t + \frac{\epsilon}{2s}[x_t]_j)e_j, x\right\rangle + V_{x_t}(x)\right\}$.
3. $y_{t+\frac{1}{2}} \leftarrow \text{argmin}_{y\in\Delta^n}\left\{\left\langle \frac{1}{\kappa}\left(b - Ax_t + \frac{\epsilon}{4\log n}\log y_t\right), y\right\rangle + V_{y_t}(y)\right\}$.
4. $\Delta_t^{(j)} := x_{t+\frac{1}{2}}^{(j)} - x_t$.
5. $x_{t+1}^{(j)} \leftarrow \text{argmin}_{x\in[-1,1]^m}\left\{\left\langle \frac{1}{\kappa p_j}(A_{:j}^\top y_{t+\frac{1}{2}} + \frac{\epsilon}{2s}[x_{t+\frac{1}{2}}^{(j)}]_j)e_j, x\right\rangle + V_{x_t}(x)\right\}$.
6. $y_{t+1}^{(j)} \leftarrow \text{argmin}_{y\in\Delta^n}\left\{\left\langle \frac{1}{\kappa}\left(b - A\left(x_t + \frac{1}{p_j}\Delta_t^{(j)}\right) + \frac{\epsilon}{4\log n}\log y_{t+\frac{1}{2}}\right), y\right\rangle + V_{y_t}(y)\right\}$.

We remark that in all but possibly the j^{th} coordinate, $x_{t+\frac{1}{2}}^{(j)}$ and $x_{t+1}^{(j)}$ are identical to x_t. We write

$$z_t = (x_t, y_t),\ w_t^{(j)} = \left(x_{t+\frac{1}{2}}^{(j)}, y_{t+\frac{1}{2}}\right),\ z_{t+1}^{(j)} = \left(x_{t+1}^{(j)}, y_{t+1}^{(j)}\right).$$

We briefly remark on the form of the iterates. The gradient estimators inducing the points $x^{(j)}_{t+\frac{1}{2}}$, $x^{(j)}_{t+1}$ are precisely those described by (4.23), where we note that the point $y_{t+\frac{1}{2}}$ is deterministic (conditioned on z_t). Moreover, the gradient estimator inducing $y^{(j)}_{t+1}$ is chosen so that our algorithm has the following property, which implies in each iteration, there is an "aggregate point" $\bar{w}_t$ whose regret we can bound. In this sense, the term $\frac{1}{p_j}\Delta^{(j)}_t$ can be viewed as a debiasing step.

Lemma 58. *Let $\bar{x}_{t+\frac{1}{2}} = x_t + \sum_j \Delta^{(j)}_t$, the point taking all coordinate steps from x_t, and denote*

$$g_j(w^{(j)}_t) = \left(\frac{1}{p_j}\left(A^\top_{:j} y_{t+\frac{1}{2}} + \frac{\epsilon}{2s}\left[x^{(j)}_{t+\frac{1}{2}}\right]_j\right) e_j,\ b - A\left(x_t + \frac{1}{p_j}\Delta^{(j)}_t\right) + \frac{\epsilon}{4\log n}\log y_{t+\frac{1}{2}}\right).$$

Then, we have for $\tilde{z} = (\tilde{x}, \tilde{y})$,

$$\mathbb{E}_j\left[\left\langle g_j(w^{(j)}_t), w^{(j)}_t - \tilde{z}\right\rangle\right] = \left\langle g(\bar{w}_t), \bar{w}_t - \tilde{z}\right\rangle,$$

where $\bar{w}_t = (\bar{x}_{t+\frac{1}{2}}, y_{t+\frac{1}{2}})$.

Proof. Recall that $x^{(j)}_{t+\frac{1}{2}}$ and $\bar{x}_{t+\frac{1}{2}}$ agree in the j^{th} coordinate. Then, expanding we have

$$\begin{aligned}
&\mathbb{E}_j\left[\left\langle g_j(w^{(j)}_t), w^{(j)}_t - \tilde{z}\right\rangle\right] \\
&= \sum_j p_j\left(\left\langle \frac{1}{p_j} A^\top_{:j} y_{t+\frac{1}{2}}, \bar{x}_{t+\frac{1}{2}} - \tilde{x}\right\rangle + \left\langle \frac{1}{p_j}\frac{\epsilon}{2s}[\bar{x}_{t+\frac{1}{2}}]_j, \bar{x}_{t+\frac{1}{2}} - \tilde{x}\right\rangle\right. \\
&\left. + \left\langle b - A\left(x_t + \frac{1}{p_j}\Delta^{(j)}_t\right), y_{t+\frac{1}{2}} - \tilde{y}\right\rangle + \left\langle \frac{\epsilon}{4\log n}\log y_{t+\frac{1}{2}}, y_{t+\frac{1}{2}} - \tilde{y}\right\rangle\right) \\
&= \left\langle A^\top y_{t+\frac{1}{2}}, \bar{x}_{t+\frac{1}{2}} - \tilde{x}\right\rangle + \left\langle \frac{\epsilon}{2s}\bar{x}_{t+\frac{1}{2}}, \bar{x}_{t+\frac{1}{2}} - \tilde{x}\right\rangle \\
&+ \left\langle b - A\bar{x}_{t+\frac{1}{2}}, y_{t+\frac{1}{2}} - \tilde{y}\right\rangle + \left\langle \frac{\epsilon}{4\log n}\log y_{t+\frac{1}{2}}, y_{t+\frac{1}{2}} - \tilde{y}\right\rangle \\
&= \left\langle g(\bar{w}_t), \bar{w}_t - \tilde{z}\right\rangle.
\end{aligned}$$

□

Next, we require the following bound on the size of the updates.

Lemma 59. *For any t, call the updates to the simplex variables due to the bilinear term*

$$\delta_t := \frac{1}{\kappa}(b - Ax_t),\ \delta^{(j)}_{t+\frac{1}{2}} := \frac{1}{\kappa}\left(b - A\left(x_t + \frac{1}{p_j}\Delta^{(j)}_t\right)\right).$$

Then, we have

$$\max\left(\|\delta_t\|_\infty, \left\|\delta^{(j)}_{t+\frac{1}{2}}\right\|_\infty\right) \le \frac{1}{4}.$$

Proof. First, the bound on δ_t follows by $\|b - Ax_t\|_\infty \le 2$ and κ is sufficiently large. Note that we may also conclude a stronger bound, that $\|\delta_t\|_\infty \le \frac{1}{8}$. By triangle inequality, it suffices to show

$$\left\|\frac{1}{\kappa p_j} A\Delta_t^{(j)}\right\|_\infty \le \frac{1}{8} \text{ for all } j \in [m].$$

Firstly, observe that $\Delta_t^{(j)}$ is 1-sparse, and can be bounded by noting (where med takes a median)

$$\left[x_{t+\frac{1}{2}}^{(j)}\right]_j = \text{med}\left(-1, [x_t]_j - \frac{1}{\kappa p_j}\left(\frac{\epsilon}{2}[x_t]_j + s \cdot A_{:j}^\top y_t\right), 1\right),$$

so that by definition of $\Delta_t^{(j)} = x_{t+\frac{1}{2}}^{(j)} - x_t$,

$$\left\|\Delta_t^{(j)}\right\|_\infty \le \frac{\epsilon}{2\kappa p_j}\left|[x_t]_j\right| + \frac{s}{\kappa p_j}\left|A_{:j}^\top y_t\right|.$$

Recall that

$$p_j(y_t) \ge \frac{\sqrt{\tilde{L}_j(y_t)}}{C\sqrt{ns} + \sqrt{mn\epsilon}} \ge \frac{8\sqrt{L_j(y_t)}}{\kappa}. \tag{4.27}$$

Here, the first inequality was from (H.12), and the second was from the definition of κ and (4.24). We now bound the size of entries of $\frac{1}{\kappa p_j} A\Delta_t^{(j)}$, recalling $\|x_t\|_\infty \le 1$:

$$\begin{aligned}
\frac{1}{\kappa p_j}\left\|A\Delta_t^{(j)}\right\|_\infty &\le \frac{\epsilon}{2\kappa^2 p_j^2}\|A_{:j}\|_\infty + \frac{s}{\kappa^2 p_j^2}\|A_{:j}\|_\infty \left|A_{:j}^\top y_t\right| \\
&= \frac{1}{\kappa^2}\frac{1}{p_j^2}\left(\frac{\epsilon}{2}\|A_{:j}\|_\infty + s\|A_{:j}\|_\infty \left|A_{:j}^\top y_t\right|\right) \\
&\le \frac{1}{64 L_j(y_t)}\left(\frac{\epsilon}{2}\|A_{:j}\|_\infty + s\|A_{:j}\|_\infty \left|A_{:j}^\top y_t\right|\right) \le \frac{1}{64}.
\end{aligned}$$

The last line follows from (4.27). This yields the claim, as $L_j(y_t) = s\|A_{:j}\|_\infty |A_{:j}|^\top y_t + \epsilon\|A_{:j}\|_\infty$. □

Leveraging this, the following lemma shows multiplicative stability of the simplex variables within a single iteration, which allows us to show that local smoothness estimates do not drift significantly. This proof is somewhat technical, and is deferred until the end of Section 4.5.3, as it requires opening up our implementation, which will yield the fact that the simplex points are not too unstable.

Lemma 60. *Coordinate-wise for any j, $y_{t+\frac{1}{2}}$, $y_{t+1}^{(j)}$ multiplicatively approximate y_t by a factor of at most 8. That is (where division is coordinate-wise),* $\max\left(y_{t+\frac{1}{2}}/y_t, y_{t+1}^{(j)}/y_t\right) \le 8$.

We also require the following (standard) local norms bound on the divergence of entropy.

Lemma 61 (Local norms)**.** *Let y, y' be on the simplex. Then for V the divergence with respect to entropy, $V_y(y') \ge \frac{1}{2}\|y - y'\|^2_{\mathbf{diag}((\,)\max(y,y'))^{-1}}$.*

Proof. Let $y_\alpha = (1-\alpha)y + \alpha y'$. By a Taylor expansion, letting h be entropy, we have

$$V_y(y') = \int_0^1 \int_0^\beta (y'-y)\nabla^2 h(y_\alpha)(y'-y)d\alpha d\beta \geq \frac{1}{2}\left\|y-y'\right\|^2_{\mathbf{diag}((\,)\max(y,y'))^{-1}}.$$

□

We now give a one-step convergence analysis of our algorithm, where use the definitions

$$g_j(z_t) := \left(\frac{1}{p_j}\left(A_{:j}^\top y_t + \frac{\epsilon}{2s}[x_t]_j\right)e_j,\ b - Ax_t + \frac{\epsilon}{4\log n}\log y_t\right),$$
$$g_j(w_t^{(j)}) := \left(\frac{1}{p_j}\left(A_{:j}^\top y_{t+\frac{1}{2}} + \frac{\epsilon}{2s}\left[x^{(j)}_{t+\frac{1}{2}}\right]_j\right)e_j,\ b - A\left(x_t + \frac{1}{p_j}\Delta_t^{(j)}\right) + \frac{\epsilon}{4\log n}\log y_{t+\frac{1}{2}}\right).$$

Lemma 62. *On any iteration t, we have (where expectations are over the randomness of the coordinate j in the iteration)*

$$\mathbb{E}\left[\frac{1}{\kappa}\left\langle g_j(w_t^{(j)}), w_t^{(j)} - \tilde{z}\right\rangle\right] \leq \mathbb{E}\left[V_{z_t}(\tilde{z}) - V_{z^{(j)}_{t+1}}(\tilde{z})\right].$$

Proof. Applying the first-order optimality conditions defining the two steps, as well as (F.1) following from the definition of Bregman divergences,

$$\frac{1}{\kappa}\left\langle g_j(z_t), w_t^{(j)} - z_{t+1}^{(j)}\right\rangle \leq V_{z_t}(z_{t+1}^{(j)}) - V_{w_t^{(j)}}(z_{t+1}^{(j)}) - V_{z_t}(w_t^{(j)})$$
$$\frac{1}{\kappa}\left\langle g_j(w_t^{(j)}), z_{t+1}^{(j)} - \tilde{z}\right\rangle \leq V_{z_t}(\tilde{z}) - V_{z^{(j)}_{t+1}}(\tilde{z}) - V_{z_t}(z_{t+1}^{(j)}).$$

Summing and rearranging slightly, we have

$$\frac{1}{\kappa}\left\langle g_j(w_t^{(j)}), w_t^{(j)} - \tilde{z}\right\rangle \leq V_{z_t}(\tilde{z}) - V_{z^{(j)}_{t+1}}(\tilde{z})$$
$$+ \frac{1}{\kappa}\left\langle g_j(w_t^{(j)}) - g_j(z_t), w_t^{(j)} - z_{t+1}^{(j)}\right\rangle - V_{w_t^{(j)}}(z_{t+1}^{(j)}) - V_{z_t}(w_t^{(j)}).$$

Taking an expectation, we have the conclusion up to proving the following claim, where we recall $\kappa = m\epsilon + 8\sqrt{mn\epsilon} + 8C\sqrt{ns} + 16n$:

$$\mathbb{E}\left[\left\langle g_j(w_t^{(j)}) - g_j(z_t), w_t^{(j)} - z_{t+1}^{(j)}\right\rangle\right] \leq \kappa\mathbb{E}\left[V_{w_t^{(j)}}(z_{t+1}^{(j)}) + V_{z_t}(w_t^{(j)})\right].$$

We will instead show the stronger claim that this is true for any particular $j \in [m]$:

$$\left\langle g_j(w_t^{(j)}) - g_j(z_t), w_t^{(j)} - z_{t+1}^{(j)}\right\rangle \leq \kappa\left(V_{w_t^{(j)}}(z_{t+1}^{(j)}) + V_{z_t}(w_t^{(j)})\right). \tag{4.28}$$

We will roughly do so by splitting the left hand side into three pieces, and then bounding them

separately. First, we rewrite it as (recalling $\Delta_t^{(j)} = x_{t+\frac{1}{2}}^{(j)} - x_t$ is 1-sparse)

$$\begin{aligned}
&\left\langle g_j(w_t^{(j)}) - g_j(z_t), w_t^{(j)} - z_{t+1}^{(j)} \right\rangle \\
&= \frac{1}{p_j}\left(\left\langle A_{:j}^\top(y_{t+\frac{1}{2}} - y_t)e_j, x_{t+\frac{1}{2}}^{(j)} - x_{t+1}^{(j)}\right\rangle + \left\langle A_{:j}^\top(y_{t+1}^{(j)} - y_{t+\frac{1}{2}})e_j, x_{t+\frac{1}{2}}^{(j)} - x_t\right\rangle\right) \\
&+\frac{\epsilon}{2sp_j}\left\langle \left(\left[x_{t+\frac{1}{2}}^{(j)}\right]_j - [x_t]_j\right)e_j, x_{t+\frac{1}{2}}^{(j)} - x_{t+1}^{(j)}\right\rangle + \frac{\epsilon}{4\log n}\left\langle \log\frac{y_{t+\frac{1}{2}}}{y_t}, y_{t+\frac{1}{2}} - y_{t+1}^{(j)}\right\rangle.
\end{aligned} \tag{4.29}$$

We bound the first term. By Lemma 61, and as Lemma 60 gives coordinatewise $y_{t+1}^{(j)}$, $y_{t+\frac{1}{2}} \le 8y_t$,

$$\begin{aligned}
V_{w_t^{(j)}}(z_{t+1}^{(j)}) + V_{z_t}(w_t^{(j)}) \ge \frac{1}{16}\left\|y_{t+1}^{(j)} - y_{t+\frac{1}{2}}\right\|_{\mathbf{diag}((\,)y_t^{-1})}^2 + \frac{1}{2s}\left\|x_{t+1}^{(j)} - x_{t+\frac{1}{2}}^{(j)}\right\|_2^2 \\
+ \frac{1}{16}\left\|y_{t+\frac{1}{2}} - y_t\right\|_{\mathbf{diag}((\,)y_t^{-1})}^2 + \frac{1}{2s}\left\|x_{t+\frac{1}{2}}^{(j)} - x_t\right\|_2^2.
\end{aligned}$$

We see that by $a^2 + b^2 \ge 2ab$ and Cauchy-Schwarz,

$$\begin{aligned}
&\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\left(\frac{1}{16}\left\|y_{t+1}^{(j)} - y_{t+\frac{1}{2}}\right\|_{\mathbf{diag}((\,)y_t^{-1})}^2 + \frac{1}{2s}\left\|x_{t+\frac{1}{2}}^{(j)} - x_t\right\|_2^2\right) \\
&\ge \sqrt{|A_{:j}^2|^\top y_t}\left(\frac{1}{\sqrt{8s}}\left|\left[x_{t+\frac{1}{2}}^{(j)} - x_t\right]_j\right|\left\|y_{t+1}^{(j)} - y_{t+\frac{1}{2}}\right\|_{\mathbf{diag}((\,)y_t^{-1})}\right) \\
&= \left(\frac{1}{\sqrt{8s}}\left|\left[x_{t+\frac{1}{2}}^{(j)} - x_t\right]_j\right|\right)\cdot\sqrt{\sum_i A_{ij}^2[y_t]_i}\sqrt{\sum_i \frac{[y_{t+1}^{(j)} - y_{t+\frac{1}{2}}]_i^2}{[y_t]_i}} \\
&\ge \left(\frac{1}{\sqrt{8s}}\left|\left[x_{t+\frac{1}{2}}^{(j)} - x_t\right]_j\right|\right)\cdot\sum_i |A_{ij}||[y_{t+1}^{(j)} - y_{t+\frac{1}{2}}]_i| \\
&\ge \frac{1}{\sqrt{8s}}\left\langle A_{:j}^\top(y_{t+1}^{(j)} - y_{t+\frac{1}{2}})e_j, x_{t+\frac{1}{2}}^{(j)} - x_t\right\rangle.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\left(\frac{1}{16}\left\|y_{t+\frac{1}{2}} - y_t\right\|_{\mathbf{diag}\left(y_t^{-1}\right)}^2 + \frac{1}{2s}\left\|x_{t+1}^{(j)} - x_{t+\frac{1}{2}}^{(j)}\right\|_2^2\right) \\
&\ge \frac{1}{\sqrt{8s}}\left\langle A_{:j}^\top(y_{t+\frac{1}{2}} - y_t)e_j, x_{t+\frac{1}{2}}^{(j)} - x_{t+1}^{(j)}\right\rangle.
\end{aligned}$$

Therefore, by the three above equations,

$$\begin{aligned}
&\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}\left(V_{w_t^{(j)}}(z_{t+1}^{(j)}) + V_{z_t}(w_t^{(j)})\right) \\
&\ge \frac{1}{p_j}\left(\left\langle A_{:j}^\top(y_{t+1}^{(j)} - y_{t+\frac{1}{2}})e_j, x_{t+\frac{1}{2}}^{(j)} - x_t\right\rangle + \left\langle A_{:j}^\top(y_{t+\frac{1}{2}} - y_t)e_j, x_{t+\frac{1}{2}}^{(j)} - x_{t+1}^{(j)}\right\rangle\right).
\end{aligned} \tag{4.30}$$

Now, we consider the second term. Directly applying strong-convexity and Cauchy-Schwarz gives

$$\frac{\epsilon}{2sp_j}\left\langle\left(\left[x^{(j)}_{t+\frac{1}{2}}\right]_j-[x_t]_j\right)e_j, x^{(j)}_{t+\frac{1}{2}}-x^{(j)}_{t+1}\right\rangle \le \frac{\epsilon}{2p_j}\left(\frac{1}{2s}\left\|x^{(j)}_{t+\frac{1}{2}}-x_t\right\|_2^2+\frac{1}{2s}\left\|x^{(j)}_{t+\frac{1}{2}}-x^{(j)}_{t+1}\right\|_2^2\right)$$
$$\le \frac{\epsilon}{2p_j}\left(V_{x_t}(x^{(j)}_{t+\frac{1}{2}})+V_{x^{(j)}_{t+\frac{1}{2}}}(x^{(j)}_{t+1})\right). \tag{4.31}$$

Finally, we consider the third term. It is straightforward to note that for any convex r (in this case, entropy), $\langle\nabla r(b)-\nabla r(a), b-c\rangle \le V_a(b)+V_b(c)$ for any three points a,b,c. Applying this,

$$\frac{\epsilon}{4\log n}\left\langle\log\frac{y_{t+\frac{1}{2}}}{y_t}, y_{t+\frac{1}{2}}-y^{(j)}_{t+1}\right\rangle \le \frac{\epsilon}{4\log n}\left(V_{y_t}(y_{t+\frac{1}{2}})+V_{y_{t+\frac{1}{2}}}(y^{(j)}_{t+1})\right). \tag{4.32}$$

Combining (4.30), (4.31), (4.32), we obtain

$$\langle g_j(w_t)-g_j(z_t), w_t-z_{t+1}\rangle \le \left(\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}+\frac{\epsilon}{2p_j}\right)\left(V_{x_t}(x^{(j)}_{t+\frac{1}{2}})+V_{x^{(j)}_{t+\frac{1}{2}}}(x^{(j)}_{t+1})\right)$$
$$+\left(\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}+\frac{\epsilon}{4\log n}\right)\left(V_{y_t}(y_{t+\frac{1}{2}})+V_{y_{t+\frac{1}{2}}}(y^{(j)}_{t+1})\right).$$

Finally, to prove (4.28), it remains to bound the size of the coefficients of the divergences by κ, and use nonnegativity of divergences. We claim the following holds:

$$\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}+\frac{\epsilon}{2p_j}\le \kappa = 16n+8\sqrt{mn\epsilon}+8C\sqrt{ns}+m\epsilon.$$

To see this, recall we assumed $p_j \ge \frac{1}{2m}$, and further that $\frac{1}{p_j}\le \frac{C\sqrt{ns}+\sqrt{mn\epsilon}}{\sqrt{\tilde{L}_j(y_t)}}$ by (H.12). Therefore,

$$\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}+\frac{\epsilon}{2p_j}\le \sqrt{8}(C\sqrt{ns}+\sqrt{mn\epsilon})+m\epsilon\le\kappa.$$

Similarly, it is easy to see that the following holds (corresponding to the coefficient of the divergences on the y side), concluding the proof:

$$\frac{\sqrt{\|A_{:j}\|_\infty |A_{:j}|^\top y_t}\sqrt{8s}}{p_j}+\frac{\epsilon}{4\log n}\le\kappa.$$

□

We require the following helper lemma which upper bounds divergence via regret, which allows

us to finally convert our regret bound into a divergence bound for the output iterate.

Lemma 63. *For any point z, we have $\langle g(z), z - \tilde{z}\rangle \geq \frac{\epsilon}{4\log n} V_z(\tilde{z})$, where we recall for $z = (x, y)$,*

$$g(z) = \left(A^\top y + \frac{\epsilon}{2s}x,\ b - Ax + \frac{\epsilon}{4\log n}\log y\right).$$

Proof. Recall that because $\tilde{z}$ is the saddle point of the convex-concave function g is the gradient operator of, by first-order optimality,

$$\langle g(\tilde{z}), z - \tilde{z}\rangle \geq 0\ \forall z.$$

Therefore, noting terms $\left\langle A^\top(y - \tilde{y}), x - \tilde{x}\right\rangle - \langle A(x - \tilde{x}), y - \tilde{y}\rangle$ cancel,

$$\begin{aligned}\langle g(z), z - \tilde{z}\rangle \geq \langle g(z) - g(\tilde{z}), z - \tilde{z}\rangle &= \frac{\epsilon}{2s}\left\|x - \tilde{x}\right\|_2^2 + \frac{\epsilon}{4\log n}\left\langle \log\frac{y}{\tilde{y}}, y - \tilde{y}\right\rangle \\ &\geq \epsilon V_x(\tilde{x}) + \frac{\epsilon}{4\log n}V_y(\tilde{y}) \geq \frac{\epsilon}{4\log n}V_z(\tilde{z}).\end{aligned}$$

The last line used nonnegativity of the Bregman divergence and $\langle \nabla r(y) - \nabla r(\tilde{y}), y - \tilde{y}\rangle \geq V_y(\tilde{y})$. □

We now give the method in phase k, initialized at $(x_{k,0}, y_{k,0})$. Here, we briefly comment on inexactness issues. The algorithm requires maintenance of variable y on the simplex, and various quantities which are functions of y, which we can only approximately compute (cheaply): the method outlined in Section 4.3.5 no longer applies, because updates to the variable are dense. Formally, we define `Y-Oracle`, a data structure which maintains an internal representation of the simplex variables. `Y-Oracle` supports the following operations in each iteration (k, t), in amoritized $\widetilde{O}(1)$ time:

- `Y-Oracle.Sample()`: Samples $j \in [m]$ from $p_j(y_{k,t})$. Returns $(j, p_j(y_{k,t}))$.
- `Y-Oracle.Coord`(i): Returns $[\breve{y}_{k,t}]_i$ such that $|[\breve{y}_{k,t}]_i - [y_{k,t}]_i| < n^{-100}$.
- `Y-Oracle.Update-Half`(v): Updates the internal representation of $y_{k,t+\frac{1}{2}}$.
- `Y-Oracle.Coord-Half`(i): Returns $[\breve{y}_{k,t+\frac{1}{2}}]_i$ such that $|[\breve{y}_{k,t+\frac{1}{2}}]_i - [y_{k,t+\frac{1}{2}}]_i| < n^{-100}$.
- `Y-Oracle.Update`(v): Updates the internal representation of $y_{k,t+1}$.

We develop `Y-Oracle` in Section 4.5.3. The following is the algorithm for phase k.

1. Let $\kappa = m\epsilon + 8\sqrt{mn\epsilon} + 8C\sqrt{ns} + 16n$ where C is the constant of Lemma 57.
2. Let $T = \left\lceil \frac{8\kappa\log n}{\epsilon}\right\rceil$ be the number of iterations per phase.
3. Sample a stopping iteration uniformly at random $t_k^* \in [T]$.

4. For iteration $t \in [t_k^* - 1]$:
   (a) Call `Y-Oracle.Sample` to obtain j, $p_j(y_{k,t})$ (for shorthand, denoted p_j).
   (b) For each non-zero entry A_{ij} of $A_{:j}$, call `Y-Oracle.Coord`(i) to obtain $[\breve{y}_{k,t}]_i$.
   (c) $x_{k,t+\frac{1}{2}} \leftarrow \operatorname{argmin}_{x\in[-1,1]^m} \left\{ \left\langle \frac{1}{\kappa p_j}(A_{:j}^\top \breve{y}_{k,t} + \frac{\epsilon}{2s}[x_{k,t}]_j)e_j, x \right\rangle + V_{x_{k,t}}(x) \right\}$.
   (d) $y_{k,t+\frac{1}{2}} \leftarrow \operatorname{argmin}_{y\in\Delta^n} \left\{ \left\langle \frac{1}{\kappa} \left(b - Ax_{k,t} + \frac{\epsilon}{4\log n} \log y_{k,t} \right), y \right\rangle + V_{y_{k,t}}(y) \right\}$.
   (e) $\Delta_{k,t} := x_{k,t+\frac{1}{2}} - x_{k,t}$.
   (f) For each non-zero entry A_{ij} of $A_{:j}$, call `Y-Oracle.Coord`(i) to obtain $[\breve{y}_{k,t+\frac{1}{2}}]_i$.
   (g) $x_{k,t+1} \leftarrow \operatorname{argmin}_{x\in[-1,1]^m} \left\{ \left\langle \frac{1}{\kappa p_j}(A_{:j}^\top \breve{y}_{k,t+\frac{1}{2}} + \frac{\epsilon}{2s}[x_{k,t+\frac{1}{2}}]_j)e_j, x \right\rangle + V_{x_{k,t}}(x) \right\}$.
   (h) $y_{k,t+1} \leftarrow \operatorname{argmin}_{y\in\Delta^n} \left\{ \left\langle \frac{1}{\kappa} \left(b - A\left(x_{k,t} + \frac{1}{p_j}\Delta_{k,t} \right) + \frac{\epsilon}{4\log n} \log y_{k,t+\frac{1}{2}} \right), y \right\rangle + V_{y_{k,t}}(y) \right\}$.
5. For iteration $t = t_k^*$:
   (a) $\forall j \in [m]$, $\Delta_{k,t}^{(j)} := \operatorname{argmin}_{x\in[-1,1]^m} \left\{ \left\langle \frac{1}{\kappa p_j}(A_{:j}^\top y_{k,t} + \frac{\epsilon}{2s}[x_{k,t}]_j)e_j, x \right\rangle + V_{x_{k,t}}(x) \right\} - x_{k,t}$.
   (b) Compute $\Delta_{k,t}^{(j)}$ for all $j \in [m]$.
   (c) Define $x_{k+1,0} = x_{k,t} + \sum_j \Delta_{k,t}^{(j)}$.
   (d) Define $y_{k+1,0} = \operatorname{argmin}_{y\in\Delta^n} \left\{ \left\langle \frac{1}{\kappa} \left(b - Ax_{k,t} + \frac{\epsilon}{4\log n} \log y_{k,t} \right), y \right\rangle + V_{y_{k,t}}(y) \right\}$.
6. Output $(x_{k+1,0}, y_{k+1,0})$.

We remark that in each loop of step 4, steps (c), (e) and (g) are implemented directly in $\tilde{O}(1)$ time, step (d) is implemented implicitly using `Y-Oracle.Update-Half`, and step (h) is implemented implicitly using `Y-Oracle.Update`. We will discuss the efficient implementation of the procedures supported by `Y-Oracle` in Section 4.5.3.

We now come to the main export of this section, which shows that the expected divergence to the saddle point halves in every phase. In this lemma, we assume exact implementation of the steps; we discuss how to deal with inexactness issues in the analysis in Section 4.5.3.

Lemma 64. *Suppose phase k is initialized with $z_{k,0} = (x_{k,0}, y_{k,0})$. Then, the output $z_{k+1,0} = (x_{k+1,0}, y_{k+1,0})$ satisfies (where expectations are taken over the randomness used in phase k)*

$$\mathbb{E}\left[V_{z_{k+1,0}}(\tilde{x}, \tilde{y})\right] \le \frac{1}{2} V_{z_{k,0}}(\tilde{x}, \tilde{y}).$$

Proof. Consider running for all of the $T = \left\lceil \frac{8\kappa \log n}{\epsilon} \right\rceil$ iterations. Taking an expectation of Lemma 62 over the entire phase, telescoping, and using nonnegativity of divergences, we obtain (for $\tilde{z} = (\tilde{x}, \tilde{y})$)

$$\mathbb{E}\left[\frac{1}{T} \sum_{t\in[T]} \langle g_{j_t}(w_{k,t}), w_{k,t} - \tilde{z} \rangle \right] \le \frac{\kappa V_{z_{k,0}}(\tilde{z})}{T} \le \frac{\epsilon}{8\log n} V_{z_{k,0}}(\tilde{z}).$$

Here, j_t is the coordinate sampled in the t^{th} iteration. Applying Lemma 58, we instead have

$$\mathbb{E}\left[\frac{1}{T}\sum_{t\in[T]}\langle g(\bar{w}_{k,t}), \bar{w}_{k,t} - \tilde{z}\rangle\right] \leq \frac{\epsilon}{8\log n} V_{z_{k,0}}(\tilde{z}).$$

Now, because we randomly sampled a $t \in [T]$ to be the index t_k^* and passed $\bar{w}_{t_k^*}$ to the $k+1^{st}$ phase as $z_{k+1,0} = (x_{k+1,0}, y_{k+1,0})$, we obtain

$$\mathbb{E}\left[\langle g(z_{k+1,0}), z_{k+1,0} - \tilde{z}\rangle\right] \leq \frac{\epsilon}{8\log n} V_{z_{k,0}}(\tilde{z}).$$

The conclusion follows from applying Lemma 63. □

Algorithm statement

We now state the full algorithm, which is composed of phases, each of which halves the expected divergence to the saddle point.

1. Initialize $x_{0,0} = 0, y_{0,0} = \frac{1}{n}\mathbf{1}$.
2. Let $\Theta_0 = 1 + \log n$ be the initial divergence bound (Lemma 56). Let $K = \left\lceil \log_2\left(\frac{16s\Theta_0}{\epsilon^2}\right)\right\rceil$.
3. For phase $k \in [K]$:
   (a) Run the procedure in Section 4.5.2, initialized at $z_{k,0}$, to produce the point $z_{k+1,0}$.
4. Return $\hat{x} := x_{K,0}$.

We now analyze the correctness and runtime of this algorithm. We assume the following lemma, which will be proven in Section 4.5.3.

Lemma 65. *Every n iterations of each phase can be implemented in $\tilde{O}(n)$ time. Furthermore, for each phase k, iteration t_k^* can be implemented in $\tilde{O}(m)$ time.*

Theorem 25. *The algorithm has runtime*

$$\tilde{O}\left(m + \frac{n + \sqrt{ns}}{\epsilon}\right),$$

and satisfies $\mathbb{E}[\|\hat{x} - \tilde{x}\|_2] \leq \frac{\epsilon}{2}$, where the expectation is over all randomness in the algorithm.

Proof. To prove the first statement, note that the algorithm computes at most K points of the form $\bar{w}_{k,t_k^*}$, and takes at most KT steps. Thus, by Lemma 65 this yields a runtime of

$$\tilde{O}(KT) + \tilde{O}(mK) = \tilde{O}\left(\frac{\kappa}{\epsilon} + m\right) = \tilde{O}\left(\frac{m\epsilon + \sqrt{mn\epsilon} + \sqrt{ns} + n}{\epsilon}\right) = \tilde{O}\left(m + \frac{n + \sqrt{ns}}{\epsilon}\right).$$

We used that $\sqrt{mn/\epsilon}$ is never larger than $m + n/\epsilon$, as it is their geometric mean. To prove the second statement, we apply Lemma 64 for $K \geq \log\left(\frac{16s\Theta_0}{\epsilon^2}\right)$:

$$\mathbb{E}\left[\|\hat{x} - \tilde{x}\|_2\right]^2 \leq \mathbb{E}\left[\|\hat{x} - \tilde{x}\|_2^2\right] = 4s\mathbb{E}[V_{\hat{x}}(\tilde{x})] \leq \frac{4s\Theta_0}{2^K} \leq \left(\frac{\epsilon}{2}\right)^2.$$

The first inequality used convexity of the square, and the second inequality repeatedly used Lemma 64 and iterated expectations. This implies the desired bound. □

We then see that $\hat{x}$ is our desired approximate minimizer, in expectation.

Corollary 14. *We have $\|\hat{x}\|_\infty \leq 1$, and $\mathbb{E}[\|A\hat{x} - b\|_\infty] \leq \mathrm{OPT} + \epsilon$.*

Proof. The first statement is immediate from the algorithm, since in each iteration $x_{k,t}$ lies in $[-1, 1]^m$, and for each j, $x_{k,t} + \Delta_{k,t}^{(j)}$ is also defined to lie in $[-1, 1]^m$, and the region decomposes coordinatewise. The second statement follows from $\|A\|_\infty \leq 1$, and

$$\mathbb{E}[\|A\hat{x} - b\|_\infty] \leq \|A\tilde{x} - b\|_\infty + \mathbb{E}[\|A(\hat{x} - \tilde{x})\|_\infty] \leq \mathrm{OPT} + \frac{\epsilon}{2} + \mathbb{E}[\|\hat{x} - \tilde{x}\|_2] \leq \mathrm{OPT} + \epsilon.$$

□

By Markov's inequality, this means that with half probability we have a 2ϵ-approximate minimizer. This can be boosted to probability $1 - \delta$ using $\log\frac{1}{\delta}$ independent runs, and it does not affect runtime asymptotically since computing objective value takes time $\tilde{O}(m)$.

4.5.3 Runtime

This section proves Lemma 65, which states that we can implement each iteration of each phase in amoritized $\tilde{O}(1)$ time for each $t \neq t_k^*$, and that we can implement the last iteration in $\tilde{O}(m)$ time. As discussed in Section 4.5.2, it suffices to show that `Y-Oracle`.{`Sample`, `Coord`, `Update`, `Update-Half`} may be implemented in amoritized time $\tilde{O}(n)$ every n iterations. In particular, assuming these operations are supported, it is simple to see that we can implement the updates to the x variables in $\tilde{O}(1)$ time per iteration by sparsity. Finally, the last iteration can be implemented in $\tilde{O}(m)$ time simply by performing the updates to the x variable m times.

Reducing sampling from and computing p_j to sampling from and computing $\sqrt{y}$

We first reduce the implementation of `Y-Oracle.Sample` in an iteration (k, t) to being able to efficiently sample proportional to $\sqrt{[y_{k,t}]_i}$ (we drop (k, t) for simplicity). Recall we sample from

$$p_j = \frac{C\sqrt{ns}}{C\sqrt{ns} + \sqrt{mn\epsilon}} \cdot \frac{\sqrt{s\|A_{:j}\|_\infty} \sum_i \sqrt{|A_{ij}|y_i}}{\sum_j \sqrt{s\|A_{:j}\|_\infty} \sum_i \sqrt{|A_{ij}|y_i}} + \frac{\sqrt{mn\epsilon}}{C\sqrt{ns} + \sqrt{mn\epsilon}} \cdot \frac{\sqrt{\epsilon\|A_{:j}\|_\infty}}{\sum_j \sqrt{\epsilon\|A_{:j}\|_\infty}}.$$

First flip a coin which is heads with probability $C\sqrt{ns}/(C\sqrt{ns}+\sqrt{mn\epsilon})$. If it comes up tails, we sample a j proportional to $\sqrt{\epsilon \left\|A_{:j}\right\|_\infty}$; clearly we may precompute all of these probabilities, and place them at the leaves of a binary tree (along with each of the subtree sums stored at roots of subtrees), flipping $\tilde{O}(1)$ appropriately biased coins to sample from this distribution. Next, in order to sample from a distribution over j proportional to $\sqrt{s \left\|A_{:j}\right\|_\infty} \sum_i \sqrt{|A_{ij}| y_i}$, it clearly suffices to instead sample an i proportional to $\sqrt{y_i}$, and then sample a j proportional to $\sqrt{s \left\|A_{:j}\right\|_\infty |A_{ij}|}$; this latter distribution we can precompute.

We now discuss computing a particular p_j in $\tilde{O}(1)$ time: we need to in fact compute the true p_j which we sampled from, because otherwise we will not have an unbiased estimator. To do so, it clearly suffices to compute the conditional probabilities

$$\frac{\sqrt{s \left\|A_{:j}\right\|_\infty} \sum_i \sqrt{|A_{ij}| y_i}}{\sum_j \sqrt{s \left\|A_{:j}\right\|_\infty} \sum_i \sqrt{|A_{ij}| y_i}}, \ \frac{\sqrt{\epsilon \left\|A_{:j}\right\|_\infty}}{\sum_j \sqrt{\epsilon \left\|A_{:j}\right\|_\infty}}.$$

The latter of these is simple to pre-compute. To compute the former, let

$$q_{ij} := \frac{\sqrt{s \left\|A_{:j}\right\|_\infty |A_{ij}|}}{\sum_i \sqrt{s \left\|A_{:j}\right\|_\infty |A_{ij}|}}.$$

We observe that for any j, at most $\tilde{O}(1)$ of the q_{ij} are non-zero, and we can also precompute all the q_{ij}. Finally, the conclusion follows from

$$\frac{\sqrt{s \left\|A_{:j}\right\|_\infty} \sum_i \sqrt{|A_{ij}| y_i}}{\sum_j \sqrt{s \left\|A_{:j}\right\|_\infty} \sum_i \sqrt{|A_{ij}| y_i}} = \sum_i \frac{\sqrt{y_i}}{\sum_i \sqrt{y_i}} \cdot q_{ij}.$$

Now, we only need to evaluate $\tilde{O}(1)$ nonzero summands. In conclusion, in order to sample from and compute p_j in time $\tilde{O}(1)$ per iteration, it suffices to sample from and compute a probability distribution proportional to $\sqrt{y}$ (we remark that our sampling procedure will be exact).

Sparse combinations

In this section we describe how to maintain v_t, $v_{t+\frac{1}{2}}$ which are $\log y_t, \log y_{t+\frac{1}{2}}$ up to an additive multiple of the ones vector, via a linear combination of sparsely updated vectors q_t, r_t, s_t. The reason for this representation is so that we may update the representation in time $\tilde{O}(1)$ per iteration, and further, for any coordinate i, we may compute $\exp([v_t]_i)$ in constant time by simply taking the appropriate linear combination of the vectors. The word "sparse" in this section denotes any vector with $\tilde{O}(1)$ nonzero entries. We begin by recalling the notation from Lemma 59,

$$\delta_t := \frac{1}{\kappa}(b - Ax_t), \ \delta^{(j)}_{t+\frac{1}{2}} := \frac{1}{\kappa}\left(b - A\left(x_t + \frac{1}{p_j}\Delta^{(j)}_t\right)\right).$$

Now we write the updates to $y_{t+\frac{1}{2}}$, $y^{(j)}_{t+1}$ in the following form, for $c = \frac{\epsilon}{4\kappa \log n}$:

$$y_{t+\frac{1}{2}} \propto \exp\left(\log y_t - c \log y_t - \delta_t\right),\ y^{(j)}_{t+1} \propto \exp\left(\log y_t - c \log y_{t+\frac{1}{2}} - \delta^{(j)}_{t+\frac{1}{2}}\right).$$

Letting vectors v_t, $v_{t+\frac{1}{2}}$ satisfy $y_t \propto \exp(v_t)$, $y_{t+\frac{1}{2}} \propto \exp(v_{t+\frac{1}{2}})$ for all t, we have the recursion

$$v_{t+\frac{1}{2}} = (1-c)v_t - \delta_t,\ v_{t+1} = v_t - c v_{t+\frac{1}{2}} - \delta^{(j)}_{t+\frac{1}{2}}.$$

Recalling the structure of these updates, we see that we can further decompose $\delta^{(j)}_{t+\frac{1}{2}}$ into $\delta_t + \zeta_t$, where $\zeta_t = -\frac{1}{\kappa p_j} A \Delta^{(j)}_t$ is sparse. Next, observing $\|b - Ax_t\|_\infty \leq 2$ and $\kappa \geq 16n$, we can assume $\|\delta_t\|_\infty \leq \frac{1}{8n}$. We additionally note that $\delta_t - \delta_{t-1} = -\frac{1}{\kappa} A(x_t - x_{t-1})$ is sparse, since $x_t - x_{t-1}$ is 1-sparse and A has sparse columns. Altogether, this yields

$$\begin{aligned}
v_{t+1} &= (1 - c + c^2)v_t - (1-c)\delta_t - \zeta_t \\
\Rightarrow v_{t+1} - v_t &= (1 - c + c^2)(v_t - v_{t-1}) - (1-c)(\delta_t - \delta_{t-1}) - (\zeta_t - \zeta_{t-1}) \\
\Rightarrow v_{t+1} &= (2 - c + c^2)v_t - (1 - c + c^2)v_{t-1} - (1-c)(\delta_t - \delta_{t-1}) - (\zeta_t - \zeta_{t-1}) \\
&\Rightarrow v_{t+1} = c_1 v_t - c_2 v_{t-1} - c_3 \mu_t - \nu_t.
\end{aligned}$$

Here, we have defined $c_1 = 2 - c + c^2, c_2 = 1 - c + c^2, c_3 = 1 - c, \mu_t = \delta_t - \delta_{t-1}, \nu_t = \zeta_t - \zeta_{t-1}$. Further, $c_1 \leq 2$ and $c_2 \leq 1$. Similarly, we can compute

$$\begin{aligned}
v_{t+\frac{1}{2}} = (1-c)v_t - \delta_t \Rightarrow v_{t+\frac{1}{2}} - v_{t-\frac{1}{2}} &= (1-c)(v_t - v_{t-1}) - (\delta_t - \delta_{t-1}) \\
&\Rightarrow v_{t+\frac{1}{2}} = c_3 v_t - c_3 v_{t-1} + v_{t-\frac{1}{2}} - \mu_t.
\end{aligned}$$

In matrix-vector multiplication notation, this update is (where M is clearly full rank)

$$\begin{pmatrix} v_{t+1} & v_{t+\frac{1}{2}} & v_t \end{pmatrix} = \begin{pmatrix} v_t & v_{t-\frac{1}{2}} & v_{t-1} \end{pmatrix} M - \begin{pmatrix} c_3\mu_t + \nu_t & \mu_t & 0 \end{pmatrix},$$

$$M = \begin{pmatrix} c_1 & c_3 & 1 \\ 0 & 1 & 0 \\ -c_2 & -c_3 & 0 \end{pmatrix}.$$

Now, suppose we have maintained a representation

$$\begin{pmatrix} v_t & v_{t-\frac{1}{2}} & v_{t-1} \end{pmatrix} = \begin{pmatrix} q_t & r_t & s_t \end{pmatrix} M^t.$$

We then require the update

$$\begin{pmatrix} q_{t+1} & r_{t+1} & s_{t+1} \end{pmatrix} M^{t+1} = \begin{pmatrix} q_t & r_t & s_t \end{pmatrix} M^{t+1} - (\begin{pmatrix} c_3\mu_t + \nu_t & \mu_t & 0 \end{pmatrix} M^{-t-1})M^{t+1}.$$

We can maintain M^{-t-1} in closed form by simply performing a single matrix multiplication of 3×3 matrices each iteration, so the updates to q_{t+1}, r_{t+1} and s_{t+1} are sparse:

$$\begin{pmatrix} q_{t+1} & r_{t+1} & s_{t+1} \end{pmatrix} = \begin{pmatrix} q_t & r_t & s_t \end{pmatrix} - \begin{pmatrix} c_3\mu_t + \nu_t & \mu_t & 0 \end{pmatrix} M^{-t-1}.$$

Maintaining the sum of exponentials

The previous section states that we can maintain a representation of v_t in $\tilde{O}(1)$ time per iteration, such that we can query for any i, the value $\exp([v_t]_i)$ in constant time (respectively, $\exp([v_{t+\frac{1}{2}}]_i)$). Consequently, in order to support `Y-Oracle.Coord` (respectively, `Y-Oracle.Coord-Half`), we need to be able to approximate

$$\sum_{i \in [n]} \exp([v_t]_i) \tag{4.33}$$

multiplicatively by $1 + \frac{1}{n^{100}}$. In this section we will discuss how to do so over n iterations in time $\tilde{O}(n)$. We then discuss how to sample from this distribution, and modify this maintenance to also support approximate coordinate queries from $y_{t+\frac{1}{2}}$. We will not formally discuss how to extend this analysis to query and sample from a distribution proportional to $\sqrt{y_t}$, as required by 4.5.3, as it is an immediate generalization; we simply also implement `Y-Oracle` with the vectors $\frac{1}{2}v_t$, which clearly suffices. For the scope of this section, define the constant

$$c = \frac{\epsilon}{4\kappa \log n}, \ \kappa > 16n.$$

The implementation problem is: for every iteration $t \in [n]$, we are given vectors δ_t, ζ_t, such that

- $\|\zeta_t\|_\infty \leq \frac{1}{8}$, and ζ_t is sparse.
- $\|\delta_t\|_\infty \leq \frac{1}{8n}$.
- Vectors v_t are defined recursively via $v_{t+1} := (1-c)v_t + \delta_t + \zeta_t$.
- We are able to maintain a representation of v_t as a linear combination $\alpha_t q_t + \beta_t r_t + \gamma_t s_t$, for sparsely changing q_t, r_t, s_t, and scalars $\alpha_t, \beta_t, \gamma_t$.

These bounds follow from the analysis in Lemma 59. We also require the following fact on the effect of a certain "squishing" operation, which states that we may take any coordinate of v_t which is significantly smaller than another, and raise it within a certain range.

Lemma 66. *Let $v \in \mathbb{R}^n$, and let $y \in \Delta^n$ be such that $y \propto \exp(v)$. Consider the following operation: let i^*, i' be coordinates of v such that $v_{i'} < v_{i^*} - \frac{16 \log n}{\epsilon}$, and set $\hat{v} = v$ in every coordinate, except $\hat{v}_{i'} \leftarrow v_{i^*} - \frac{16 \log n}{\epsilon}$. Then, for $\hat{y} \propto \exp(\hat{v})$, assuming $\epsilon < \frac{1}{7}$,*

$$V_y(\tilde{y}) - V_{\hat{y}}(\tilde{y}) > -n^{-100}.$$

Proof. We explicitly compute

$$V_y(\tilde{y}) - V_{\hat{y}}(\tilde{y}) = \sum_i \tilde{y}_i \log \frac{\hat{y}_i}{y_i}.$$

Note that the only possible i such that $\frac{\hat{y}_i}{y_i} \geq 1$ is $i = i'$. Furthermore, for every other coordinate i,

$$\frac{y_i}{\hat{y}_i} = \frac{\frac{\exp(v_i)}{\|\exp(v)\|_1}}{\frac{\exp(\tilde{v}_i)}{\|\exp(\tilde{v})\|_1}} = \frac{\|\exp(\tilde{v})\|_1}{\|\exp(v)\|_1} < \frac{1 + n^{-\frac{16}{\epsilon}}}{1}.$$

Here, we used that $\exp(\hat{v}_{i'})$ can be at most $\exp(-\frac{16 \log n}{\epsilon})$ of the sum, due to the contribution of the v_{i^*} term, and all coordinates $i \neq i'$ have $\hat{v}_i = v_i$. Finally,

$$\sum_i \tilde{y}_i \log \frac{\hat{y}_i}{y_i} \geq -\sum_{i \neq i'} \tilde{y}_i \log(1 + n^{-100}) \geq -n^{-100}.$$

□

We assume that in the first iteration, we have spent $O(n)$ time computing v_0 explicitly, using our sparse representation, and squishing so its coordinates lie in the range $[0, \frac{16 \log n}{\epsilon}]$.

The case $\zeta_t = 0$.

We first handle the case when all of the $\zeta_t = 0$. At iteration 0, suppose we have spent $O(n)$ time to compute $i^* = \text{argmax}_i [v_0]_i$. Also, recall we guaranteed $[v_0]_{i^*} - [v_0]_i \leq \frac{16 \log n}{\epsilon}$. We use the following fact:

Fact 7 (Taylor expansion of exponential)**.** *Let $|x| \leq \frac{1}{2}$. Then, letting $\text{Tay}_d(x)$ be the degree d Taylor approximation of the exponential, we can bound $|\text{Tay}_d(x) - \exp(x)| \leq \frac{1}{2^d}$.*

To approximate (4.33) on iteration t, we will maintain a scalar σ_t with the guarantee

$$\|v_0 - \sigma_t \mathbf{1} - v_t\|_\infty \leq \frac{1}{2}. \tag{4.34}$$

We will explicitly compute $[v_t]_{i^*}$ each iteration t, and set

$$\sigma_{t+1} = \sigma_t + c[v_t]_{i^*}. \tag{4.35}$$

First of all, we show the invariant (4.34).

Lemma 67. *Every iteration $t \le n$, and for all i, $|[v_t]_i - [v_t]_{i^*}| \le \frac{17 \log n}{\epsilon}$.*

Proof. We claim that the range of the coordinates of v_t is never larger than $\frac{17 \log n}{\epsilon}$: certainly, this implies the conclusion. To show this, we inductively claim that the range satisfies

$$\max_i [v_t]_i - \min_j [v_t]_j \le \frac{16 \log n}{\epsilon} + \frac{t}{4n}.$$

Taking $t \le n$ yields the result. Clearly for $t = 0$ this is true; now, for $t+1$, recall $v_{t+1} = (1-c)v_t + \delta_t$. Let $i = \text{argmax}_i [v_t]_i$, $j = \text{argmin}_j [v_t]_j$. Then,

$$[v_{t+1}]_i - [v_{t+1}]_j = (1-c)([v_t]_i - [v_t]_j) + ([\delta_t]_i - [\delta_t]_j) \le \frac{16 \log n}{\epsilon} + \frac{t}{4n} + \frac{1}{4n}.$$

Here we used the inductive guarantee and the range of δ_t (we may clearly assume $\log n/\epsilon > 1$). □

Lemma 68. *Every iteration $t \le n$, (4.34) holds.*

Proof. For some particular i, we show it holds; this implies the ℓ_∞ guarantee. Note that

$$|[v_0]_i - \sigma_{t+1} - [v_{t+1}]_i| \le |[v_0]_i - \sigma_t - [v_t]_i| + |([v_t]_i - [v_{t+1}]_i) - c[v_t]_{i^*}|.$$

Here we used triangle inequality and the definitions of $\sigma, \bar{v}$. Now, we have

$$|([v_t]_i - [v_{t+1}]_i) - c[v_t]_{i^*}| \le |c[v_t]_i - c[v_t]_{i^*}| + |[\delta_t]_i| \le \frac{3}{8n} + \frac{1}{8n} \le \frac{1}{2n}.$$

Thus, inductively we have that

$$|[v_0]_i - \sigma_t - [v_t]_i| \le \frac{t}{2n}.$$

Using $t \le n$ yields the result. □

Finally, we describe how to compute an accurate approximation (4.33) by Taylor expansion in $\tilde{O}(1)$ time per iteration. We approximate, for some $d = O(\log n)$,

$$\begin{aligned} \sum_{i \in [n]} \exp([v_t]_i) &= \sum_{i \in [n]} \exp\left([v_0]_i - \sigma_t\right) \exp\left([v_t]_i - ([v_0]_i - \sigma_t)\right) \\ &\approx \exp(-\sigma_t) \sum_{i \in [n]} \exp\left([v_0]_i\right) \text{Tay}_d\left(\alpha_t [q_t]_i + \beta_t [r_t]_i + \gamma_t [s_t]_i - ([v_0]_i - \sigma_t)\right). \end{aligned} \tag{4.36}$$

We now group by the degree of the Taylor expansion, $0 \le k \le d$, and each quintuple $0 \le d_1 + d_2 +$

$d_3 + d_4 + d_5 = k \le d$:

$$\exp(-\sigma_t) \sum_{i\in[n]} \exp([\bar{v}_t]_i) \sum_{0\le k\le d} \frac{(\alpha_t[q_t]_i + \beta_t[r_t]_i + \gamma_t[s_t]_i - ([v_0]_i - \sigma_t))^k}{k!}$$

$$= \exp(-\sigma_t) \sum_{i\in[n]} \exp([v_0]_i) \sum_{d_1,d_2,d_3,d_4,d_5} \frac{\binom{k}{d_1,d_2,d_3,d_4,d_5}}{k!} (\alpha_t)^{d_1}(\beta_t)^{d_2}(\gamma_t)^{d_3}(-1)^{d_4}[q_t]_i^{d_1}[r_t]_i^{d_2}[s_t]_i^{d_3}[v_0]_i^{d_4}[\sigma_t]^{d_5}$$

$$= \exp(-\sigma_t) \sum_{d_1,d_2,d_3,d_4,d_5} \frac{\binom{k}{d_1,d_2,d_3,d_4,d_5}}{k!} (\alpha_t)^{d_1}(\beta_t)^{d_2}(\gamma_t)^{d_3}(-1)^{d_4} \sum_{i\in[n]} \exp([v_0]_i)[q_t]_i^{d_1}[r_t]_i^{d_2}[s_t]_i^{d_3}[v_0]_i^{d_4}[\sigma_t]^{d_5}.$$

Consider the complexity of computing the last expression. There are at most $(d+1)^5 = O(d^5)$ quintuplets d_1, d_2, d_3, d_4, d_5 with $0 \le d_1 + d_2 + d_3 + d_4 + d_5 \le d$. For each quintuplet, we maintain

$$\sum_{i\in[n]} \exp([v_0]_i)[q_t]_i^{d_1}[r_t]_i^{d_2}[s_t]_i^{d_3}[v_0]_i^{d_4}[\sigma_t]^{d_5}.$$

Because each of q_t, r_t, s_t, are sparsely changing, we can spend $\tilde{O}(d^5)$ time updating the relevant terms in each of these summations. Furthermore, $[\sigma_t]^{d_5}$ is simply a scalar so we can rescale its contribution to the entire sum in constant time. Now, in order to compute the overall sum, we can spend constant time computing each coefficient

$$\frac{\binom{k}{d_1,d_2,d_3,d_4,d_5}}{k!} (\alpha_t)^{d_1}(\beta_t)^{d_2}(\gamma_t)^{d_3}(-1)^{d_4};$$

this takes $\tilde{O}(d^5)$ time altogether. Lastly, updating $\exp(-\sigma_t)$, the scaling of the entire sum, takes constant time, and computing the overall sum thus takes $\tilde{O}(d^5)$.

Finally, we must argue that performing this procedure for $d = O(\log n)$ suffices for a multiplicative guarantee of $1 + \frac{1}{n^{O(1)}}$. Comparing the approximation in (4.36) to the required (4.33), the only difference is each of the approximations

$$\exp([v_t]_i - ([v_0]_i - \sigma_t)) \approx \mathrm{Tay}_d([v_t]_i - ([v_0]_i - \sigma_t)).$$

Because the left hand side is bounded between $\exp(\pm\frac{1}{2})$, an additive approximation is (up to constants) a multiplicative approximation as well. Further, Fact 7 implies that $d = O(\log n)$ suffices for this quality of approximation, as desired.

Binomial heap data structures for ζ_t.

In this section, we reduce the general case to the case where $\zeta_t = 0$ via a binomial heap data structure, a fairly general reduction. We note that the analysis in the previous section also clearly holds when the number of iterations is less than n, and when there are less than n coordinates. The main idea of the reduction is that we will maintain data structures for sets $\{S_k\}$ for $0 \le k \le \lceil \log n \rceil$,

such that a S_k either contains no elements, or between $2^{k-1}+1$ and 2^k elements. In particular, we maintain on every iteration

- A hashmap which, for each $i \in [n]$, tracks which S_k it belongs to.
- For each S_k,
  - The cardinality of S_k.
  - $\sum_{i\in[n]} \exp([v_0]_i)[q_t]_i^{d_1}[r_t]_i^{d_2}[s_t]_i^{d_3}[v_0]_i^{d_4}[\sigma_t]^{d_5}$, for each quintuplet $0 \leq d_1+d_2+d_3+d_4+d_5 \leq d$.

The main difficulty is maintaining the invariant that there is at most one set of each rank (we call k the "rank" of a nonempty S_k). To this end, if there are two sets S_k, S'_k both with cardinality between $2^{k-1}+1$ and 2^k elements, e.g. of rank k, we allow the operation $\texttt{Merge}(S_k, S'_k)$ which creates a new S_{k+1} of rank $k+1$, containing all of the coordinates associated with either S_k or S'_k.

Whenever we perform a merge, we explicitly compute all coordinates involved in the merge, designate the largest as i^* for the updates to σ_t for that particular set, and squish if necessary to guarantee that the range of the set is at most $\frac{16\log n}{\epsilon}$; clearly, given our sparse representation $q_t, r_t, s_t, \alpha_t, \beta_t, \gamma_t$, we can appropriately modify a coordinate of say q_t to handle the squishing. We also instantiate all relevant quintuplet sums for our particular set. Furthermore, if $|S'_k|+|S_k| \leq 2^{k+1}$, `Merge` will also spend $\tilde{O}(2^{k+1}-|S'_k|-|S_k|)$ time to create "initialization credits", so that the sum of the initialization credits and the size of S_{k+1} is always exactly 2^{k+1}; these credits will be useful for our amoritized analysis. It takes time $\tilde{O}(2^{k+1})$ to update the hashmap, reinstantiate all the relevant quintuplet sums, and create credits, for S_{k+1}. We note we may need to recursively call `Merge` if there was already a set of rank $k+1$.

At the start of the n iterations, we initialize a single set of rank $\lceil \log n \rceil$, and put all of the coordinates in this set (and pay any additional cost required for initialization credits), in time $\tilde{O}(n)$. Each iteration $t+1$ will proceed in three stages. In the first stage, we compute the approximation (4.36) to the sum of exponentials as in the previous section, ignoring the effect of ζ_t. The complexity of this stage is at most $O(\log n)$ times its complexity in the previous section, because we may need to perform updates for each S_k; thus, it can be implemented in amoritized time $\tilde{O}(1)$.

In the second stage, for each coordinate in the support of ζ_t, we delete it from its corresponding S_k and instantiate a new set of rank 0, now explicitly factoring in the effect of ζ_t. Furthermore, if this causes its corresponding S_k to become rank $k-1$, e.g. if before the deletion S_k had $2^{k-1}+1$ elements, and there was already a set of rank $k-1$, we will call the `Merge` operation on the two sets of rank $k-1$. The amoritized cost of the second stage is $\tilde{O}(1)$. To see this, every time we create a new set of rank 0, we spend $\tilde{O}(1)$ time to both initialize the rank 0 set, and pay for $\tilde{O}(1)$ "deletion credits". Now, whenever we must use the `Merge` operation to create a new set of rank k, we can pay for the operation (which costs $\tilde{O}(2^k)$, both to merge and pay for new initialization credits) by

using existing credits: between when S_k was initialized and when it needed to be reinitialized due to becoming rank $k-1$, the sum of its initialization credits and the deletion credits created by removing elements is at least $\tilde{O}(2^k)$. We call any such merges Type-1 merges.

In the third stage, we recursively call `Merge`, starting from the rank 0 sets, in order to maintain the invariant that there is at most one set of any given rank. We call any such merges Type-2 merges. We claim the amoritized cost of all Type-2 merges over all n iterations is $\tilde{O}(n)$. Consider the number of times a rank k set can be created through Type-2 merges: we claim it is upper bounded by $\tilde{O}\left(\frac{n}{2^k}\right)$. If this is true, overall the complexity of the third stage is at most

$$\tilde{O}\left(\sum_{k=0}^{\lceil \log n \rceil} 2^k \frac{n}{2^k}\right) = \tilde{O}(n).$$

The number of deletions due to the ζ_t throughout n iterations is at most $\tilde{O}(n)$. Thus, it suffices to prove that between creations of rank k sets due to Type-2 merges, there must have been at least 2^{k-1} deletions. To see this, for each rank k, maintain a potential Φ_k for the sum of the cardinalities of all rank l sets for $l < k$. Each deletion increases Φ_k by at most 1. Each Type-1 merge does not increase Φ_k, because it can only cause coordinates to belong to sets which increase in rank. In order for a Type-2 merge to be used to create a rank k set, Φ_k must have been at least $2^{k-1}+2$; after the merge, it is 0, because in its creation, all rank l sets for $l < k$ must have been merged. Thus, for the potential to become large enough to require a merge again, there must have been at least 2^{k-1} deletions, as desired.

Finally, we remark that the analysis of each constituent data structure, i.e. the case when $\zeta_t = 0$ for the supported coordinates, remains correct under deletions. In particular, (4.35) may still use the original value of $[v_t]_{i^*}$ in its recursion, even if the coordinate i^* is deleted; it is easy to see that by the original boundedness of the range of supported coordinates, the analysis still holds.

Maintaining $y_{t+\frac{1}{2}}$.

In order to compute coordinates of $y_{t+\frac{1}{2}}$, we discuss approximating the sum

$$\sum_{i\in[n]} \exp\left(\left[v_{t+\frac{1}{2}}\right]_i\right).$$

It is easy to see that because $v_{t+\frac{1}{2}}$ and $(1-c)v_t$ never vary by more than a small additive constant $\frac{1}{8n}$, and furthermore we also maintain a sparsely updated representation of $v_{t+\frac{1}{2}}$ in terms of q_t, r_t, s_t, we may suitably modify the approximation (4.36) to approximate this sum. In particular, we may compute an appropriate scaling $\sigma_{t+\frac{1}{2}}$ by estimating all coordinates of cv_t by scaling some particular coordinate, and estimate the coefficients of the quintuplet sums in terms of the coefficients in the linear combination. The complexity of this computation in each step is at most $O(d^5)$ in each step, which never asymptotically dominates.

Sampling from the sum of exponentials.

Here, we discuss how to sample from the sum of the exponentials. We use the following fact about rejection sampling.

Fact 8 (Rejection sampling)**.** *Suppose P and Q are probability distributions over $[n]$, and $\frac{P[i]}{Q[i]} \in [\frac{1}{2}, 2]$ for all $i \in [n]$. Further, suppose we may sample from P in time $\tilde{O}(1)$. The following strategy samples exactly from Q in expected $O(1)$ time: sample a coordinate of i according to P, and accept with probability $\frac{Q[i]}{2P[i]}$; repeat until acceptance.*

In our setting, in each iteration P is the distribution over coordinates $i \in S_k$ proportional to $\exp([v_0]_i - \sigma_t)$, where we overload the definitions of v_0, σ_t to refer to the point the set S_k uses to approximate the Taylor expansion. We can sample from this distribution P by maintaining for each of the $\tilde{O}(1)$ sets S_k, $\sum_{i \in S_k} \exp([v_0]_i - \sigma_t)$, by initializing it with the sum when $\sigma_t = 0$, and then appropriately scaling the sums each iteration. Further, we may initialize each set S_k with a binary tree data structure for sampling proportional to $[v_0]_i$ for each $i \in S_k$, because the uniform scaling $\exp(-\sigma_t)$ does not affect this distribution. In conclusion, we sample from P by first sampling a set S_k proportional to its weight given by P, and then sampling a coordinate in the set appropriately.

We then rejection sample from P with respect to Q, the true distribution. By the invariant (4.34), this rejection sampling scheme meets the requirements to succeed in expected time $\tilde{O}(1)$, which yields the conclusion.

Cleaning up: effects of approximate sums and squishing

We first prove Lemma 60, using additional structure afforded by our data structure implementation.

Proof of Lemma 60. Recall the notation and bounds from Lemma 59,

$$\delta_t := \frac{1}{\kappa}(b - Ax_t),\ \delta^{(j)}_{t+\frac{1}{2}} := \frac{1}{\kappa}\left(b - A\left(x_t + \frac{1}{p_j}\Delta^{(j)}_t\right)\right),$$
$$\|\delta_t\|_\infty,\ \left\|\delta^{(j)}_{t+\frac{1}{2}}\right\|_\infty \leq \frac{1}{4}.$$

We write the updates to $y_{t+\frac{1}{2}}$, $y^{(j)}_{t+1}$ in the form, for $c = \frac{\epsilon}{4\kappa \log n}$:

$$y_{t+\frac{1}{2}} \propto \exp\left(\log y_t - c \log y_t - \delta_t\right),\ y^{(j)}_{t+1} \propto \exp\left(\log y_t - c \log y_{t+\frac{1}{2}} - \delta^{(j)}_{t+\frac{1}{2}}\right).$$

Letting vectors v_t, $v_{t+\frac{1}{2}}$ satisfy $y_t \propto \exp(v_t)$, $y_{t+\frac{1}{2}} \propto \exp(v_{t+\frac{1}{2}})$ for all t, the goal of this lemma is to show that $\|v_{t+1} - v_t\|_\infty$, $\left\|v_{t+\frac{1}{2}} - v_t\right\|_\infty$ are both bounded by 1. Indeed, we have the recursion

$$v_{t+\frac{1}{2}} = (1-c)v_t - \delta_t,\ v_{t+1} = v_t - cv_{t+\frac{1}{2}} - \delta^{(j)}_{t+\frac{1}{2}}.$$

Based on the bounds on δ_t and $\delta^{(j)}_{t+\frac{1}{2}}$, it suffices to show that $\|cv_t\|_\infty$, $\left\|cv_{t+\frac{1}{2}}\right\|_\infty \leq \frac{3}{4}$. By the squishing operations performed by the data structure, at the beginning of n iterations (when the data structure is restarted), the range of v_t is contained in $[0, 16\log n/\epsilon]$.

Over the course of n iterations, this fact is preserved for each particular data structure supporting a set of coordinates. Moreover, we recall that we used squishing whenever we initialize a new data structure in the binomial heap to maintain the fact that the additive range over all coordinates is $O(\log n/\epsilon)$. The final issue which may come up is the additive drift caused by the vectors δ_t or $\delta^{(j)}_{t+\frac{1}{2}}$; however, over the course of n iterations, this can only shift the largest coordinate of v_t by $n/4$. Altogether, it is clear we may assume $\|v_t\|_\infty < \frac{n}{4} + \frac{33\log n}{\epsilon} \ll \frac{3}{4c}$; the conclusion follows. Similarly, we inductively have $\left\|v_{t+\frac{1}{2}}\right\|_\infty \ll \frac{3}{4c}$ by bounding its difference to v_t. □

We now consider the effect of only approximately maintaining the sums of exponentials in our algorithm, and applying squishing. In particular, the inequality in Lemma 62 only holds up to an additive constant. The additive error comes into play in two ways: the first-order optimality condition only holds up to the discrepancy between y_t and $\breve{y}_t$, and each time we apply squishing affects the value of $\mathbb{E}\left[V_{z_{t+1}}(\tilde{z})\right]$. Regarding the former, all problem parameters and the number of phases of our algorithm are all bounded by a small polynomial in n, so the guarantees of `Y-Oracle.Coord` mean that the cumulative error does not amount to more than $n^{-90} \ll \epsilon$ (we assume $\epsilon > n^{-3}$, else an interior point method achieves our stated runtime). Similarly, regarding the latter, Lemma 255 implies that even if we squish $O(n)$ coordinates each iteration, the cumulative error in the Bregman divergence does not amount to more than n^{-90}.

Chapter 5

Semi-Streaming Combinatorial Optimization

This chapter is based on [293, 39], with Sepehr Assadi, Arun Jambulapati, Yujia Jin, and Aaron Sidford.

5.1 Introduction

We study the fundamental problem of designing semi-streaming algorithms for maximum cardinality matching in bipartite graphs (MCM). We consider the insertion-only problem where an unknown n-vertex m-edge undirected bipartite graph is presented as a stream of edge insertions. Our goal is to compute an ϵ-approximate MCM (a matching with value $\geq 1-\epsilon$ times the optimum) in the semi-streaming model [221], i.e. using $\widetilde{O}(n)$ space,[1] and as few passes as possible.

Due to its canonical and prevalent nature, MCM is well-studied in the graph streaming literature. In a single pass, a simple greedy algorithm achieves a $\frac{1}{2}$-approximation in $O(n)$ space (cf. Lemma 71) and obtaining better than a $(\frac{1}{1+\ln 2}) \approx 0.59$ approximation requires $n^{1+\Omega(1/\log\log(n))}$ space [309] (see also [240, 308]). Correspondingly, there is a line of research on designing multi-pass algorithms for ϵ-approximating MCM. In this chapter, we focus on $\widetilde{O}(\text{poly}(\epsilon^{-1}))$-pass semi-streaming algorithms. Here, the state-of-the-art includes the $O(\epsilon^{-2})$-pass $O(n)$-space algorithm of [42] (see also [12]) and the $O(\epsilon^{-1}\log n)$-pass $\widetilde{O}(n\cdot\text{poly}(\epsilon^{-1}))$-space algorithm of [13].[2]

For different ranges of ϵ, these semi-streaming MCM algorithms provide non-trivial trade-offs

[1] Throughout, we measure space in machine words of size $\Theta(\log n)$ and use $\widetilde{O}$ to hide factors polylogarithmic in n, ϵ^{-1}, and $C_{\max}$, the largest edge weight of the relevant problem. In all our applications, without loss of generality, ϵ^{-1} and $C_{\max}$ are $O(\text{poly}(n))$. Consequently, in our applications an $\widetilde{O}(n)$ space algorithm is a semi-streaming, i.e. $O(n\cdot\text{poly}(\log(n))$, algorithm for our applications.

[2] Streaming MCM has been studied under several other variants and parameter settings; see Section 5.1.4 for details.

between space and pass complexity. While [13] provides an improved $\widetilde{O}(\epsilon^{-1})$ passes compared to $O(\epsilon^{-2})$ in [42], the space complexity of [13] is larger by $\text{poly}(\epsilon^{-1})$ factors[3] and therefore only adheres to the semi-streaming restriction of using $\widetilde{O}(n)$ space when $\epsilon \geq (\text{polylog}\,(n))^{-1}$. The central question we address in this paper is whether this trade-off is necessary:

Is it possible to obtain an $\widetilde{O}(\epsilon^{-1})$-pass semi-streaming algorithm for ϵ-approximating maximum bipartite matchings for the entire range of $\epsilon > 0$?

In this paper we answer this question in the affirmative. We provide multiple ways to leverage recent advances in continuous and combinatorial optimization for this goal. For example, in Appendix D.5 we show how to obtain this result by a careful application of recent results on the box-constrained Newton method [145] and streaming sparsification [388].

Our main result is a simple algorithm that further improves the space complexity. Our starting point is to demonstrate that the advances in area convexity [483] (see also Section 2.5), which achieved a state-of-the-art algorithm for approximate maximum flow, are applicable to matching and transportation problems as well through similar "soft Lagrangian" formulations. By directly applying these tools, we develop a direct, efficient parallel solver for optimal transport in Appendix D.1. We further leverage these new optimization methods to provide an $O(n)$-space and $O(\epsilon^{-1} \cdot \log(\epsilon^{-1}) \cdot \log n)$-pass algorithm for semi-streaming bipartite matching. This space-dependency is optimal, as $\Omega(n)$ space is needed to simply output the final matching. Further, this algorithm is deterministic, in contrast to the previous $\widetilde{O}(\epsilon^{-1})$-pass algorithm of [13], and the method's runtime is $\widetilde{O}(m \cdot \epsilon^{-1})$.

To obtain this result we introduce a more general algorithmic framework of independent interest. We design semi-streaming algorithms which can approximately determine the value of and produce low-space *implicit* fractional solutions to linear programs, given in the form of box-simplex games, when rows of the constraint matrix are presented in a stream. We obtain our MCM results by applying this optimization method to a linear programming representation of the problem and show that this implicit solution can be converted into a sparse solution in low space.

We believe our results demonstrate the power of recent optimization advances for solving streaming problems. Beyond resolving the space complexity of $\widetilde{O}(\epsilon^{-1})$-pass algorithms, we show that our techniques extend to yield the following additional results.

- *Exact MCM in $o(n)$ passes for all densities.* Recent work [368] asked whether the $O(n \log n)$-pass semi-streaming algorithm for *exact* MCMs following from a careful implementation of the classical Hopcroft-Karp algorithm [274] is improvable. The authors designed an $\widetilde{O}(n)$-space algorithm based on interior-point methods using $\widetilde{O}(\sqrt{m})$ passes, which is $o(n)$ passes except for in dense graphs. By combining our approximate MCM method with recent advances in

[3] We remark that the introduction and main theorem statements of [13] are written for constant ϵ. However, for subconstant ϵ it is straightforward to see their space requirement incurs a $\text{poly}(\epsilon^{-1})$ dependence.

streaming reachability [290], we obtain an $O(n^{\frac{3}{4}+o(1)})$-pass, $\widetilde{O}(n)$-space algorithm for exact MCM, thereby answering the main open problem of [368] for all densities. Interestingly, this improvement is a direct byproduct of resolving our motivating question as the space-pass trade-offs in [13] and [42] prevent either method from obtaining nontrivial semi-streaming exact MCM algorithms even when combined with [290] (see Remark 7).

- *Optimal transport.* Closely-related to computing MCMs, the discrete optimal transport problem has received widespread recent interest due to applications in machine learning [151, 25]. Several recent works have designed different optimization algorithms (influenced by developments for matching) [450, 82] solving the problem on a support size of n using $\widetilde{O}(n^2\epsilon^{-1})$ total work; we give a presentation of a new, direct method attaining this rate in Appendix D.1. In this chapter, we further provide the first semi-streaming algorithm in this setting within $\widetilde{O}(\epsilon^{-1})$ passes, $O(n)$ space, and comparable work.

- *Transshipment and shortest path.* Yet another application of our method is to solving the transshipment problem, a type of uncapacitated minimum cost flow problem, on undirected graphs. Several recent works have focused on this problem in streaming and parallel models [67, 358, 29] and used this to obtain approximate shortest path algorithms. We provide a semi-streaming algorithm that achieves an ϵ-approximation to transshipment within $\widetilde{O}(\epsilon^{-1})$ passes, $\widetilde{O}(n)$ space, and $\widetilde{O}(m\epsilon^{-1})$ work. As a direct corollary of this result, we obtain ϵ-approximate semi-streaming algorithms for the s-t shortest path problem with same complexities. Our results on transshipment and shortest path improve upon the previous best semi-streaming algorithms of [67] by $\widetilde{O}(\epsilon^{-1})$ factors in passes and work.

5.1.1 Problem setup

The basic (and motivating) problem we consider in this work is that of computing an approximate MCM in bipartite graph, $G = (V, E)$, given as an insertion-only stream defined as follows.

Definition 17 (Semi-streaming graph model)**.** *In the* semi-streaming graph model*, a graph $G = (V, E)$ with vertex set $V = [n]$ is presented to the algorithm as an arbitrarily ordered stream of edges (u, v) for $u, v \in V$ (the tuple contains the weight for weighted graphs). The algorithm can read this stream in sequential* passes *and is constrained to use $\widetilde{O}(n)$ space.*

We typically let $n = |V|$ and $m = |E|$. M^* denotes the the size of the MCM in G, and $L, R \subseteq V$ with $L \cap R = \emptyset$ and $L \cup R = V$ denotes the vertex bipartition. A key goal of this paper is to efficiently compute approximate MCMs in the semi-streaming graph model. Formally, we refer to any flow $x \in [0, 1]^E$ such that the total flow adjacent to any $v \in V$ is ≤ 1 as a *fractional matching* and call the matching integral if $x \in \{0, 1\}^E$; we say such a matching is a ϵ-approximate MCM if $\|x\|_1 \geq (1 - \epsilon)M^*$. The outputs of our algorithms typically include both *the approximate problem*

value and *an approximate solution*. For MCM problems with value M^*, the algorithm returns this value within a $(1-\epsilon)$ factor and an integral matching that achieves this approximate value.

To obtain our main result, we develop streaming algorithms for a more general set of problems. These problems are linear programming in the form of *box-simplex games*, i.e. bilinear minimax games between a *box*, or ℓ_∞-constrained player, and *simplex*, or nonnegative ℓ_1-constrained player:

$$\min_{x\in\Delta^m}\max_{y\in[-1,1]^n} y^\top \mathbf{A}^\top x + c^\top x - b^\top y. \tag{5.1}$$

Maximizing over y for a fixed x, problem (5.1) is equivalent to the following variant of ℓ_1-regression:

$$\min_{x\in\Delta^m} c^\top x + \left\|\mathbf{A}^\top x - b\right\|_1. \tag{5.2}$$

We call $x\in\Delta^m$ an ϵ-approximate minimizer for (5.2) if it satisfies $c^\top x+\left\|\mathbf{A}^\top x-b\right\|_1 \le \min_{x'\in\Delta^m} c^\top x'+\left\|\mathbf{A}^\top x'-b\right\|_1+\epsilon$. We relate (5.1), (5.2) to MCM via reduction: we construct an appropriate instance of (5.1), (5.2) such that any approximate minimizer to (5.2) can be efficiently converted into an approximate MCM. We note the semi-streaming solver we develop in Section 5.3 attains an approximate solution for (5.1), but in applications we only ever use that the x block approximately minimizes (5.2).

Interestingly, we provide a general streaming algorithm for approximating the value of (5.1), (5.2) (and implicitly representing an optimal solution) in the following access model.

Definition 18 (Semi-streaming matrix model)**.** *In the* semi-streaming matrix model*, a matrix $\mathbf{A}\in\mathbb{R}^{m\times n}$ and a vector $c\in\mathbb{R}^m$ are presented to the algorithm respectively as an arbitrarily ordered stream of rows $\{\mathbf{A}_{i:}\}_{i\in[m]}$, and a similarly ordered stream of coordinates $\{c_i\}_{i\in[m]}$ (the ordering on $[m]$ is arbitrary, but each $\{\mathbf{A}_{i:},c_i\}$ pair is given together). The algorithm can read this stream in sequential* passes *and is constrained to use $\widetilde{O}(n)$ space.*

By choosing $\mathbf{A}$ to be the incidence matrix of a graph and setting c to be the weights if applicable, Definition 18 generalizes Definition 17. Note that obtaining semi-streaming algorithms for this problem is nontrivial only when $m=\omega(n)$.

Further applications. In Section 5.5, we study additional combinatorial optimization problems which are also reducible to box-simplex games. In the problem of *discrete optimal transportation*, a complete bipartite graph $G=(V,E)$ where $E=L\times R$ has associated demands $\ell\in\Delta^L$, $r\in\Delta^R$, and edge costs $c\in\mathbb{R}^E$. The demands ℓ, r are probability distributions on domains of equal size, and the goal is to compute the minimum cost transport plan which attains the prescribed marginals on the vertex sets L and R. The semi-streaming access model gives the cost of each edge as it is presented; note that the marginals ℓ, r can be stored explicitly in $O(n)$ space. In this setting, our designed algorithm gives an approximate value of the optimal transportation, as well as a fractional transportation plan $x\in\Delta^E$ that meets the demands exactly (has the correct marginals at every

vertex) and achieves an approximately optimal value. Using our techniques we also give a result for computing maximum weight matchings (MWMs), which is competitive with the state-of-the-art when the optimal matching is highly saturated (e.g. when the MWM value is within a constant factor of $\|w\|_\infty n$ in a graph with weights w, the largest possible value).

Finally, in Section 5.6 we go beyond "bipartite matching-type" applications of our framework and consider the transshipment problem. Given a demand vector $d \in \mathbb{R}^V$ on vertices of an undirected graph with non-negative edge weights, the goal of transshipment is to route a flow satisfying this demand while minimizing the sum of weighted flow magnitude over all edges. Among other applications, setting demands of any pairs of vertices s, t in the transshipment problem to 1 and -1 respectively, and other vertices to zero, reduces the s-t shortest path problem to transshipment. Combining our techniques with standard tools, we give a semi-streaming algorithm for transshipment improving the state-of-the-art by roughly an ϵ^{-1}-factor in the number of passes and work.

5.1.2 Our results

Here we state several key results of our paper. The following is our main result.

Theorem 26 (Approximate MCM)**.** *There is a deterministic semi-streaming algorithm which given any bipartite $G = (V, E)$ with $|V| = n, |E| = m$, finds a ϵ-multiplicatively approximate MCM in $O\big(\log n \cdot \log(\epsilon^{-1}) \cdot \epsilon^{-1}\big)$ passes, $O(n)$ space, and $O(m \log^2 n \cdot \epsilon^{-1})$ total work.*

The result most closely related to our Theorem 26 is the semi-streaming algorithm of [13] that achieves a ϵ-approximate MCM using $O(\epsilon^{-1} \cdot \log n)$ passes and $O(n \cdot \text{poly}(\log n, \epsilon^{-1}))$ space. Theorem 26 improves the space dependency to the optimal bound of $O(n)$ at a cost of a $\log \epsilon^{-1}$ factor in the number of passes. Additionally, our algorithm is arguably more straightforward than [13] and is deterministic (the randomized algorithm of [13] works with high probability).

An interesting feature of our algorithm in Theorem 26 its space complexity has no dependence on the parameter ϵ. Consequently, this algorithm can obtain a *very accurate* approximation of MCM in $O(n)$ space, albeit at a cost of a large pass complexity. We leverage this feature and complement the algorithm with standard augmenting path approaches for MCM, implemented efficiently using recent advances on PRAM (and semi-streaming) algorithms for directed reachability problem in [290]. This yields the following result for exactly computing an *exact* MCM, resolving an open problem of [368] on obtaining $o(n)$-pass semi-streaming algorithms for exact MCM (note however that [368] handles weights, while we primarily focus on MCM).

Theorem 27 (Exact MCM)**.** *There is a randomized semi-streaming algorithm which given any bipartite $G = (V, E)$ with $|V| = n$, finds an exact MCM with high probability in $O(n^{\frac{3}{4}+o(1)})$ passes.*

Our techniques in obtaining Theorem 26 yields results for several other semi-streaming combinatorial optimization problems, such as the following.

Theorem 28 (Optimal transport)**.** *There is a deterministic semi-streaming algorithm which given any optimal transport instance on a complete bipartite graph on $V = L \cup R$, costs $c \in \mathbb{R}^E_{\geq 0}$, and two sets of demands $\ell \in \Delta^L$, $r \in \Delta^R$, finds an $\epsilon \|c\|_\infty$-additive approximate optimal transport plan using $O\left(\epsilon^{-1} \log n \log \epsilon^{-1}\right)$ passes, $O(n)$ space, and $O\left(n^2 \epsilon^{-1} \log n \log \epsilon^{-1}\right)$ work.*

To our knowledge, this is the first non-trivial semi-streaming algorithm for optimal transport (however, it is plausible one can use existing semi-streaming algorithms for b-matching such as [12] in conjunction with the reduction of [82] to obtain non-trivial semi-streaming algorithms.) We further give an algorithm for MWM using similar techniques (see Section 5.5.3).

Finally, yet another application of our techniques is to the transshipment problem. We define transshipment and prove the following result in Section 5.6.

Theorem 29 (Approximate transshipment and shortest path)**.** *There is a randomized semi-streaming algorithm which given weighted undirected $G = (V, E, w)$ and demand vector $d \in \mathbb{R}^n$, finds an ϵ-multiplicatively approximate minimizer to the minimum transshipment cost in $O(\log^{O(1)} n \cdot \epsilon^{-1})$ passes, $O(n \log^{O(1)} n)$ space, and $O(m \log^{O(1)} n \cdot \epsilon^{-1})$ total work with high probability in n. This, in particular, yields a semi-streaming algorithm that ϵ-multiplicative approximates the s-t shortest path problem with the same pass, space, and work complexities.*

5.1.3 Our techniques

In this section, we overview our approach. We first give an overview of a natural reformulation of MCM as an appropriate ℓ_1-regression problem. We then discuss the two main components of our semi-streaming MCM implementation, which are representative of our overall framework in other applications. The first component is a low-space solver for general ℓ_1 regression problems in the form of box-simplex games, which returns an implicit representation of an approximate (possibly dense) solution. The second component is a rounding procedure which sparsifies implicit approximate solutions and rounds them to feasibility and (in certain contexts) integrality.

Our main contribution is to give efficient semi-streaming implementations and applications of each of these components. Our semi-streaming methods for optimal transport, MWM, and transshipment similarly follow from our framework, via appropriately formulating these applications as regression problems, applying our semi-streaming tools to approximate the regression problem, and rounding the approximate solution to an explicit sparse solution of no worse objective value.

MCM as ℓ_1-regression. The first piece of our MCM algorithm is to formulate an appropriate ℓ_1-regression problem, whose approximate solution also implies an approximately optimal matching. Variants of this (fairly natural) reduction have found previous use, but its proof of correctness is suggestive of our overall approach, so we describe it here for completeness. To obtain this reduction, we follow a "penalizing overflow" approach which has also found recent applications to approximate maximum flow [482, 318, 441, 483, 486]. We remark that in Appendix D.1, this strategy is also

followed to develop a solver for approximate optimal transport based on box-simplex games.

Recall that the (fractional) MCM problem asks to find a flow $x \in \mathbb{R}^E_{\geq 0}$ with maximal ℓ_1 norm, such that no vertex has more than 1 unit of flow adjacent to it (the "matching constraints"). In linear-algebraic terms, letting $\mathbf{B} \in \{0,1\}^{E \times V}$ be the (unsigned) incidence matrix for the graph, the problem asks to maximize $\|x\|_1$ subject to the matching constraints, $\mathbf{B}^\top x \leq \mathbf{1}$ entrywise. The penalizing overflow approach relaxes this problem to an appropriate *unconstrained* optimization problem, by appropriately penalizing matching constraint violations (rather than treating them as hard constraints), and using combinatorial structure to argue solving the unconstrained problem suffices. We now describe how to instantiate this reduction for the MCM problem.

We begin by obtaining M, a 2-approximation to the MCM value M^*, in a single pass using a greedy algorithm (Lemma 71), which is used to parameterize our ℓ_1-regression problem. The next step in our reduction is a rounding procedure adapted from [25], which gives an algorithm showing that any flow $\hat{x} \in \mathbb{R}^E_{\geq 0}$ (not necessarily a feasible matching) can be converted into a feasible (fractional) matching $\tilde{x}$ on the same support, with size loss proportional to the total amount of constraint violations $\text{Overflow}_d(\hat{x}) := \left\|(\mathbf{B}^\top \hat{x} - d)_+\right\|_1$, where subscripting + denotes the nonnegative part and $\mathbf{B}^\top \hat{x} - d$ encodes constraint violations (cf. Lemma 73). Using this observation, we demonstrate that to find a large fractional matching it suffices to solve the following ℓ_1-regression problem

$$\min_{x \in \Delta^{\widetilde{E}}} - \langle \mathbf{1}_E, 2Mx \rangle + 2M \left\|(\mathbf{B}^\top \hat{x} - d)_+\right\|_1$$

to ϵM additive accuracy, where the graph $\widetilde{G} := (\widetilde{V}, \widetilde{E})$ modifies our original $G = (V, E)$ by adding one dummy edge, and the restriction of $2Mx$ to the original edges E plays the role of our desired matching; these complications are to fix the ℓ_1 norm of our x variable, to put it in a form suitable for canonical optimization techniques. Further, to explicitly return an approximate MCM, we show that we can take an implicit representation of our approximate optimizer x, and use dynamic data structures in $O(n)$ space to return $x' \in \Delta^{\widetilde{E}}$ on a sparse support satisfying $\mathbf{B}^\top x = \mathbf{B}^\top x'$. We can then explicitly compute an approximately optimal MCM on this sparse support.

We now discuss the technical tools which go into developing both of these pieces: our semi-streaming ℓ_1-regression solver, and obtaining an approximately optimal matching on a sparse support.

Iterative methods in low space. Our semi-streaming algorithm for approximately solving (5.1), (5.2) follows recent improved algorithms solving these box-simplex problems in the standard (non-streaming) setting. Classical first-order methods for solving these problems, such as smoothing [419] or decoupled extragradient methods [415, 420] have iteration counts incurring either a quadratic dependence on the error ratio $\|\mathbf{A}\|_\infty \epsilon^{-1}$ or growing polynomially with the dimension (each iteration running in time linear in the sparsity of the matrix $\mathbf{A}$). A line of work, beginning with a breakthrough by [483] (which solved a more general problem), has designed improved first-order methods bypassing

both of these bottlenecks, with an iteration count of $\widetilde{O}(\|\mathbf{A}\|_\infty \epsilon^{-1})$. The [483] algorithm is analyzed in the box-simplex setting we study using a different regularizer in Appendix D.1; we recall that Section 2.5 demonstrates that the classical methods of [415, 420] could also be analyzed using the regularizer of Appendix D.1, and obtain a matching rate.

Our contribution to this line of work is the crucial observation that the [293] algorithm can deterministically be implemented implicitly in $O(n)$ space under the semi-streaming matrix model of Definition 18. This observation is perhaps surprising, since the simplex variable throughout the algorithm is typically fully-dense, and we do not subsample to preserve sparsity. We instead utilize the recursive structure of the box-simplex algorithms to give a low-space representation of simplex iterates, such that the actual iterate can be accessed in $O(1)$ passes.

We begin by giving a high-level overview of the algorithm of Appendix D.1; we remark that we will use this algorithm (as opposed to [483]) because its analysis is convenient for obtaining tighter dependencies on problem parameters for our applications. The algorithm of Appendix D.1 runs in $\widetilde{O}(\|\mathbf{A}\|_\infty \epsilon^{-1})$ iterations, each computing a box-simplex pair $(x_t, y_t) \in \Delta^m \times [-1,1]^n$. Each box-simplex pair is an approximate solution to a regularized linear objective of the form (hereinafter we let $|\cdot|$ applied to matrix or vector arguments act entrywise)

$$\min_{x\in\Delta^m, y\in[-1,1]^n} \langle g^{\mathsf{x}}, x\rangle + \langle g^{\mathsf{y}}, y\rangle + r(x,y), \text{ where } r(x,y) := x^\top |\mathbf{A}|(y^2) + 10\,\|\mathbf{A}\|_\infty \sum_{i\in[m]} x_i \log x_i. \quad (5.3)$$

Typically, closed-form solutions to the above displayed problem are difficult to compute, because the regularizer r couples the two blocks. However, Appendix D.1 provides an alternating minimization subroutine which rapidly converges to an approximate solution, which suffices for our purposes.

Our first observation is that in every iteration, the simplex variable x_t, is always proportional to a vector $\exp(\mathbf{A}v_t + |\mathbf{A}|u_t + \lambda_t c)$ for n-dimensional vectors v_t, u_t and a scalar λ_t. This is because throughout the alternating subroutine for solving (5.3), every simplex iterate is the minimizer to a entropy-regularized linear objective, where the linear term is with respect to a linear combination of two pieces. The first is a gradient operator of the box-simplex problem, of the form $\left(\mathbf{A}y + c,\ b - \mathbf{A}^\top x\right)$ for iterate x, y, and the second is the gradient of our regularizer (which on the x side is either a linear combination of $\mathbf{A}$ and $|\mathbf{A}|$ applied to n-dimensional vectors or an entropic gradient; the latter yields a logarithm of previous iterates, which have an implicit representation inductively). Combining shows all linear terms have the form $\mathbf{A}v + |\mathbf{A}|u + \lambda c$, enabling our recursion.

Our second observation is that for any (v_t, λ_t) pair, computing the vectors $\mathbf{A}^\top \exp(\mathbf{A}v_t + |\mathbf{A}|u_t + \lambda_t c)$ and $|\mathbf{A}|^\top \exp(\mathbf{A}v_t + |\mathbf{A}|u_t + \lambda_t c)$ (required in the gradient operator) can be performed in a single semi-streaming pass, and $O(n)$ space, because the resulting matrix-vector product decouples by edge. Combining these pieces implies a deterministic semi-streaming iterative method for approximating the value of box-simplex games, as stated in the following (cf. Section 5.2 for definitions).

Theorem 30. *There is a deterministic algorithm obtaining an ϵ-additive approximation to the value*

of (5.1), (5.2) *in* $O(\frac{\|\mathbf{A}\|_\infty}{\epsilon} \cdot \log(m) \log(\max(\|\mathbf{A}\|_\infty, \|b\|_1) \cdot \epsilon^{-1}))$ *passes and* $O(n)$ *space. Moreover, the algorithm outputs a stream of nonnegative* 1*-sparse vectors in* $\mathbb{R}^m$ *of length* $O(m \cdot \frac{\|\mathbf{A}\|_\infty \log m}{\epsilon})$ *whose sum is* $\bar{x}$*, an* ϵ*-approximate optimizer to* (5.2)*.*

Theorem 30 gives an $O(n)$ space, $\widetilde{O}\left(\|\mathbf{A}\|_\infty \epsilon^{-1}\right)$-pass algorithm for obtaining ϵ-additive approximations to the value of box-simplex games. The remainder of our work is in applying this method to matching problems and providing semi-streaming implementations for rounding the solution.

Detecting a sparse support via cycle cancelling. So far, we have given a way of running a solver for the problems (5.1), (5.2) entirely in $O(n)$ space. However, it remains to discuss how to extract an MCM from these iterates; although we have a low-space implicit representation, the iterates themselves remain fully dense, and to write them down explicitly for postprocessing is too expensive. To get around this obstacle, we give a data structure which is compatible with our implicit representation, and can take a flow x given as a sequence of edge additions and obtain an $O(n)$-sparse x' such that $\|x\|_1 = \|x'\|_1$ and $\mathbf{B}^\top x = \mathbf{B}^\top x'$ (i.e. the "quality of the matching" is unaffected), but we can afford to explicitly write down x'. Our data structure is based on cycle cancelling via link/cut trees, a standard technique in the literature following e.g. [488, 243].

Ultimately, we show our framework of implicitly solving a box-simplex game and feeding the implicit representation of its solution into a link/cut tree data structure to detect a sparse support containing a good-quality approximate matching is quite flexible. We leverage this framework to also solve semi-streaming variants of optimal transport, weighted matching, and transshipment problems.

5.1.4 Previous work

Maximum matching. MCM and its many variants are arguably the most extensively studied problems in the graph streaming literature; see, e.g. [221, 387, 240, 327, 213, 308, 12, 310, 41, 439, 40, 305, 326, 506, 13, 37, 228, 218, 72, 312, 309] and references therein.

The first multi-pass MCM algorithm was given in [221]—alongside the introduction of the semi-streaming model itself—achieving a $(\frac{2}{3} - \epsilon)$-approximation in $O(\epsilon^{-1} \log \epsilon^{-1})$ passes. The first $(1 - \epsilon)$-approximation was then achieved by [387], for both bipartite and non-bipartite graphs, using $(\epsilon^{-1})^{O(\epsilon^{-1})}$ passes. The pass-complexity of this result for bipartite graphs was improved in [213, 12, 308, 13, 42], leading to the aforementioned state-of-the-art results of [13, 42]. More recently, [368] designed a semi-streaming algorithm for this problem with $\widetilde{O}(\sqrt{m} \cdot \log(\epsilon^{-1}))$ passes, which achieves better pass-dependence on ϵ, at a cost of a polynomial dependence on n, m in the number of passes. Finally, [224] very recently gave the first $\text{poly}(\epsilon^{-1})$-pass algorithm for this problem for non-bipartite graphs.

The aforementioned works focus on insertion-only streams. When allowing deletions to the stream, i.e. in the turnstile streaming model, the problem becomes significantly harder. In particular, the best approximation ratio possible to MCM in a single-pass is provably only $\Omega(n^{1/3})$ [41, 158]

which is achieved by the algorithms of [41, 138] (see also [325]). Nevertheless, many multi-pass streaming algorithms for MCM can be extended to turnstile streams at the cost of an additional $O(\log n)$ multiplicative factor in their number of passes [14]. It is worth noting that the algorithm of [13] directly works in the turnstile streaming model.

Alongside MCM, maximum weight matching (MWM) has also been studied extensively. Some key results include $(\frac{1}{2}-\epsilon)$-approximation in a single pass [439], and $(1-\epsilon)$-approximation in $(\epsilon^{-1})^{O(\epsilon^{-2})}$ [228], $O(\epsilon^{-2}\log\epsilon^{-1})$ [12], and $O(\epsilon^{-1}\log n)$ [13] passes. The exact algorithm of [368] in $\widetilde{O}(\sqrt{m})$ also works for the more general MWM problem.

We also note that several other streaming variants of MCM and MWM have also been studied, including random-order streams [327, 218, 326, 310, 37, 72, 38], $n^{1+\Omega(1)}$-space algorithms [37, 13, 69], or *size* estimation in $o(n)$-space [310, 216, 40, 389]. Reviewing this vast literature is beyond the scope of this paper and we refer the interested reader to these references.

As in many computational models, MCM and MWM have been a testbed for development and adaptation of various algorithmic tools for streaming including local-ratio algorithms [439, 228], augmenting paths [221, 387, 213], water-filling or auction algorithms [308, 42], iterative methods e.g. multiplicative weights [12, 13] and interior-point methods [368].

Optimal transport. The design of efficient algorithms for discrete optimal transportation has received widespread attention in recent years [151, 25, 24, 365], due in large part to its many applications in modern statistical modeling and machine learning. To our knowledge, ours is the first result which applies to this problem in the semi-streaming setting (though as we remark earlier, it may be possible to combine prior-work to achieve non-trivial results). Our approach for optimal transport follows straightforwardly from our algorithms for solving the fractional maximum matching problem, via known reductions in the literature.

Transshipment and shortest path. The transshipment problem in the semi-streaming was previously studied in [67] which designed an $\widetilde{O}(\epsilon^{-2})$ pass semi-streaming algorithm for this problem using gradient descent. This algorithm in turn allowed [67] to obtain ϵ-approximate semi-streaming algorithms for the (single-source) shortest path problem in $\widetilde{O}(\epsilon^{-2})$ passes, improving upon the $n^{o(1)}$-pass algorithm of [269] that required $n^{1+o(1)}$-space (more than the restriction of semi-streaming algorithms). More recently, [117] gave a semi-streaming algorithm for finding exact single-source shortest paths in $\widetilde{O}(\sqrt{n})$ passes. On the lower bound front, [261] showed that ϵ-approximate semi-streaming algorithms for the s-t shortest path problem require $\Omega(\epsilon^{-1})$ passes (see also [43, 125] for more advances on this front).

5.2 Preliminaries

General notation. We use $[n]$ to denote $\{1, ..., n\}$ and $\mathbf{0}_n$ and $\mathbf{1}_n$ to denote the all-zeros and all-ones vectors in $\mathbb{R}^n$. An event holds with high probability if it holds with probability $\geq 1 - n^{-c}$ for $c > 0$

(in our results, c can be chosen to be arbitrarily large affecting guarantees by constant factors). We use $\text{nnz}(\mathbf{A})$ to denote the number of nonzero entries in a matrix $\mathbf{A}$, and its row i and column j are denoted $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$. We say scalar $\alpha > 0$ is an ϵ-multiplicative approximation to scalar $\beta > 0$ if $|\frac{\alpha}{\beta} - 1| \leq \epsilon$; similarly, we say it is an ϵ-additive approximation if $|\alpha - \beta| \leq \epsilon$. For a vector v, we use v_+ or $(v)_+$ to denote the vector which entrywise has $[v_+]_i = \max(v_i, 0)$, and use $v_S \in \mathbb{R}^n$ to denote the vector which zeroes out entries of v in $[n] \setminus S$ for $S \subseteq [n]$.

Norms. We let $\|v\|_p$ denote the ℓ_p norm of vector v, and $\|\mathbf{M}\|_p$ denote the ℓ_p operator norm of a matrix. In particular, $\|\mathbf{M}\|_\infty$ is the maximum ℓ_1 norm over rows of $\mathbf{M}$, and $\|\mathbf{M}\|_1$ is the maximum ℓ_1 norm over columns. We let $\Delta^m \subset \mathbb{R}^m_{\geq 0}$ denote the nonnegative simplex, so $x \in \Delta^m \iff \|x\|_1 = 1$ and entrywise $x \geq 0$. Finally, we let $\|v\|_0$ be the number of nonzero entries of a vector v.

Graphs. We denote a graph by $G = (V, E)$, and define $n = |V|$ and $m = |E|$ when context is clear. We refer to the independent vertex subsets of a bipartite graph by L and R. We denote the (unsigned edge-vertex) incidence matrix of a graph by $\mathbf{B} \in \mathbb{R}^{E \times V}_{\geq 0}$, where $\mathbf{B}_{ev} = 1$, if and only if $v \in V$ is an endpoint of e and is zero otherwise. We refer to any assignment of values to edges $f \in \mathbb{R}^E$ as a *flow*. When the graph is weighted (with associated edge weights $w \in \mathbb{R}^E_{\geq 0}$), we refer to the graph by $G = (V, E, w)$.

Computation model. Throughout, we operate in the standard word RAM model of computation, where basic arithmetic operations on $O(\log n)$-bit words can be performed in constant time. For weighted graphs, we assume all weights can be stored in $O(1)$ words, and all results concerning additional memory overhead count the number of additional words necessary for algorithms. We also assume always that $\epsilon^{-1} = O(\text{poly}(n))$. We measure space overhead complexity by the number of words used. In other related computational models, this may increase our space or work complexities by a logarithmic factor.

5.3 Box-simplex games in low space

In this section, we provide semi-streaming algorithms (cf. Definition 18) for approximately solving box-simplex bilinear games of the form (5.1), which will yield approximate minimizers for the ℓ_1 regression problem. The complexity of our algorithms depend on the quantity $\|\mathbf{A}\|_\infty$. We prove in this section that we can implement recent improved algorithms for problems of the form (5.1) based on *area-convex regularization* in the semi-streaming model in low memory, and give the guarantees as Theorem 30.

Our algorithm for Theorem 30 is a low-space implementation of a primal-dual algorithm inspired by [483], specialized to box-simplex games (as analyzed in Appendix D.1). Our main algorithm, Algorithm 13, follows the framework of Appendix D.1, which analyzes the convergence of a dual extrapolation scheme for solving (5.1). Lines 8 and 11 of the algorithm are implemented with AltMin

(Algorithm 14), an alternating minimization subroutine also provided in Appendix D.1. Finally, we note that we peform pre-processing and post-processing of the problem in Lines 1 and 2, so that our final guarantees only depend on properties of $\mathbf{A}$ (and not b or c). The result of these steps only affect the problem by a constant scalar shift, and can be implemented in one semi-streaming pass (more detail is given as Lemma 256 in Appendix D.1.7).

Algorithm 13: $\mathsf{ACDualEx}(\mathbf{A}, b, c, \epsilon, T)$

1 **Input:** $\mathbf{A} \in \mathbb{R}^{m \times n}_{\geq 0}$, $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, $0 \leq \epsilon \leq \|\mathbf{A}\|_\infty$, $T \in \mathbb{N}$ Remove all coordinates $i \in [m]$ where $c_i > \min_{i^* \in [m]} c_{i^*} + 2\|\mathbf{A}\|_\infty$ from c and corresponding rows of $\mathbf{A}$ from the problem (ignoring the entries from the stream), then set $c \leftarrow c - (\min_{i^* \in [m]} c_{i^*})\mathbf{1}$ entrywise ;
2 $b \leftarrow \min\left(\max\left(b, -\|\mathbf{A}\|_\infty\right), \|\mathbf{A}\|_\infty\right)$;
3 $r(x, y) := x^\top |\mathbf{A}|(y^2) + 10\|\mathbf{A}\|_\infty \sum_{i \in [m]} x_i \log x_i$, $g(x, y) := (\mathbf{A}y + c, -\mathbf{A}^\top x + b)$;
4 $z_0 \leftarrow (\frac{1}{m}\mathbf{1}_m, \mathbf{0}_n)$, $s_0 \leftarrow (\mathbf{0}_m, \mathbf{0}_n)$;
5 **for** $0 \leq t < T$ **do**
6 $\quad \gamma^{\mathsf{x}} \leftarrow s_t^{\mathsf{x}} + \frac{1}{3}(\mathbf{A}y_t + c)$;
7 $\quad \gamma^{\mathsf{y}} \leftarrow s_t^{\mathsf{y}} + \frac{1}{3}\left(b - \mathbf{A}^\top x_t\right)$;
8 $\quad w_t := (x'_t, y'_t) \leftarrow \mathsf{AltMin}(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \mathbf{A}, K, x_t, y_t)$ with $K = O(\log \frac{\max(\|\mathbf{A}\|_\infty, \|b\|_1)}{\epsilon})$ giving $\frac{\epsilon}{2}$-approx. minimizer to $\left\langle \frac{1}{3}g(z_t) - \nabla r(z_t), w \right\rangle + r(w)$ in $\Delta^m \times [-1, 1]^n$;
9 $\quad \gamma^{\mathsf{x}} \leftarrow s_t^{\mathsf{x}} + \frac{1}{6}(\mathbf{A}y'_t + c)$;
10 $\quad \gamma^{\mathsf{y}} \leftarrow s_t^{\mathsf{y}} + \frac{1}{6}\left(b - \mathbf{A}^\top x'_t\right)$;
11 $\quad z_{t+1} := (x_{t+1}, y_{t+1}) \leftarrow \mathsf{AltMin}(s_t^{\mathsf{x}} + \gamma^{\mathsf{x}}, s_t^{\mathsf{y}} + \gamma^{\mathsf{y}}, \mathbf{A}, K, x_t, y_t)$ with $K = O(\log \frac{\max(\|\mathbf{A}\|_\infty, \|b\|_1)}{\epsilon})$ giving $\frac{\epsilon}{2}$-approx. minimizer to $\left\langle \frac{1}{3}g(w_t) - \nabla r(z_t), z \right\rangle + r(z)$ in $\Delta^m \times [-1, 1]^n$;
12 $\quad s_{t+1} \leftarrow (\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}})$;

Algorithm 14: $\mathsf{AltMin}(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \mathbf{A}, \delta, K, x_{\text{init}}, y_{\text{init}})$

1 **Input:** $\mathbf{A} \in \mathbb{R}^{m \times n}_{\geq 0}$, $\gamma^{\mathsf{x}} \in \mathbb{R}^m$, $\gamma^{\mathsf{y}} \in \mathbb{R}^n$, $K \in \mathbb{N}$;
2 **Output:** Approx. minimizer to $\langle (\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}), z \rangle + r(z)$ for $r(z)$ in Line 2, Algorithm 13 ;
3 $x^{(0)} \leftarrow x_{\text{init}}$, $y^{(0)} \leftarrow y_{\text{init}}$;
4 **for** $0 \leq k < K$ **do**
5 $\quad x^{(k+1)} \leftarrow \operatorname{argmin}_{x \in \Delta^m} \left\{ \langle \gamma^{\mathsf{x}}, x \rangle + r(x, y^{(k)}) \right\}$;
6 $\quad y^{(k+1)} \leftarrow \operatorname{argmin}_{y \in [-1,1]^n} \left\{ \langle \gamma^{\mathsf{y}}, y \rangle + r(x^{(k+1)}, y) \right\}$;
7 **Return:** $(x^{(K)}, y^{(K)})$;

The guarantees of Algorithms 13 and 14 are demonstrated by Appendix D.1, and summarized in Proposition 14. The main technical modification in it is a tighter characterization of the alternating minimization subroutine, based on bounding the initial error independently of the number of iterations (whereas Appendix D.1 assumed a worst-case bound on initial error).

Proposition 14. *By taking $T = O(\frac{\|\mathbf{A}\|_\infty \log m}{\epsilon})$ for a sufficiently large constant, Algorithm 13 results in iterates $\{(x'_t, y'_t)\}_{0 \leq t < T}$ so that $\bar{x} := \frac{1}{T}\sum_{0 \leq t < T} x'_t$ is an ϵ-approximate minimizer to* (5.2).

We defer a proof of Proposition 14 to Appendix D.1.7, as it largely stems from ideas used in Appendix D.1. We next turn to our main technical contribution, the development of semi-streaming algorithms for implementing Algorithms 13 and 14 (Lemma 70), the primary innovation of this section. Concretely, our implementation has the following key property: in every iteration all box variables are maintained explicitly, and all simplex variables are maintained implicitly as points proportional to $\exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c)$, for some vectors $v, u \in \mathbb{R}^n$ and scalar λ computed and stored in $O(n)$ space. We first make a crucial technical observation regarding computing gradients with a single pass.

Lemma 69. *There is a one pass semi-streaming algorithm that computes* $\|\exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c)\|_1$, $\mathbf{A}^\top \exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c)$, *and* $|\mathbf{A}|^\top \exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c)$ *for input* $v, u \in \mathbb{R}^n$ *and* $\lambda \in \mathbb{R}$ *with* $O(n)$ *space and* $\mathrm{nnz}(\mathbf{A})$ *work.*

Proof. First, to compute $\|\exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c)\|_1$, for each $i \in [m]$ the algorithm sequentially computes each $\exp(\langle \mathbf{A}_{i:}, v\rangle + \langle |\mathbf{A}_{i:}|, u\rangle + \lambda c_i)$ through the pass and sums these values. Second, note

$$\mathbf{A}^\top \exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c) = \sum_{i\in[m]} \mathbf{A}_{i:} \exp(\langle \mathbf{A}_{i:}, v\rangle + \langle |\mathbf{A}_{i:}|, u\rangle + \lambda c_i),$$
$$|\mathbf{A}|^\top \exp(\mathbf{A}v + |\mathbf{A}|u + \lambda c) = \sum_{i\in[m]} |\mathbf{A}_{i:}| \exp(\langle \mathbf{A}_{i:}, v\rangle + \langle |\mathbf{A}_{i:}|, u\rangle + \lambda c_i).$$

Throughout the pass, the algorithm adds $\exp(\langle \mathbf{A}_{i:}, v\rangle + \langle |\mathbf{A}_{i:}|, u\rangle + \lambda c_i)$ to the coordinates of the resulting vector in $\mathbf{A}$ and $|\mathbf{A}|$ it contributes to (specified by the row given in the stream). □

Using ideas from Lemma 69, we demonstrate that the entirety of Algorithm 13 and its subroutine Algorithm 14 can be implemented in low space.

Lemma 70. *Consider an instance of Algorithm 13 using Algorithm 14 to implement Lines 5 and 6. Throughout, the following invariants on the iterates hold:*

$$x_t \propto \exp(\mathbf{A}v_t + |\mathbf{A}|u_t + \lambda_t c),\ x'_t \propto \exp(\mathbf{A}v'_t + |\mathbf{A}|u'_t + \lambda'_t c) \text{ for some } v_t, v'_t, u_t, u'_t \in \mathbb{R}^n,\ \lambda_t, \lambda'_t \in \mathbb{R}.$$

Moreover, we also have

$$s^{\mathsf{x}}_t = \mathbf{A}v''_t + \lambda''_t c \text{ for some } v''_t \in \mathbb{R}^n,\ \lambda''_t \in \mathbb{R}.$$

Given their values in the previous iteration, in every iteration of Lines 6-12 *of Algorithm 13 we can compute such* $(v_t, v'_t, v''_t, u_t, u'_t, \lambda_t, \lambda'_t, \lambda''_t, y_t, y'_t, s^{\mathsf{y}}_t)$ *in* $O(n)$ *space. Each iteration takes* $O(K)$ *passes and* $O(\mathrm{nnz}(\mathbf{A}) \cdot K)$ *total work, where* K *is the input parameter to Algorithm 14.*

Proof. We proceed by induction on t; in the first iteration, we have $v_0 = u_0 = v''_0 = \mathbf{0}_n$, $\lambda_0 = \lambda''_0 = 0$.

Preserving the s_t *invariant.* Recall that $s_{t+1} = s_t + (\mathbf{A}v + \frac{1}{6}c, \frac{1}{6}(b - \mathbf{A}^\top x'_t))$ where $v := \frac{1}{6}y'_t$. By induction on w_t, we can explicitly compute v in $O(n)$ space, and Lemma 69 lets us explicitly compute

$\frac{1}{6}(b-\mathbf{A}^\top x'_t)$ in $O(n)$ space as well. Hence, we can inductively perform the following updates in $O(n)$ space, maintaining the invariant on s_{t+1}:

$$\lambda''_{t+1} \leftarrow \lambda''_t + \frac{1}{6},\ v''_{t+1} \leftarrow v''_t + v,\ s^{\mathsf{y}}_{t+1} \leftarrow s^{\mathsf{y}}_t + \frac{1}{6}(b - \mathbf{A}^\top x'_t).$$

Preserving the w_t invariant. Suppose inductively that $z_t = (x_t, y_t)$ where $x_t \propto \exp(\mathbf{A}v_t + |\mathbf{A}|u_t + \lambda_t c)$ for explicitly stored values v_t, u_t, λ_t, y_t; we will drop the index t for simplicity and refer to these as $\bar{v}$, $\bar{u}$, $\bar{\lambda}$, $\bar{y}$. Also, we refer to v''_t, λ''_t, and s^{y}_t as v_s, λ_s, and s^{y} for simplicity. Consider the procedure Algorithm 14 initalized with these values, and note that

$$\begin{aligned}
\gamma^{\mathsf{x}} &= \frac{1}{3}\left(\mathbf{A}\bar{y} + c\right) + \mathbf{A}v_s + \lambda_s c, \\
\gamma^{\mathsf{y}} &= \frac{1}{3}\left(b - \mathbf{A}^\top \frac{\exp(\mathbf{A}\bar{v} + |\mathbf{A}|\bar{u} + \bar{\lambda}c)}{\left\|\exp(\mathbf{A}\bar{v} + |\mathbf{A}|\bar{u} + \bar{\lambda}c)\right\|_1}\right) + s^{\mathsf{y}}.
\end{aligned}$$

Since $r(x,y) = \left\langle |\mathbf{A}|(y^2), x\right\rangle + 10\left\|\mathbf{A}\right\|_\infty \sum_{i\in[m]} x_i \log x_i$, we can compute that for each $0 \le k < K$,

$$\begin{aligned}
x^{(k+1)} &= \operatorname{argmin}_{x\in\Delta^m}\left\{\left\langle |\mathbf{A}|\left((y^{(k)})^2\right) + \gamma^{\mathsf{x}}, x\right\rangle + 10\left\|\mathbf{A}\right\|_\infty \sum_{i\in[m]} x_i \log x_i\right\} \\
&\propto \exp\left(-\frac{1}{10\left\|\mathbf{A}\right\|_\infty}\left(\mathbf{A}\left(\frac{1}{3}\bar{y} + v_s\right) + |\mathbf{A}|(y^{(k)})^2 + \left(\frac{1}{3} + \lambda_s\right)c\right)\right), \\
y^{(k+1)} &= \operatorname{argmin}_{y\in[-1,1]^n}\left\{\langle\gamma^{\mathsf{y}}, y\rangle + \left\langle |\mathbf{A}|^\top x^{(k+1)}, y^2\right\rangle\right\} \\
&= \text{median}\left(-1, 1, -\frac{\gamma^{\mathsf{y}}}{2|\mathbf{A}|^\top x^{(k+1)}}\right) \text{ entrywise.}
\end{aligned} \tag{5.4}$$

In the last line, the median operation truncates the vector $-\frac{\gamma^{\mathsf{y}}}{2\mathbf{A}^\top x^{(k+1)}}$ coordinatewise on the box $[-1,1]^n$. Now, suppose at the start of iteration k of Algorithm 14, we have the invariant

$$x^{(k)} \propto \exp\left(\mathbf{A}v^{(k)} + |\mathbf{A}|u^{(k)} + \lambda^{(k)}c\right), \tag{5.5}$$

and we have explicitly stored the tuple $(v^{(k)}, u^{(k)}, \lambda^{(k)}, y^{(k)})$; in the first iteration, we can clearly choose $(v^{(0)}, u^{(0)}, \lambda^{(0)}, y^{(0)}) = (\bar{v}, \bar{u}, \bar{\lambda}, \bar{y})$. By the derivation (5.4), we can update

$$\begin{aligned}
v^{(k+1)} &\leftarrow -\frac{1}{10\left\|\mathbf{A}\right\|_\infty}\left(\frac{1}{3}\bar{y} + v_s\right), \\
u^{(k+1)} &\leftarrow -\frac{1}{10\left\|\mathbf{A}\right\|_\infty}\left((y^{(k)})^2\right), \\
\lambda^{(k+1)} &\leftarrow -\frac{1}{10\left\|\mathbf{A}\right\|_\infty}\left(\frac{1}{3} + \lambda_s\right),
\end{aligned}$$

preserving representation (5.5) in the next iteration. Moreover, note that γ^{y} can be explicitly stored

(it can be computed in one pass using Lemma 69), and by applying Lemma 69 and the form (5.5), we can compute the vector $|\mathbf{A}|^\top x^{(k+1)}$ explicitly in one pass over the data and $O(m)$ work. This allows us to explicitly store $y^{(k+1)}$ as well. The final point w_t is one of the iterates $(x^{(K)}, y^{(K)})$, proving the desired invariant. Finally, we remark in order to perform these computations we only need to store the tuple $(v^{(k)}, u^{(k)}, \lambda^{(k)}, y^{(k)})$ from the prior iteration, in $O(n)$ memory.

Preserving the z_{t+1} invariant. The argument for preserving the invariant on z_{t+1} is exactly analogous to the above argument regarding w_t; the only modification is that the input vector to Algorithm 14 is

$$\gamma^{\mathsf{x}} = \frac{1}{6}\left(\mathbf{A}y'_t + c\right) + \mathbf{A}v_s + \lambda_s c,$$
$$\gamma^{\mathsf{y}} = \frac{1}{6}\left(b - \mathbf{A}^\top x'_t\right) + s^{\mathsf{y}}.$$

However, the previous argument shows that the vector y'_t can be explicitly computed, and the vector x'_t satisfies the invariant in the lemma statement. The same inductive argument shows that we can represent every intermediate iterate in the computation of z_{t+1} in the desired form.

Numerical stability. We make a brief comment regarding numerical stability in the semi-streaming model, which may occur due to exponentiating vectors with a large range $\omega(\log m)$ (in defining simplex variables). It was shown in Appendix D.1.5 that Algorithm 13 is stable to increasing the value of any coordinate of a simplex variable which is m^{10} multiplicatively smaller than the largest to reach this threshold, and renormalizing (i.e. this implicit "padding" operation only affects the resulting minimizer by an inverse-polynomial amount, which we can assume ϵ is larger than since otherwise the number of passes is super-polynomial). In computations of Lemma 69, we can first store the largest coordinate of $\mathbf{A}v + |\mathbf{A}|u + \lambda c$ in one pass. Then, for every coordinate more than $10 \log m$ smaller than the largest coordinate, we will instead treat it as if its value was $10 \log m$ smaller than the maximum in all computations, requiring one extra pass over the data. □

We explicitly state a complete low-space implementation of Algorithms 13 and 14 under our semi-streaming implementation here for completeness as Algorithm 15.

Corollary 15. *Algorithm 15 is an implementation of Algorithms 13 and 14 with parameters given by Proposition 14 in $O(n)$ space and $O(T \cdot \log \frac{\max(\|\mathbf{A}\|_\infty, \|b\|_1)}{\epsilon})$ passes.*

Proof. It is immediate from Lemma 70 that each loop of Lines 6-10 and Lines 14-18 in Algorithm 15 is an implementation of Algorithm 14 for computing each iterate w_t and z_{t+1} in iteration t of Algorithm 13, in $O(n)$ space. The pass complexity follows since passes are only used $O(1)$ times in each run of Lines 6-11 and 15-20 of Algorithm 15. □

Finally, we conclude with our proof of Theorem 30.

Algorithm 15: LowSpaceACDualEx($\mathbf{A}, b, c, \epsilon$)

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}_{\geq 0}$, $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, $0 \leq \epsilon \leq \|\mathbf{A}\|_\infty$;

Output: $\{v'_t\}_{0 \leq t < T} \subset \mathbb{R}^n$, $\{u'_t\}_{0 \leq t < T} \subset \mathbb{R}^n$, $\{\lambda'_t\}_{0 \leq t < T} \subset \mathbb{R}$, $\{y'_t\}_{0 \leq t < T} \subset \mathbb{R}^n$ so that for

$$\bar{y} := \frac{1}{T} \sum_{0 \leq t < T} y'_t, \; \bar{x} := \frac{1}{T} \sum_{0 \leq t < T} \frac{\exp(\mathbf{A} v'_t + |\mathbf{A}| u'_t + \lambda'_t c)}{\|\exp(\mathbf{A} v'_t + |\mathbf{A}| u'_t + \lambda'_t c)\|_1},$$

the pair $(\bar{x}, \bar{y})$ is an ϵ-approximate saddle point to (5.1);

$T \leftarrow O(\frac{\|\mathbf{A}\|_\infty \log m}{\epsilon})$, $K \leftarrow O(\log \frac{n \|\mathbf{A}\|_\infty}{\epsilon})$;

$t \leftarrow 0$, $\lambda_0 \leftarrow 0$, $v_0 \leftarrow \mathbf{0}_n$, $u_0 \leftarrow \mathbf{0}_n$, $y_0 \leftarrow \mathbf{0}_n$, $\lambda''_0 \leftarrow 0$, $v''_0 \leftarrow \mathbf{0}_n$, $s^{\mathsf{y}}_0 \leftarrow \mathbf{0}_n$;

while $t < T$ **do**

  $(v^{(0)}, u^{(0)}, \lambda^{(0)}, y^{(0)}) \leftarrow (v_t, u_t, \lambda_t, y_t)$;

  $\gamma^{\mathsf{y}} \leftarrow \frac{1}{3}(b - \mathbf{A}^\top \frac{\exp(\mathbf{A} v_t + |\mathbf{A}| u_t + \lambda_t c)}{\|\exp(\mathbf{A} v_t + |\mathbf{A}| u_t + \lambda_t c)\|_1}) + s^{\mathsf{y}}_t$, computed using Lemma 69;

  for $0 \leq k < K$ **do**

    $v^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}(\frac{1}{3} y_t + v''_t)$;

    $u^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}((y^{(k)})^2)$;

    $\lambda^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}(\frac{1}{3} + \lambda''_t)$;

    $d^{(k+1)} \leftarrow 2|\mathbf{A}|^\top \frac{\exp(\mathbf{A} v^{(k+1)} + |\mathbf{A}| u^{(k+1)} + \lambda^{(k+1)} c)}{\left\|\exp(\mathbf{A} v^{(k+1)} + |\mathbf{A}| u^{(k+1)} + \lambda^{(k+1)} c)\right\|_1}$, computed using Lemma 69;

    $y^{(k+1)} \leftarrow \mathrm{med}(-1, 1, -\frac{\gamma^{\mathsf{y}}}{d^{(k+1)}})$ entrywise;

  $(v'_t, u'_t, \lambda'_t, y'_t) \leftarrow (v^{(K)}, u^{(K)}, \lambda^{(K)}, y^{(K)})$;

  $(v^{(0)}, u^{(0)}, \lambda^{(0)}, y^{(0)}) \leftarrow (v_t, u_t, \lambda_t, y_t)$;

  $\gamma^{\mathsf{y}} \leftarrow \frac{1}{6}(b - \mathbf{A}^\top \frac{\exp(\mathbf{A} v'_t + |\mathbf{A}| u'_t + \lambda'_t c)}{\|\exp(\mathbf{A} v'_t + |\mathbf{A}| u'_t + \lambda'_t c)\|_1}) + s^{\mathsf{y}}_t$, computed using Lemma 69;

  for $0 \leq k < K$ **do**

    $v^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}(\frac{1}{6} y'_t + v''_t)$;

    $u^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}((y^{(k)})^2)$;

    $\lambda^{(k+1)} \leftarrow -\frac{1}{10\|\mathbf{A}\|_\infty}(\frac{1}{6} + \lambda''_t)$;

    $d^{(k+1)} \leftarrow 2|\mathbf{A}|^\top \frac{\exp(\mathbf{A} v^{(k+1)} + |\mathbf{A}| u^{(k+1)} + \lambda^{(k+1)} c)}{\left\|\exp(\mathbf{A} v^{(k+1)} + |\mathbf{A}| u^{(k+1)} + \lambda^{(k+1)} c)\right\|_1}$, computed using Lemma 69;

    $y^{(k+1)} \leftarrow \mathrm{med}(-1, 1, -\frac{\gamma^{\mathsf{y}}}{d^{(k+1)}})$ entrywise ;

  $(v_{t+1}, u_{t+1}, \lambda_{t+1}, y_{t+1}) \leftarrow (v^{(K)}, u^{(K)}, \lambda^{(K)}, y^{(K)})$;

  $(v''_{t+1}, \lambda''_{t+1}, s^{\mathsf{y}}_{t+1}) \leftarrow (v''_t, \lambda''_t, s^{\mathsf{y}}_t) + (\frac{1}{6} y'_t, \frac{1}{6}, \mathbf{0}_n)$;

  $t \leftarrow t + 1$;

Proof of Theorem 30. We use the parameter settings in Proposition 14, which implies that it suffices to compute the value of (5.2) with $\bar{x} := \frac{1}{T} \sum_{0 \leq t < T} x'_t$. By linearity, we have

$$c^\top \bar{x} = \frac{1}{T} \sum_{0 \leq t < T} \langle c, x'_t \rangle = \frac{1}{T} \sum_{0 \leq t < T} \sum_{i \in [m]} c_i \exp\left(\langle \mathbf{A}_{i:}, v'_t \rangle + \langle |\mathbf{A}_{i:}|, u'_t \rangle + \lambda'_t c_i\right).$$

Since the summands in the above display (for each $0 \leq t < T$) separate componentwise in the

stream, we can compute each $\sum_{i\in[m]} c_i \exp(\langle \mathbf{A}_{i:}, v'_t\rangle + \langle |\mathbf{A}_{i:}|, u'_t\rangle + \lambda'_t c_i)$ in one pass per iteration and store their average. Moreover, we can similarly compute the vector $\mathbf{A}^\top \bar{x}$ using Lemma 69 in one pass per iteration and store it explicitly in $O(n)$ space, at which point we can compute $\left\|\mathbf{A}^\top \bar{x} - b\right\|_1$. Combining these two parts, we have the desired approximation to the value.

Finally, to demonstrate the guarantee on streaming $\bar{x}$, the approximate minimizer has the form

$$\bar{x} = \frac{1}{T} \sum_{0\le t<T} \frac{\exp\left(\mathbf{A}v'_t + |\mathbf{A}|u'_t + \lambda'_t c\right)}{\left\|\exp\left(\mathbf{A}v'_t + |\mathbf{A}|u'_t + \lambda'_t c\right)\right\|_1}.$$

We can thus take one additional pass per iteration to compute the value of

$$\left[\frac{1}{T} \frac{\exp\left(\mathbf{A}v'_t + |\mathbf{A}|u'_t + \lambda'_t c\right)}{\left\|\exp\left(\mathbf{A}v'_t + |\mathbf{A}|u'_t + \lambda'_t c\right)\right\|_1}\right]_i \text{ for all } i \in [m],$$

and produce a stream of these values without affecting the overall pass complexity. □

We note that $\max(\|\mathbf{A}\|_\infty, \|b\|_1)$ is never larger than $n\|\mathbf{A}\|_\infty$ by Line 2 of Algorithm 13. Finally, on a query $S \subseteq E$ with $|S| = q$, using $O(n+q)$ space (and no more passes), for $(\bar{x}, \bar{y})$ an ϵ-approximate saddle point to (5.1) computed by our algorithm, we can output the set of values $(\{\bar{x}_i\}_{i\in S}, y)$. This is accomplished by calling Lemma 69 and explicitly storing values of x'_t on coordinates in S each iteration, which is implementable in $O(q)$ additional space. While we do not use this fact for finding an approximate MCM solution or for our applications, this gives the memory requirement for querying entries of an approximate solution in the most general setting for box-simplex games.

5.4 Approximate maximum cardinality matching

In this section, we give specific treatment to the problem of approximate MCM and prove our main result Theorem 26. We prove Theorem 26 by assembling a variety of tools, centered around casting MCM as an instance of (5.1) and using several helper procedures to complete the result.

1. In Section 5.4.1, we give the specific box-simplex problem formulation which serves as the optimization workhorse of this section, and prove that an appropriate approximate solution to this problem results in an approximate MCM.

2. In Section 5.4.2, we give pre- and post-processing tools for manipulating the box-simplex problem and its output. Specifically, we give a vertex-size reduction enabling a tighter runtime analysis, and a cycle cancelling procedure for sparsifying the support of the approximate MCM.

3. Finally, we put these pieces together to prove Theorem 26 in Section 5.4.3.

5.4.1 Reducing MCM to a box-simplex problem

Throughout this section, we consider the problem of finding a ϵ-approximate MCM of a bipartite graph $G = (V, E)$ with $|V| = n$, $|E| = m$, with unsigned incidence matrix $\mathbf{B} \in \{0,1\}^{E \times V}$, and maximum matching size M^*. Any maximum (possibly fractional) matching solves the problem

$$\max_{M^* > 0} M^* \text{ such that } \exists x \in \Delta^E \text{ with } \mathbf{B}^\top (M^* x) \leq \mathbf{1}_V \text{ entrywise.}$$

Here, the variable $M^* x$ takes on the role of the matching; observe that since $x \in \Delta^E$, the ℓ_1 norm of $M^* x$ is precisely M^*. Before giving the box-simplex objective which we will approximate with the value algorithm of Section 5.3, we recall that we can obtain a 2-approximation to M^* in one pass.

Lemma 71. *The greedy algorithm can be implemented in one semi-streaming pass over a graph using $O(n)$ space and $O(m)$ work to return a matching of size M with $M \leq M^* \leq 2M$.*

The proof is standard and deferred to Appendix D.2. As the first step of our approximate MCM algorithm, we obtain M so that $M \leq M^* \leq 2M$ using the above greedy algorithm. We then use this estimate to construct a problem of the form (5.2), whose approximate solution also yields an approximate MCM, which we describe now.

From the unweighted graph $G = (V, E)$, we construct a modified graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ with two extra vertices and one extra zero-weight edge between them, and refer to its weighted adjacency matrix as $\widetilde{\mathbf{B}}$, with an all-zero row for the extra edge. We define $d_{\text{MCM}} \in \{0,1\}^{\widetilde{V}}$ to be the vector which is 1 in every coordinate of $V \in \widetilde{V}$, and 0 in the two extra vertices. Finally, we require one additional piece of notation: for a graph $(\widetilde{V}, \widetilde{E})$, some *demands* $d \in \mathbb{R}^{\widetilde{V}}_{\geq 0}$, and some flow $\tilde{x} \in \mathbb{R}^{\widetilde{E}}_{\geq 0}$, we define the *overflow* by

$$\text{Overflow}_d(\tilde{x}) := \sum_{j \in \widetilde{V}} \left[\left(\widetilde{\mathbf{B}}^\top \tilde{x} - d \right)_+ \right]_j.$$

Combinatorially, this can be interpreted as the total amount the flow $\tilde{x}$ violates d. We show that to obtain an approximate matching in G, it suffices to find an ϵM-approximate minimizer to the following problem (where d_{MCM} as defined earlier is the demands for the MCM problem):

$$\min_{x \in \Delta^{\widetilde{E}}} -2M \|x_E\|_1 + \text{Overflow}_{d_{\text{MCM}}}(2Mx). \tag{5.6}$$

In particular, note that we can represent any flow on the original edges E with ℓ_1 norm *at most* $2M$ (in particular, the maximum cardinality matching) as the restriction of a flow in $\mathbb{R}^{\widetilde{E}}_{\geq 0}$ with ℓ_1 norm exactly $2M$. Hence, we can interpret (5.6) combinatorially as attempting to put as much flow on the edges of the original graph E as possible, while penalizing the overflow.

We begin by demonstrating that to find approximate minimizers to (5.6), it suffices to find an

approximate minimizer to the following ℓ_1 regression problem:

$$\min_{x \in \Delta^{\widetilde{E}}} \left\| \mathbf{A}^\top x - b \right\|_1, \text{ where } \mathbf{A} := M\widetilde{\mathbf{B}},\ b := \frac{1}{2} d_{\mathrm{MCM}}. \tag{5.7}$$

Lemma 72. *Any Δ-additively approximate minimizer to problem (5.6) is a Δ-additively approximate minimizer to problem (5.7), for all $\Delta > 0$, and vice versa.*

Proof. The property of being a Δ-approximate minimizer is preserved if the entire problem is shifted by a constant scalar, so it suffices to show (5.6) and (5.7) differ by a constant scalar. Observe that for any scalar v, $\max(v, 0) = \frac{1}{2}(v + |v|)$. Thus, we can write

$$\begin{aligned}
\mathrm{Overflow}_{d_{\mathrm{MCM}}}(2Mx) &= \sum_{j \in V} \max\left(2M\widetilde{\mathbf{B}}^\top x - 1, 0\right) \\
&= \frac{1}{2}\left\| 2M\widetilde{\mathbf{B}}^\top x - d_{\mathrm{MCM}} \right\|_1 + \frac{1}{2}\sum_{j \in V}\left(2M\widetilde{\mathbf{B}}^\top x - 1\right) \\
&= \left\| \mathbf{A}^\top x - b \right\|_1 + \langle M\mathbf{B} d_{\mathrm{MCM}}, x \rangle - \frac{1}{2}|V|.
\end{aligned}$$

Comparing with (5.6) yields the conclusion, since $M\mathbf{B}d_{\mathrm{MCM}}$ is $2M$ times the indicator on E. □

Finally, to conclude this section, we give a simple procedure for rounding an almost-feasible flow to a feasible matching. This procedure comes with a guarantee which converts an overflow bound on the original flow to a bound on the difference in quality of the resulting rounded matching. We remark that a similar algorithm was developed in [25], and reproduced in Appendix D.1; however, we give a somewhat tighter analysis which is suitable for our purposes here.

Algorithm 16: RemoveOverflow(x, G, d)

Input: Bipartite graph $G = (V, E)$ with incidence matrix $\mathbf{B}$, $x \in \mathbb{R}^E_{\geq 0}$, demands $d \in \mathbb{R}^V_{\geq 0}$;
Output: $\tilde{x} \in \mathbb{R}^E_{\geq 0}$ with $\tilde{x} \leq x$ entrywise, $\mathbf{B}^\top \tilde{x} \leq d$, and $\|\tilde{x}\|_1 \geq \|x\|_1 - \left\| \left(\mathbf{B}^\top x - d\right)_+ \right\|_1$;
$d^x \leftarrow \mathbf{B}^\top x$;
$f \leftarrow (d^x - d)_+$;
$\tilde{x}_e \leftarrow x_e \left(1 - \max\left(\frac{f_a}{d^x_a}, \frac{f_b}{d^x_b}\right)\right)$ for all $e = (a, b) \in E$ with $x_e > 0$;
Return: $\tilde{x}$;

Lemma 73 (Overflow removal)**.** RemoveOverflow *is correct, i.e. on input $x \in \mathbb{R}^E_{\geq 0}$, it outputs $\tilde{x} \in \mathbb{R}^E_{\geq 0}$ with $\tilde{x} \leq x$, $\mathbf{B}^\top \tilde{x} \leq d$, and $\|\tilde{x}\|_1 \geq \|x\|_1 - \mathrm{Overflow}_d(x)$.*

Proof. First, $x_e > 0$ for $e = (a, b)$ implies $d^x_a > 0$ and $d^x_b > 0$ and therefore $\tilde{x}$ is well-defined. As $d, f \geq 0$ and $e \leq d_x$ entrywise we also see that $\tilde{x} \geq 0$ and $\tilde{x} \leq x$, giving the first guarantee.

Next, note that for all $a \in V$ if $[\mathbf{B}^\top x]_a \le d$ then $[\mathbf{B}^\top \tilde{x}]_a \le d_a$ as $\tilde{x} \le x$. On the other hand, if $[\mathbf{B}^\top \tilde{x}]_a > d_a$ then $f_a = d_a^x - d_a$ and

$$[\mathbf{B}^\top \tilde{x}]_a = \sum_{(a,b)\in E} \tilde{x} \le \sum_{(a,b)\in E} x_e \left(1 - \frac{f_a}{d_a^x}\right) = d_a^x \left(1 - \frac{d_a^x - d_a}{d_a}\right) = d_a.$$

Consequently, in either case $[\mathbf{B}^\top \tilde{x}]_a \le d_a$ so $\mathbf{B}^\top \tilde{x} \le d$. The last claim of the lemma follows from

$$\begin{aligned}\|\tilde{x}\|_1 &= \sum_{e\in E} x_e - \sum_{e=(a,b)\in E | x_e \neq 0} x_e \max\left\{\frac{f_a}{d_a^x}, \frac{f_b}{d_b^x}\right\} \ge \|x\|_1 - \sum_{e=(a,b)\in E | x_e\neq 0} x_e \left(\frac{f_a}{d_a^x} + \frac{f_b}{d_b^x}\right) \\ &= \|x\|_1 - \sum_{a\in V} \sum_{e=(a,b)\in E | x_e \neq 0} x_e \frac{e_a}{d_a^x} = \|x\|_1 - \sum_{a\in V} e_a = \|x\|_1 - \left\|\left(\mathbf{B}^\top x - d\right)_+\right\|_1.\end{aligned}$$

□

As a consequence of Lemma 73 with $d = d_{\mathrm{MCM}}$, we have an (algorithmic) proof that any $\tilde{x}$ satisfying

$$\|\tilde{x}\|_1 - \mathrm{Overflow}_d(\tilde{x}) \ge (1-\epsilon)M^* \tag{5.8}$$

can be rounded to a feasible matching on the same support with ℓ_1 norm at least $(1-\epsilon)M^*$.

5.4.2 Additional tools

Before proving Theorem 26, we give a few helper tools building upon prior works in the literature, to prove Theorem 26, as Propositions 15 and 16. First, when $n \gg M^*$, we note that we can reduce the number of vertices to $O(M^* \log(\epsilon^{-1}))$ while only slightly decreasing the MCM size; a similar result is given in [12]. We state the reduction here, and defer its proof to Appendix D.2 for completeness.

Proposition 15 (Vertex size reduction)**.** *There is a procedure* VertexReduction *(Algorithm 73) which takes as input unweighted bipartite $G = (V, E)$ with MCM size M^* and with $O(\log \epsilon^{-1})$ passes, $O(M^* \log \epsilon^{-1})$ space, and $O(m \log \epsilon^{-1})$ work outputs a subset $V' \subseteq V$ of size $O(M^* \log \epsilon^{-1})$ such that the induced subgraph $G[V']$ has a MCM of size at least $(1-\epsilon)M^*$.*

We also require a procedure for sparsifying supports based on cycle cancelling. Our cycle cancelling procedure takes a (possibly infeasible) flow given as an insertion-only stream, and produces a flow of at least the same value and with no additional overflow, with support of size at most n. The procedure generalizes to weighted bipartite matching problems as well; we remark that similar procedures have appeared in prior work (see e.g. [306]). We provide a detailed implementation of a data structure to prove Proposition 16 in Appendix D.3.

Proposition 16 (Cycle cancelling)**.** *Consider a (possibly weighted) matching problem on a bipartite graph $G = (V, E, w)$ (for MCM, we let $w = \mathbf{1}$). There is an algorithm that has the following property:*

given a stream of length L, consisting of edge-flow tuples (e, f_e) where $e \in E$ and $f_e \in \mathbb{R}_{\geq 0}$, define $x \in \mathbb{R}^E_{\geq 0}$ to be the sum of all $f_e \mathbf{1}_e$ in the stream where $\mathbf{1}_e$ is the 1-sparse indicator of edge e. Then the algorithm runs in $O(n)$ space and $O(L \log n)$ time, and outputs a flow $\tilde{x}$ supported on $O(n)$ edges forming a forest, so that $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$, and $\mathbf{B}^\top x = \mathbf{B}^\top \tilde{x}$.

5.4.3 Approximate MCM in fewer passes and optimal space

We finally give a proof of Theorem 26 by applying the tools we have built.

Theorem 26 (Approximate MCM)**.** *There is a deterministic semi-streaming algorithm which given any bipartite $G = (V, E)$ with $|V| = n, |E| = m$, finds a ϵ-multiplicatively approximate MCM in $O\big(\log n \cdot \log(\epsilon^{-1}) \cdot \epsilon^{-1}\big)$ passes, $O(n)$ space, and $O(m \log^2 n \cdot \epsilon^{-1})$ total work.*

Proof. We first define the steps of the algorithm, where we adjust the error parameter ϵ by an appropriate constant in each of the following subroutines.

1. Use Lemma 71 to obtain an estimate $M \leq M^* \leq 2M$.
2. If $M \log \frac{1}{\epsilon} \leq n$, use Proposition 15 to reduce to $O(M \log \frac{1}{\epsilon})$ vertices; overload V, E to be the vertex and edge sets on the resulting induced subgraph.
3. Solve (5.7) to ϵM additive accuracy using Algorithm 13, with approximate minimizer x. Set $\tilde{x} = 2Mx_E$.
4. Apply Proposition 16 to $\tilde{x}$ in streaming fashion to obtain a flow $\hat{x}$.
5. Greedily find an exact MCM on the support of $\hat{x}$.

We next demonstrate correctness. Proposition 14 shows that Algorithm 13 is correct for implementing Step 3. By Proposition 15, the result of Step 3 of the above algorithm satisfies (5.8) (by appropriately adjusting constants), since $\epsilon M^* \geq \epsilon M$ by Step 1, and the optimal value of (5.6) is at most M^* by choosing $2Mx$ to be the restriction of the MCM to E, with all overflow placed on the extra edge. Proposition 16 then returns a sparse $O(\epsilon)$-approximate MCM supported on a forest, also satisfying (5.8). Lemma 73 then implies that this support contains an $O(\epsilon)$-approximate MCM. It is well-known that to compute a maximum cardinality matching on a tree, the greedy algorithm of repeatedly taking a leaf edge and removing both endpoints from the graph suffices [239]. Applying this to each tree in the forest yields correctness.

Finally, we discuss pass, space, and work complexities. Since Proposition 15 gives a vertex size bound of at most $M \log \frac{1}{\epsilon}$, using Algorithm 13 and 14 in Step 3 of the algorithm implies by Corollary 15 that the pass complexity is $\log n \cdot \log(\epsilon^{-1}) \cdot \epsilon^{-1}$, since the additive accuracy level is $O(\epsilon M^*)$ and $\|\mathbf{A}\|_\infty = \|b\|_1 = O(M^*)$. Next, by applying Proposition 16 to the input stream x which is the average of the iterates of Algorithm 13, we reduce the support of x to be on $O(n)$ edges

supported on a forest, while maintaining that (5.8) is satisfied. We can give x in a stream of size $O(m \log n \cdot \epsilon^{-1})$ which induces the average iterate, per Theorem 30. This takes $O(m \log^2 n \cdot \epsilon^{-1})$ work by Proposition 15, and does not dominate the pass complexity. Finally, we explicitly store the support of the forest, so it is clear that Step 5 is implementable in $O(n)$ space and work. □

5.5 Further matching applications

Here we give a few applications using the semi-streaming matching algorithm proposed in the paper. Section 5.5.1 shows how to apply the semi-streaming matching algorithm to give exact MCM. Sections 5.5.2, 5.5.3 and 5.5.4 extend our semi-streaming framework and algorithms to the model where the cost vector c is non-uniform. We first state a general result on solving box-simplex games induced by weighted matching problems in Section 5.5.2. Then in Section 5.5.3 and 5.5.4, we show how to adapt the model to specific applications. Specifically, we develop space-effcient solvers for optimal transportation and MWM respectively, by reductions to our more general solver.

5.5.1 Exact maximum cardinality matching

Here we show how to leverage Theorem 26 to obtain the following result on computing MCMs.

Theorem 66 (Exact MCM)**.** *There is a randomized semi-streaming algorithm which given any bipartite $G = (V, E)$ with $|V| = n$, finds an exact MCM with high probability in $O(n^{\frac{3}{4}+o(1)})$ passes.*

We remark that the number of passes in Theorem 27 can be reduced to $\widetilde{O}((M^*)^{\frac{3}{4}+o(1)})$ for any MCM problem with MCM size $M^* > 0$, using the vertex size reduction of Proposition 15 in Section 5.4.2. To prove Theorem 27, we use the following result from [290] regarding reachability in directed graphs. This result was originally proved for parallel (PRAM) algorithms, but its extension to semi-streaming algorithms is straightforward and has been observed previously, e.g. in [368].

Proposition 17 ([290])**.** *There is a semi-streaming algorithm that given* directed *$G = (V, E)$ with $|V| = n$ and vertices s, t, with high probability finds a path from s to t (assuming one exists) in $O(n^{\frac{1}{2}+o(1)})$ passes.*

We remark that as stated, [290] only solves the decision problem of whether s can reach t, while in Proposition 17 we require the path. However, here we provide a brief sketch of how it is straightforward to modify [290] to compute a s-t path and thereby prove Proposition 17.

The algorithm of [290] computes a *hopset* H of $\widetilde{O}(n)$ edges, i.e. a graph H with $\widetilde{O}(n)$ edges with the property that any vertex u can reach another vertex v in $G \cup H$ if and only u can reach v in G also. Further, this hopset has the property that with high probability there exists a path of length $O(n^{\frac{1}{2}+o(1)})$ from s to t in $G \cup H$. The hopset H is computed in $O(\log n)$ iterations as $H_1 \cup H_2 \cup \ldots H_{O(\log n)}$, where each H_i is computed from $O(n^{\frac{1}{2}+o(1)})$-depth breath-first searches in

vertex-induced subgraphs of $G_i = G \cup H_1 \cup H_2 \cup \cdots \cup H_{i-1}$. This computation requires $\widetilde{O}(m)$ work and $O(n^{\frac{1}{2}+o(1)})$ depth in the PRAM model (as was the motivation in [290]). Similarly, it can also be implemented in $O(n^{\frac{1}{2}+o(1)})$ passes in the semi-streaming model by performing the operations of each level of parallel depth within $O(1)$ passes. Furthermore, a directed edge (u, v) is added to H_i if and only if u is the root of one of the BFS arborescences used to form H_i, and can reach v in the same BFS computation.

To recover a path from s to t, we run the algorithm of [290] and store the arborescences used to form H. We then perform one last $O(n^{\frac{1}{2}+o(1)})$-depth breadth-first search from s in $G \cup H$: if a path P from s to t in $G \cup H$ is found, we mark the edges of P and store them in the memory. We can then replace the edges of P that belong to H to the edges of G stored as parts of arborescences used to form H. For i from $O(\log n)$ down to 1, we replace all edges in this s-t path belonging to H_i with the corresponding arborescence paths belonging to G_i. In the end we obtain a (not necessarily simple) path from s to t in G, which we may make simple by removing duplicates (notice that technically for this step, we only need to run a BFS algorithm offline, i.e., with no more passes on the stream, from s to t among the stored edges of G that form $H \cup P$).

Proof of Theorem 27. We set $\epsilon := n^{-\frac{3}{4}}$ and run our algorithm in Theorem 26 on G to obtain a matching F with $|F| \geq (1-\epsilon) \cdot M^* \geq M^* - n^{\frac{1}{4}}$ (as $M^* \leq n$). This requires $O(n^{\frac{3}{4}+o(1)})$ passes.

We then augment this approximate MCM F to an exact MCM by repeatedly finding augmenting paths for it. We form the (directed) residual graph for F as follows: we take vertices $\{V, s, t\}$ and add the edges (s, v) for all $v \in L$, (v, t) for all $v \in R$, and for any $e = (i, j) \in E(G)$ for $i \in L, j \in R$, we add (i, j) if $e \notin F$ and (j, i) if $e \in F$. As long as F is suboptimal, we can increase its size by 1 by finding a path from s to t in this residual graph and augmenting the flow with this path's edges: such a path can be found in $O(n^{\frac{1}{2}+o(1)})$ passes by applying Proposition 17. Since $|F| \geq M^* - n^{\frac{1}{4}}$ initially, it can be augmented at most $n^{\frac{1}{4}}$ times, and the result follows. □

Remark 7. *The exact MCM in Theorem 27 is a direct byproduct of the simultaneous space and pass-efficiency of our approximate algorithm and cannot be achieved directly by the prior state-of-the-art. While the space of algorithm of [42] is $O(n)$ like ours, their algorithm requires $O(\epsilon^{-2})$ passes which combined with the augmentation approach and $O(n^{\frac{1}{2}+o(1)})$-pass algorithm of [290] for each augmentation, leads to $\widetilde{O}(n)$ passes for exact MCM. On the other hand, while the pass complexity of the algorithm of [13] is sufficient on the surface for this augmentation approach, its space-complexity exceeds the $\widetilde{O}(n)$ requirement of semi-streaming algorithms when $\epsilon \ll (\text{poly}\log(n))^{-1}$. More precisely, by setting ϵ small enough in the algorithm of [13] to obtain an $o(n)$ pass algorithm using the above approach, the space requirement would become $\Omega(n^2)$, which matches the trivial bound obtained by storing the input in just one pass and solving the problem optimally offline.)*

5.5.2 Weighted bipartite matching under an ℓ_1 constraint

In this section $G = (V, E)$ is a bipartite graph where $V = L \cup R$ with unweighted incidence matrix $\mathbf{B} \in \{0,1\}^{E \times V}$. We consider a weighted matching problem parameterized by a (possibly non-uniform) demand vector $d \in [0,1]^V$, and weights $w \in \mathbb{R}^E_{\geq 0}$, formally defined as follows:

$$M^* := \max_{x \geq 0} w^\top x \text{ subject to } \mathbf{B}^\top x \leq d. \tag{5.9}$$

We also assume we know the value S of the ℓ_1 norm of an optimal matching which is feasible for the demands d, yielding the maximum matching weight M^*; in all our relevant applications, we will have $S \geq 1$. The assumption that we exactly know S may seem restrictive; in all our applications (cf. Section 5.5.3 and 5.5.4), it will suffice to know an upper bound and pad the graph with a dummy edge appropriately. We will refer to the optimal matching as Sx^*, for some $x^* \in \Delta^E$.

Next, we state the ℓ_1 regression problem we solve to compute an approximate solution to (5.9). Define the extended graph with vertex and edge sets $\widetilde{V}$, $\widetilde{E}$ and incidence matrix $\widetilde{\mathbf{B}}$, as in the reduction of Section 5.4.1 (as a reminder, it simply adds an extra isolated edge with no constraints on endpoints). The overflow formulation we will solve to obtain an approximate minimizer to (5.9) is:

$$\min_{x \in \Delta^{\widetilde{E}}} -\langle w_E, Sx \rangle + \|w\|_\infty \operatorname{Overflow}_d(Sx). \tag{5.10}$$

By the arguments of Lemma 72, (5.10) is equivalent to the following ℓ_1 regression problem up to a constant scalar shift:

$$\begin{gathered} \min_{x \in \Delta^{\widetilde{E}}} -c^\top x + \left\| \mathbf{A}^\top x - b \right\|_1, \\ \text{where } \mathbf{A} := \frac{1}{2} S \|w\|_\infty \widetilde{\mathbf{B}},\ b := \frac{1}{2} \|w\|_\infty d, \text{ and } c := S\left(\|w\|_\infty \mathbf{1}_E - w_E\right). \end{gathered} \tag{5.11}$$

We next show we can use Algorithm 16 to obtain a feasible approximate MWM on a weighted graph.

Lemma 74 (Weighted overflow removal)**.** *Suppose that for a weighted graph (V, E, w) with MWM value M^*, $\hat{x} \in \mathbb{R}^{\widetilde{E}}_{\geq 0}$ is a flow satisfying*

$$\langle w_E, \hat{x} \rangle - \|w\|_\infty \operatorname{Overflow}_d(\hat{x}) \geq M^* - \epsilon.$$

Applying RemoveOverflow *to $\hat{x}$ outputs $\tilde{x} \in \mathbb{R}^E_{\geq 0}$ with $\tilde{x} \leq x$, $\mathbf{B}^\top \tilde{x} \leq d$, and $\langle w_E, \tilde{x} \rangle \geq M^* - \epsilon$.*

Proof. By the ℓ_1-ℓ_∞ Hölder's inequality and the guarantees of Algorithm 16,

$$\|w_E\| \tilde{x} \geq \|w_E\| \hat{x} - \|w\|_\infty \|\hat{x} - \tilde{x}\|_1 \geq \|w_E\| \hat{x} - \|w\|_\infty \operatorname{Overflow}_d(\hat{x}) \geq M^* - \epsilon.$$

□

Finally, by combining Proposition 14 applied to (5.11), the cycle cancelling procedure of Proposition 16, and the rounding procedure in Algorithm 16, we obtain the following guarantee for solving (5.9), summarized in Corollary 16.

Corollary 16. *There is an algorithm that computes an n-sparse ϵ-additively approximate solution $\hat{x}$ to the problem* (5.9) *satisfying $w^\top \hat{x} - w^\top (Sx^*) \geq \epsilon$, $\mathbf{B}^\top \hat{x} \leq d$ using $O\left(\gamma \log n \log(\gamma + \|w\|_\infty \|d\|_1 \epsilon^{-1})\right)$ passes, $O(n)$ memory, and $O\left(m\gamma \log n \log(n\gamma + n\|w\|_\infty \|d\|_1 \epsilon^{-1})\right)$ work, where $\gamma = \frac{S\|w\|_\infty}{\epsilon}$.*

Proof. The proof is analogous to that of Theorem 26. We first define the steps of the algorithm, where we adjust the error parameter ϵ by an appropriate constant in each of the following subroutines.

1. Solve (5.11) to ϵ additive approximation, and let the approximate solution be x. Set $\tilde{x} = Sx_E$.
2. Apply Proposition 16 to $\tilde{x}$ in streaming fashion to obtain a flow $\hat{x}$.
3. Greedily find an exact MWM on the support of $\hat{x}$.

The correctness and runtime follow in the same way as in Theorem 26, where we use Lemma 74 in place of Lemma 73, which implies the support of $\hat{x}$ contains a ϵ-additively approximate MWM. □

We briefly remark on the utility of Corollary 16. The generality of being able to handle arbitrary costs has the downside of an additive error guarantee rather than multiplicative. We will show how to apply this general result in different settings where either the optimal solution is supported on simplex (e.g. $S = 1$ for optimal transport), or we can modify the graph appropriately to have a saturated optimal matching (e.g. $S = n$ for maximum weight matching).

5.5.3 Optimal transportation

In this section, we give a semi-streaming implementation for solving the discrete optimal transportation problem. This problem is parameterized by costs[4] $c \in \mathbb{R}^E_{\geq 0}$, and two sets of demands $\ell \in \Delta^L$, $r \in \Delta^R$ where $L \cup R = V$ is the bipartition of the vertices, we wish to find a transportation plan $x \in \Delta^E$ between the demands with (approximately) minimal cost, as specified by c; we defer a further discussion of this formulation to [25]. Appendix D.1 showed that to obtain a transport plan approximating the optimum to ϵ-additive accuracy, it suffices to solve the following problem to ϵ duality gap, where d is the vertical concatenation of ℓ and r, $C_{\max} := \|c\|_\infty$, and $\mathbf{B}$ is the adjacency matrix of the unweighted complete bipartite graph:

$$\min_{x \in \Delta^E} -c^\top x + 2C_{\max} \text{Overflow}_d(x). \tag{5.12}$$

The formulation above is an instance of the weighted formulation (5.10) with $S = 1$, $w = c$. Note here we also use the standard adjacency matrix $\mathbf{B}$ instead of the extended one $\tilde{\mathbf{B}}$ in denoting overflows

[4] Costs are without loss of generality nonnegative, as adding a uniform multiple of $\|c\|_\infty \mathbf{1}$ affects the cost of all transportation plans by a fixed amount and the quantity $C_{\max}$ by at most a constant factor.

Overflow$(\cdot)$ for simplicity as opposed to the rest of the paper. We thus apply Corollary 16 to solve the problem and conclude by giving a complete result for semi-streaming optimal transportation.

Theorem 28 (Optimal transport)**.** *There is a deterministic semi-streaming algorithm which given any optimal transport instance on a complete bipartite graph on $V = L \cup R$, costs $c \in \mathbb{R}^E_{\geq 0}$, and two sets of demands $\ell \in \Delta^L$, $r \in \Delta^R$, finds an $\epsilon \|c\|_\infty$-additive approximate optimal transport plan using $O\left(\epsilon^{-1} \log n \log \epsilon^{-1}\right)$ passes, $O(n)$ space, and $O\left(n^2 \epsilon^{-1} \log n \log \epsilon^{-1}\right)$ work.*

Proof. The proof follows by first applying Corollary 16 to (5.12), with error parameter $\epsilon \|c\|_\infty$, to obtain an $O(n)$ sparse solution x in the desired work and space budget. For demands $d \in \mathbb{R}^V$ set to be the vertical concatenation of the given ℓ and r, this is a transportation plan that satisfies $\mathbf{B}^\top x \leq d$, achieves an additive ϵ approximation in the optimal cost, and can be stored in $O(n)$ space. To make $\mathbf{B}^\top x = d$, it suffices to add a rank-one correction term as in Algorithm 2 of [25], whose coordinates can be computed in a stream in one pass and $O(n)$ space (by storing the rank-one components). This can only help the objective, and the resulting exact transport plan can be made sparse via Proposition 16 within the specified work budget. □

5.5.4 Maximum weight matching

We give a result for computing an approximate maximum weight matching for a bipartite graph in the semi-streaming model. Using the construction in Section 5.5.2 with a demand vector $d = \mathbf{1}_V$ (i.e. the vector which is all-ones on V and zeroes on $\widetilde{V} \setminus V$), it is clear the ℓ_1 norm of a maximum weight matching is n, simply by putting all additional flow on the extra edge. Applying Corollary 16 then directly gives a complete result for semi-streaming maximum weight matching.

Theorem 31. *There is a deterministic semi-streaming algorithm which given any weighted bipartite graph $G = (V, E, w)$ with $|V| = n$, $|E| = m$, and defining $\gamma := \frac{n\|w\|_\infty}{\epsilon}$, finds an ϵ-additive maximum weight matching using $O(\gamma \log n \log(n\gamma))$ passes, $O(n)$ space, and $O(m\gamma \log n \log(n\gamma))$ total work.*

Finally, in the case when we have side information upper bounding the ℓ_1 norm of any MWM by $S < n$, note that it suffices to solve (5.10) with this scale S. This implies analogous wins in Theorem 31, so that the parameter γ scales linearly in S rather than n.

5.6 Transshipment

To demonstrate the versatility of our approach, we also adapt our method to solve the transshipment problem on graphs in the semi-streaming model. In this problem we are given access to an undirected graph with non-negative edge weights $G = (V, E, w)$ and a vector $d \in \mathbb{R}^V$. Here d is a demand vector, i.e. it specifies the desired flow imbalance on every vertex, and w_e specifies the cost per routing each unit of flow (in magnitude) on edge e.

The goal of the transshipment problem is to compute a flow which routes the demands d of minimum cost, given as the sum of weighted flow magnitudes. Formally, for a flow $f \in \mathbb{R}^E$ the imbalance of the flow at every vertex is given by $\mathbf{B}^\top f$ where $\mathbf{B}$ in this section refers to the signed incidence matrix of G.[5] Consequently, the transshipment problem is $\min_{f \in \mathbb{R}^E : \mathbf{B}^\top f = d} \sum_{e \in E} w_e |f_e|$. Rescaling $f \leftarrow \mathbf{W}^{-1} f$ for $\mathbf{W} := \mathbf{diag}\,(w)$ shows that equivalently we can define the problem as follows.

Definition 19 (Transsshipment problem)**.** *For a graph $G = (V, E, w)$ with nonnegative edge weights, let $\mathbf{B} \in \{-1, 0, 1\}^{m \times n}$ be its oriented (signed) unweighted incidence matrix. For any demand vector $d \in \mathbb{R}^n$, we write the* transshipment problem*, for a diagonal positive*[6] *matrix $\mathbf{W} \in \mathbb{R}^{m \times w}$ as*

$$\min_{f : \mathbf{B}^\top \mathbf{W}^{-1} f = d} \|f\|_1 . \tag{5.13}$$

Additionally, let $\mathsf{opt}(d)$ refer to an arbitrary minimizer of problem (5.13) *for a given demand d.*

Applying our algorithmic framework, namely using the semi-streaming box-simplex game solver in Section 5.3 and our tools for rounding and maintaining sparsity, along with known techniques previously developed in recent work [358], we prove Theorem 29:

Theorem 29 (Approximate transshipment and shortest path)**.** *There is a randomized semi-streaming algorithm which given weighted undirected $G = (V, E, w)$ and demand vector $d \in \mathbb{R}^n$, finds an ϵ-multiplicatively approximate minimizer to the minimum transshipment cost in $O(\log^{O(1)} n \cdot \epsilon^{-1})$ passes, $O(n \log^{O(1)} n)$ space, and $O(m \log^{O(1)} n \cdot \epsilon^{-1})$ total work with high probability in n. This, in particular, yields a semi-streaming algorithm that ϵ-multiplicative approximates the s-t shortest path problem with the same pass, space, and work complexities.*

We prove this result in several parts. We begin by giving a semi-streaming construction of an ℓ_1-stretch approximator matrix $\mathbf{R} \in \mathbb{R}^{K \times n}$, defined in the following (a similar definition is given in [358]). Our semi-streaming implementation is largely based on known tools developed in [358].

Definition 20 (Stretch approximator)**.** $\mathbf{R} \in \mathbb{R}^{K \times n}$ *is an* (α, β)-stretch approximator *if it satisfies the following.*

- For any $d \in \mathbb{R}^n$, $\|\mathsf{opt}(d)\|_1 \le \|\mathbf{R} d\|_1 \le \alpha \|\mathsf{opt}(d)\|_1$.
- $\sum_{v \in V} \deg(v) \mathrm{nnz}(\mathbf{R}_{:v}) \le m\beta$, where $\deg(v)$ is the degree of v in G and $\mathrm{nnz}(\mathbf{R}_{:v})$ is the number of nonzero entries in the v^{th} column of $\mathbf{R}$.
- $\mathbf{R}$ has at most $K\beta$ nonzero entries.

[5] We overload the notation $\mathbf{B}$ for just this section, the only application other than Appendix D.5 where it is signed. The signed unweighted incidence matrix is defined in the same way as the unsigned one, except we choose an arbitrary orientation for every edge $e = (u, v)$ so the corresponding row in $\mathbf{B}$ has a 1 in column u and a -1 in column v.

[6] For simplicity, we assume no zero-weight edges. Otherwise, we can form a spanning forest in one pass to remove zero-weight edges (since these shortcut the transshipment problem), and decompose into connected components.

In Section 5.6.1, we provide an algorithm which construct an (α, β)-stretch approximator $\mathbf{R} \in \mathbb{R}^{K \times n}$ with $K = O(n \log n)$ and $\alpha, \beta = \log^{O(1)} n$; we remark we did not try to control for the logarithmic factors, as many are inherited from prior work. In Section 5.6.2, we then use our stretch approximator to reduce a decision variant of transshipment to an appropriate box-simplex game. In Section 5.6.3, we demonstrate how to use our cycle-cancelling toolkit to round an approximate transshipment plan f to be sparse. In Section 5.6.4 we we put these pieces together to prove Theorem 29.

5.6.1 Constructing stretch approximators

In this section, we give a semi-streaming construction of a stretch approximator. Our construction is a straightforward adaptation of a previous parallel construction by [358], proceeding in two steps. First we compute H, an $O(\log n)$-spanner of G using $O(\log n)$ semi-streaming passes. We apply the construction of [358] to H offline, which no longer requires additional access to the edges of G. We then prove the approximation and sparsity bounds of the obtained matrix.

Definition 21 (Spanner)**.** *We say subgraph $H \subseteq G$ is an σ-spanner of graph $G = (V, E)$ if for all $u, v \in V$, where $d_G(u, v)$ denotes the shortest path distance through G,*

$$d_G(u, v) \leq d_H(u, v) \leq \sigma d_G(u, v).$$

It is well-known that Definition 21 implies that the optimal value of the objective (5.13) is preserved up to a multiplicative σ factor when restricting to H. This follows as every path in a decomposition of the solution to (5.13) has its cost preserved up to a σ factor upon routing through H.

Lemma 75 (Corollary 5.2 of [67])**.** *Let $G = (V, E, w)$ be a weighted graph given in an insertion-only stream. An $O(\log n)$-spanner of G with $O(n \log n)$ edges can be computed using $O(\log n)$ passes, $O(n \log n)$ space, and $O(m \log n)$ total work.*

Lemma 76 (Theorem 4.2 of [358])**.** *Given a graph $G = (V, E, w)$, we can compute a matrix $\mathbf{R} \in \mathbb{R}^{O(n \log n) \times n}$ satisfying the following properties with high probability in n:*

- $\|\mathsf{opt}(d)\|_1 \leq \|\mathbf{R}d\|_1 \leq O(\log^{4.5} n) \|\mathsf{opt}(d)\|_1$ *for all $d \in \mathbf{R}^n$.*
- *For any $v \in V$, the v^{th} column of $\mathbf{R}$ has $O(\log^5 n (\log \log n)^{O(1)})$ nonzero entries in expectation over the construction of $\mathbf{R}$.*

The algorithm runs in $O(n \log^{10} n (\log \log n)^{O(1)})$ work and space.

Proof. The first and third condition on $\mathbf{R}$ follow directly from Theorem 4.2 of [358]. The second condition follows from Lemma 4.15 and the proof of Lemma 4.16 of [358]. □

Lemma 77. *Let $G = (V, E, w)$ be a weighted undirected graph given in the semi-streaming graph model. Algorithm 17 computes $\mathbf{R} \in \mathbb{R}^{K \times n}$, an (α, β)-stretch approximator with $K = O(n \log n)$, $\alpha = O(\log^{5.5} n)$, and $\beta = O(\log^5 n (\log \log n)^{O(1)})$, in $O(\log n)$ passes, $O(n \log^{10} n (\log \log n)^{O(1)})$ space, and $O(m \log^{10} n (\log \log n)^{O(1)})$ work with high probability in n.*

Proof. We first show correctness of the algorithm. First, we note that the cost of transshipment over H is at most $O(\log n)$ times greater than the cost over G since $d_G(u, v) \leq d_H(u, v) \leq O(\log n) d_G(u, v)$ for all $u, v \in V$. We thus have $\|\mathsf{opt}(d)\|_1 \leq \|\mathbf{R}d\|_1 \leq O(\log^{5.5} n) \|\mathsf{opt}(d)\|_1$ by the first condition of Lemma 76 applied to the matrix $\mathbf{R}$ we return; this guarantee is with high probability, by a union bound over all the routing constructions. Next, the second condition of Lemma 76 implies that each column of $\mathbf{R}$ computed on line 6 of Algorithm 17 has γ nonzero entries in expectation. Thus $\mathbb{E}\left[\sum_{v \in V} \text{nnz}(\mathbf{R}_{:v})\right] \leq n\gamma$ and

$$\mathbb{E}\left[\sum_{(u,v) \in E} \text{nnz}(\mathbf{R}_{:u}) + \text{nnz}(\mathbf{R}_{:v})\right] = 2\mathbb{E}\left[\sum_{v \in V} \mathsf{deg}(v) \text{nnz}(\mathbf{R}_{:v})\right] \leq 2m\gamma.$$

By Markov's inequality and a union bound, we thus have $S \leq 6m\gamma$ and $\text{nnz}(\mathbf{R}) \leq 3n\gamma$ with probability at least $\frac{1}{3}$ over the randomness of Lemma 76. Consequently, with high probability in n after $O(\log n)$ repetitions the algorithm returns a matrix $\mathbf{R}$ with all the desired properties.

We now bound the pass, space, and work complexity. By Lemma 75, the graph H can be computed using $O(\log n)$ passes over G using $O(n \log n)$ space and $O(m \log n)$ total work. Next, the matrix $\mathbf{R}$ on Line 6 of Algorithm 17 can be computed offline in $O(m \log^{10} n (\log \log n)^{O(1)})$ work and $O(n \log^{10} n (\log \log n)^{O(1)})$ space. The condition on Line 7 can be checked in a single pass over G with $O(m\gamma)$ extra work (as we can terminate if the check fails). □

Algorithm 17: StretchApprox(G)

```
1 Input: Graph G = (V, E, w) ;
2 Output: ℓ1-stretch approximator R;
3 γ = O(log^5 n(log log n)^{O(1)});
4 H = O(log n)-spanner of G computed by Corollary 5.2 of [67];
5 for t ∈ [O(log n)] do
6     R = matrix computed by Theorem 4.2 of [358] applied to H;
7     Compute S = Σ_{v∈V} deg(v)nnz(Re_v) in a stream over G;
8     if S ≤ 6mγ and nnz(R) ≤ 3nγ then
9         Return: R;
```

5.6.2 Reduction to box-simplex game

In this section, we describe our reduction of transshipment to a box-simplex game. We begin by defining a flow-constrained variant of the transshipment problem.

Definition 22. *Let $G = (V, E, w)$ be a graph, and let $d \in \mathbb{R}^n$. Let $\mathbf{R}$ be any matrix satisfying $\|\mathsf{opt}(d)\|_1 \leq \|\mathbf{R}d\|_1$. For any $t \geq 0$, we define the* flow-constrained *transshipment problem as*

$$\min_{f:\|f\|_1 \leq t} \left\|\mathbf{R}\mathbf{B}^\top \mathbf{W}^{-1} f - \mathbf{R}d\right\|_1. \tag{5.14}$$

We next relate solutions to the flow-constrained transshipment problem to the original problem (5.13).

Lemma 78. *Let G be a graph, $d \in \mathbb{R}^n$, and $\mathbf{R}$ satisfy $\|\mathsf{opt}(d)\|_1 \leq \|\mathbf{R}d\|_1$.*

- *If $t \geq \|\mathsf{opt}(d)\|_1$, the optimal value of (5.14) is 0.*
- *If $t < \|\mathsf{opt}(d)\|_1$, the optimal value of (5.14) is at least $\|\mathsf{opt}(d)\|_1 - t$.*

Proof. For notational convenience, let $f^* = \mathsf{opt}(d)$. By definition we have $\mathbf{B}^\top \mathbf{W}^{-1} f^* = d$. If $t \geq \|f^*\|_1$, we note f^* is feasible for (5.14) and therefore achieves an objective value of 0. If instead $t < \|f^*\|_1$, consider any f with $\|f\|_1 \leq t$ and define $f' = \mathsf{opt}(d - \mathbf{B}^\top \mathbf{W}^{-1} f)$. We have

$$\left\|\mathbf{R}\mathbf{B}^\top \mathbf{W}^{-1} f - \mathbf{R}d\right\|_1 = \left\|\mathbf{R}(d - \mathbf{B}^\top \mathbf{W}^{-1} f)\right\|_1 \geq \|f'\|_1,$$

where for the last inequality we use the definition of $\mathbf{R}$ and the optimality of f'. On the other hand we have $\mathbf{B}^\top \mathbf{W}^{-1} f' = d - \mathbf{B}^\top \mathbf{W}^{-1} f$ and so $\mathbf{B}^\top \mathbf{W}^{-1}(f + f') = d$. By optimality of f^* for (5.13),

$$\|f^*\|_1 \leq \|f + f'\|_1 \leq \|f\|_1 + \|f'\|_1.$$

Combining these inequalities, we have the desired

$$\left\|\mathbf{R}\mathbf{B}^\top \mathbf{W}^{-1} f - \mathbf{R}d\right\|_1 \geq \|f^*\|_1 - \|f\|_1 \geq \|f^*\|_1 - t.$$

□

We remark that any (α, β)-stretch approximator $\mathbf{R}$ satisfies the assumption of Lemma 78. Finally, we demonstrate how to express problems of the form (5.14) as box-simplex games.

Lemma 79. *Let G be a graph, $d \in \mathbb{R}^n$, and $t \geq 0$. Let $\mathbf{R} \in \mathbb{R}^{K \times n}$ be an (α, β)-stretch approximator. Then (5.14) is equivalent to*

$$\min_{f' \in \Delta^{2E}} \left\|t\mathbf{A}^\top f' - b\right\|_1 = \min_{f' \in \Delta^{2E}} \max_{y \in [-1,1]^K} t y^\top \mathbf{A}^\top f' - b^\top y$$

for $b = \mathbf{R}d$ *and*

$$\mathbf{A} = \begin{pmatrix} \mathbf{W}^{-1}\mathbf{B}\mathbf{R}^\top \\ -\mathbf{W}^{-1}\mathbf{B}\mathbf{R}^\top \end{pmatrix}.$$

Additionally, $\|\mathbf{A}\|_\infty \leq \alpha$ *and* $\mathrm{nnz}(\mathbf{A}) = O(m\beta)$*. If* R *is stored explicitly, we may simulate access to a stream of the rows of* $\mathbf{A}$ *and* $|\mathbf{A}|$ *using* $O(m\beta)$ *total work and a single pass over* G*.*

Proof. By duality of the ℓ_1 norm and the ℓ_∞ norm, (5.14) is equivalent to

$$\min_{f:\|f\|_1 \leq t} \left\|\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}f - \mathbf{R}d\right\|_1 \equiv \min_{f:\|f\|_1 \leq t} \max_{y:\|y\|_\infty \leq 1} y^\top\left(\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}f - \mathbf{R}d\right).$$

Next, we can write any $f \in \mathbb{R}^E$ as $f = f_+ - f_-$, where $f_+ = \max\{f, 0\}$ and $f_- = -\min\{f, 0\}$ are entrywise nonnegative. Further, if $\|f\|_1 \leq t$ there exists nonnegative f_+ and f_- satisfying this condition with $\mathbf{1}^\top f_+ + \mathbf{1}^\top f_- = t$: one can simply select an arbitrary edge in G and increase the corresponding entries in f_+ and f_- by the same amount, which increases $\|f\|_1$ without affecting $f_+ - f_-$. Thus, for any f with $\|f\|_1 \leq t$ there exists $\hat{f} = \frac{1}{t}[f_+; f_-] \in \Delta^{2E}$ such that

$$f = t\begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \end{pmatrix}^\top \hat{f}.$$

Applying this variable substitution, our problem is equivalent to

$$\min_{\hat{f}\in\Delta^{2E}} \max_{\|y\|_\infty \leq 1} ty^\top\mathbf{A}^\top\hat{f} - y^\top b$$

as desired. The bound on $\|\mathbf{A}\|_\infty$ follows from the observation

$$\left\|\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}u\right\|_1 \leq \alpha\left\|\mathsf{opt}\left(\mathbf{B}^\top\mathbf{W}^{-1}u\right)\right\|_1 \leq \alpha\|u\|_1$$

for any $u \in \mathbb{R}^E$, by the first condition of Definition 20 and definitions of $\mathbf{B}, \mathbf{W}$. This yields

$$\|\mathbf{A}\|_\infty = \left\|\mathbf{A}^\top\right\|_1 = \max_{\|v\|_1=1}\left\|\mathbf{A}^\top v\right\|_1 \leq \max_{\|v_1\|_1+\|v_2\|_1=1}\left\|\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}v_1\right\|_1 + \left\|\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}v_2\right\|_1 \leq \alpha$$

as claimed. We conclude by bounding $\mathrm{nnz}(\mathbf{A})$ and showing that we can simulate streaming access to the columns of $\mathbf{A}$ and $|\mathbf{A}|$ in $O(m\beta)$ total work. Let δ_v denote the number of nonzero entries in the v^{th} column of $\mathbf{R}$. The column of $\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}$ corresponding to $e = (u, v) \in E$ is of the form

$$w_e^{-1}\left(\mathbf{R}_{:u} - \mathbf{R}_{:v}\right),$$

and thus has $O(\delta_u + \delta_v)$ nonzero entries. By the second assumed condition on $\mathbf{R}$, summing over

all columns gives $O(m\beta)$ nonzero entries together, thus the bound on $\mathrm{nnz}(\mathbf{A})$ follows. Finally, we may simulate access to columns of $\mathbf{A}^\top$ and $|\mathbf{A}^\top|$ via a stream over G and forming the corresponding columns of $\mathbf{A}^\top$ and $|\mathbf{A}^\top|$ directly using $O(m\beta)$ work, from the above characterization of $\mathbf{R}\mathbf{B}^\top\mathbf{W}^{-1}$. □

5.6.3 Recovering a sparse flow

We next give a simple procedure for taking a flow f and rounding it to a sparse flow f', such that the weighted cost is no larger and the marginal imbalances are preserved. At a high level, our algorithm (Algorithm 18) performs the following steps.

1. We form the "double cover graph" of $G = (V, E, w)$, a bipartite graph consisting of vertices $V_{\mathrm{in}} \cup V_{\mathrm{out}}$, which are two copies of V. For every $(u, v) \in E$, the double cover graph contains edges $(u_{\mathrm{in}}, v_{\mathrm{out}})$, denoting positive flow, and $(u_{\mathrm{out}}, v_{\mathrm{in}})$, denoting negative flow.
2. We sparsify both the positive flow f_+ and the negative flow f_- using Proposition 16.
3. We identify the sparsified flows into the original graph by using the signs appropriately, preserving marginal demands and not hurting the weighted cost.

We make the transformation $f \leftarrow \mathbf{W}f$ in the following algorithm for consistency with Proposition 16, so the marginal imbalances in accordance with transshipment are $\mathbf{B}^\top f$ and the ℓ_1 weight is $\|\mathbf{W}f\|_1$.

Algorithm 18: RoundStream(f_+, f_-)

Input: Incremental stream of $f_+, f_- \in \mathbb{R}^E_{\geq 0}$ on disjoint supports, graph $G = (V, E, w)$;
Output: Flow $f' \in \mathbb{R}^E$ with

$$\|f'\|_0 = O(n),\ \mathbf{B}^\top f' = \mathbf{B}^\top(f_+ - f_-),\ \|\mathbf{W}f'\|_1 \leq \|\mathbf{W}f_+\|_1 + \|\mathbf{W}f_-\|_1 .$$

;
Form $G'_+ = (V' = V_{\mathrm{in}} \cup V_{\mathrm{out}}, E'_+ = \emptyset)$ and $G'_- = (V' = V_{\mathrm{in}} \cup V_{\mathrm{out}}, E'_- = \emptyset)$;
for $x_{(u,v)}\mathbf{1}_{(u,v)}$ *in stream over* f_+ **do**
  Add edge $(u_{\mathrm{in}}, v_{\mathrm{out}})$ with weight $x_{(u,v)}$ to G'_+;
  Apply Proposition 16 to G'_+;
for $x_{(u,v)}\mathbf{1}_{(u,v)}$ *in stream over* f_- **do**
  Add edge $(u_{\mathrm{out}}, v_{\mathrm{in}})$ with weight $x_{(u,v)}$ to G'_-;
  Apply Proposition 16 to G'_-;
Let f'_+ be the output of Proposition 16 on G'_+ and f'_- the output of Proposition 16 on G'_-;
Return: $f'_+ - f'_-$;

Lemma 80. *Algorithm 18 satisfies its output statement using a single pass over the streams f_+ and f_-, $O(n)$ space, and $O(\mathrm{nnz}(f_+ + f_-)\log n)$ work.*

Proof. The pass, space, and work complexities follow from Proposition 16. The sparsity of f' and the guarantee on weighted ℓ_1 norm follows from the two applications of Proposition 16, since f_+ and f_- have disjoint supports. Finally, by construction of G'_+ and G'_-, the signed marginals of f_+ are preserved by f'_+, and the signed marginals of f_- are preserved by f'_-. The conclusion follows since $f_+ - f_-$ then places the same amount of net flow on any vertex as $f'_+ - f'_-$. □

5.6.4 Semi-streaming transshipment

We finally give our full algorithm to solve the transshipment problem, and a proof of Theorem 29.

Algorithm 19: $\mathsf{ApproxTransshipment}(G, d, \epsilon)$

1 **Input:** Graph $G = (V, E, w)$, demand vector $d \in \mathbb{R}^n$, error tolerance $\epsilon \in (0, 1)$;
2 **Output:** Flow f with $\|f\|_0 = O(n), \mathbf{B}^\top \mathbf{W}^{-1} f = d$, and $\|f\|_1 \le (1+\epsilon) \|\mathsf{opt}(d)\|_1$.;
3 $H = O(\log n)$-spanner of G computed by Corollary 5.2 of [67];
4 $\mathbf{R} = \mathsf{StretchApprox}(G)$;
5 $t_{\max} \leftarrow$ 2-approximation to $\min_{\mathbf{B}_H^\top \mathbf{W}_H^{-1} f = d} \|f\|_1$, scaled to be an overestimate to $\mathsf{opt}(d)$;
▷ We let $\mathbf{B}_H$ and $\mathbf{W}_H$ be the appropriate restrictions of $\mathbf{B}$ and $\mathbf{W}$ to the subgraph H.
6 $t_{\min} \leftarrow \frac{t_{\max}}{O(\log n)}$;
7 **while** $t_{\max} \ge (1 + \frac{\epsilon}{\log n}) t_{\min}$ **do**
8 $t \leftarrow \frac{1}{2}(t_{\min} + t_{\max})$;
9 $\mathbf{A} \leftarrow t \begin{pmatrix} \mathbf{R}\mathbf{B}^\top \mathbf{W}^{-1} & -\mathbf{R}\mathbf{B}^\top \mathbf{W}^{-1} \end{pmatrix}^\top, b \leftarrow \mathbf{R}d$;
10 $Z, f_{\mathsf{stream}} \leftarrow$ approximate value and streamed solution of (5.2) with $\mathbf{A}$, b as defined above, and $c = \mathbf{0}$, to accuracy $\frac{\epsilon t}{O(\log n)}$;
11 **if** $Z \le \frac{\epsilon t}{O(\log n)}$ **then**
12 $t_{\max} \leftarrow t$;
13 **else**
14 $t_{\min} \leftarrow t$;
15 $f' \leftarrow \mathsf{RoundStream}([\mathbf{W} f_{\mathsf{stream}}]_+, [\mathbf{W} f_{\mathsf{stream}}]_-)$ where $[\mathbf{W} f_{\mathsf{stream}}]_+$, $[\mathbf{W} f_{\mathsf{stream}}]_-$ are the restrictions of f_{stream} to the top and bottom rows of $\mathbf{A}$ respectively after cancelling any nonnegative flow on corresponding edges in both parts;
16 $d' \leftarrow d - \mathbf{B}^\top f'$;
17 $f_{\mathsf{res}} \leftarrow$ 2-approximate solution to $\min_{\mathbf{B}_H^\top \mathbf{W}_H^{-1} f = d'} \|f\|_1$;
18 **Return:** $\mathbf{W}^{-1} (\mathsf{RoundStream}([f' + \mathbf{W} f_{\mathsf{res}}]_+, [f' + \mathbf{W} f_{\mathsf{res}}]_-))$;

Proof of Theorem 29. We begin by proving correctness of Algorithm 19, and then discuss implementation costs to show they fit within the specified budgets.

First, $t_{\max}$ is an overestimate of $\mathsf{opt}(d)$ by the definition of a spanner in Definition 21, and similarly $t_{\min}$ is an underestimate. Next, by solving the problem in Line 8 to the stated accuracy, Lemma 79 shows we have an $\frac{\epsilon t}{O(\log n)}$-additive approximation to the value of the problem (5.14). Hence, Lemma 78 shows that when the binary search terminates, t is an $O(\frac{\epsilon}{\log n})$-multiplicative

approximation to $\|\mathsf{opt}(d)\|_1$. The additive error incurred due to the approximate routing of demands d' on Line 15 can only affect the solution by $O(\epsilon)$ multiplicatively because of the quality of the spanner. Moreover, the applications of RoundStream do not affect meeting the demands and can only improve the ℓ_1 value, by Lemma 80. Hence, the output of the algorithm meets the demands and attains an ϵ-multiplicative approximation to the value of the transshipment problem (5.13).

We next discuss pass, space, and work complexities. The costs of Lines 1 and 2 are given by Lemma 75 and Lemma 77 respectively. Lines 3 and 15 can be performed without streaming access to G once H has been stored, e.g. using [358], and do not dominate any of the costs.

There are at most $\log \frac{\log n}{\epsilon}$ iterations of the binary search, given the multiplicative range on t. The cost of each run in Line 8 is given by Theorem 30, where $\|\mathbf{A}\|_\infty = t\alpha$ for the α in Lemma 77, and where the additive accuracy is given in Line 8. Finally, the costs of calls to RoundStream are given by Lemma 80 and do not dominate, completing the proof.

To conclude, we describe our algorithm for $(1+\epsilon)$-approximate shortest paths. We choose the demands $d = \mathbf{1}_u - \mathbf{1}_v$ for transshipment, and consider the flow f computed by Algorithm 19. We simply return the shortest s-t path contained in the support of f: as f satisfied $\mathbf{B}^\top f = d$, it corresponds to a linear combination of s-t paths. Returning the shortest path found in its support must therefore have length at most that of f, which is itself $\leq (1+\epsilon)\mathsf{opt}(d)$. □

Chapter 6

Dynamic Decremental Bipartite Matching

This chapter is based on [284], with Arun Jambulapati, Yujia Jin, and Aaron Sidford.

6.1 Introduction

Efficient approximate solvers for graph-structured convex programming problems have led to a variety of recent advances in combinatorial optimization. Motivated by problems related to maximum flow and optimal transportation, a recent line of work [482, 318, 483], as well as Chapters 4, 2, and 5 developed near-linear time, accelerated solvers for a particular family of convex programming objectives we refer to in this paper as *box-simplex games*:

$$\min_{x\in\Delta^m}\max_{y\in[-1,1]^n} y^\top \mathbf{A}x + c^\top x - b^\top y \text{ where } \Delta^m := \{x \in \mathbb{R}^m_{\geq 0} | \, \|x\|_1 = 1\}. \tag{6.1}$$

Box-simplex games, (6.1), are bilinear problems where a maximization player is constrained to the box (the ℓ_∞ ball) and a minimization player is constrained to the simplex (the nonnegative ℓ_1 shell). The problem provides a convenient encapsulation of linear programming problems with ℓ_1 or ℓ_∞ structure; (6.1) can be used to solve box-constrained ℓ_∞ regression problems (see e.g. [483, 486]) and maximizing over the box-constrained player yields the following ℓ_1 regression problem

$$\min_{x\in\Delta^m} c^\top x + \|\mathbf{A}x - b\|_1 . \tag{6.2}$$

Furthermore, solvers for (6.1) and (6.2) are used in state-of-the-art algorithms for approximate maximum flow [483], optimal transport (OT) [293], (width-dependent) positive linear programming [91], and semi-streaming bipartite matching [39].

One of the main goals of our work is to develop efficient algorithms for solving *regularized* variants of the problems (6.1) and (6.2). An example of particular interest is the following

$$\min_{x\in\Delta^m|\mathbf{B}^\top x=d} c^\top x+\mu H(x),\ \text{where } \mu\geq 0 \text{ and } H(x):=\sum_{i\in[m]} x_i\log x_i. \tag{6.3}$$

The particular case of (6.3) when $\mathbf{B}\in\mathbb{R}^{m\times n}$ is the (unsigned) edge-vertex incidence matrix of a complete bipartite graph, and d is a pair of discrete distributions supported on the sides of the bipartition, is known as the *Sinkhorn distance* objective. This objective is used in the machine learning literature [151] as an efficiently-computable approximation to optimal transport distances: c corresponds to pairwise movement costs, and d encodes the prescribed marginals. This objective has favorable properties, e.g. differentiability [523], and there has been extensive work by both theorists and practitioners to solve (6.3) and analyze its properties (see e.g. [151, 25] and references therein). Choosing $\mathbf{A}$ and b to be sufficiently large multiples of $\mathbf{B}^\top$ and d, it can be shown that solutions to the following regularized variant of (6.2) yield approximate solutions to (6.3),

$$\min_{x\in\Delta^m} c^\top x+\|\mathbf{A}x-b\|_1+\mu H(x)\,. \tag{6.4}$$

Beyond connections to Sinkhorn distances, there are additional reasons why it may be desirable to solve regularized box-simplex games. For example, regularization could speed up algorithms and allow high-precision solutions to be computed more efficiently. Further, obtaining a high-precision solution to a regularized version of the problem yields a more canonical and predictable approximate solution than an arbitrary low-precision approximation to the unregularized problem. Moreover, regularization potentially makes optimal solutions more stable to input changes. For box-simplex games stemming from bipartite matching we quantify this stability and show all of these properties allow regularized solvers to yield faster algorithms for a particular dynamic matching problem.

Altogether, the main contributions of this paper are the following.

1. We give improved running times for the problem of *dynamic decremental bipartite matching* (DDBM) with an *adaptive adversary*, a fundamental problem in dynamic graph algorithms. Our algorithm follows from a general black-box reduction we develop from DDBM to solving (variants of) regularized box-simplex games to high precision.

2. We give efficient solvers for (variants of) the regularized box-simplex problems (6.3), (6.4).

3. As a byproduct, we also show how to apply our new solvers (and additional tools from the literature) to obtain state-of-the-art methods for computing Sinkhorn distances.

Formally, the *DDBM* problem we consider in this paper is the following: given a bipartite graph undergoing edge deletions maintain, at all times, an *ϵ-approximate (maximum) matching*,[1] that is

[1]This is sometimes also referred to as a $(1+\epsilon)$-multiplicatively approximate matching in the literature.

a matching which has size at least a $(1-\epsilon)$-fraction of the maximum (for a pre-specified) value of ϵ. Unless specified otherwise, we consider the *adaptive adversary model* where edge deletions can be specified adaptively to the matching returned. Further, unless specified otherwise, we allow the matching output by the algorithm to be *fractional*, rather than integral.

We show how to reduce solving the DDBM problem to solving a sequence of regularized box-simplex games. This reduction yields a new approach to dynamic matching; this approach is inspired by prior work, e.g. [73], but conceptually distinct in that it decouples the solving of optimization subproblems from characterizing their solutions. For our specific DDBM problem, the only prior algorithm achieving an amortized polylogarithmic update time (for constant ϵ) is in the recent work of [73], which derives their dynamic algorithm as an application of two techniques: *congestion balancing* and *local flows*. Our reduction eschews these combinatorial tools and directly argues, via techniques from convex analysis, that solutions to appropriate regularized matching problems can be used dynamically as approximate matchings while requiring few recomputations. We emphasize our use of fast *high-accuracy solvers*[2] in the context of our reduction to obtain our improved runtimes, as our approach leverages structural characteristics of the exact solutions which we only show are inherited by approximate solutions when solved to sufficient accuracy.

Our work both serves as a proof-of-concept of the utility of regularized linear programming solvers as a subroutine in dynamic graph algorithms, and provides the tools necessary to solve said problems in various structured cases. This approach to dynamic algorithm design effectively separates a "stability analysis" of the solution to a suitable optimization problem from the computational burden of solving that problem to high accuracy: any improved solver would then have implications for faster dynamic algorithms as well. As a demonstration of this flexibility, we give three uses of our reduction framework (which proceed via different solvers) in obtaining our improved DDBM update time. We hope our work opens the door to exploring the use of the powerful continuous optimization toolkit, especially techniques originally designed for non-dynamic problems, for their dynamic counterparts.

Chapter organization. We give a detailed overview of our contributions in Section 6.1.1, and overview related prior work in Section 6.1.2. We state preliminaries in Section 6.2. In Section 6.3.1, we describe our framework for reducing DDBM to a sequence of regularized optimization problems satisfying certain properties, and in Section 6.3.2 we give three different instantiations of the DDBM framework for obtaining a variety of DDBM solvers. Finally in Section 6.4 we provide our main algorithm for regularized box-simplex games. We defer proofs for Section 6.3 and Section 6.4 to Appendix E.1 and Appendix E.2 respectively, and provide additional results for approximating Sinkhorn distances efficiently in Appendix E.3.

[2]Throughout, we typically use the term "high-accuracy" to refer to an algorithm whose runtime scales polylogarithmically in the inverse accuracy (as opposed to e.g. polynomially).

6.1.1 Our results

A framework for faster DDBM. We develop a new framework for solving the DDBM problem of computing an ϵ-approximate maximum matching in a dynamic graph undergoing edge deletions from an adaptive adversary. Our framework provides a reduction from this DDBM problem to solving various regularized formulations of box-simplex games.

To illustrate the reduction, suppose we have a bipartite graph $G = (V, E)$, and suppose, for simplicity, that we know M^*, the size of the (maximum cardinality) matching. As demonstrated in Chapter 5, solving the ℓ_1 regression problem $\min_{x \in M^* \Delta^m} -c^\top x + \|\mathbf{A}x - b\|_1$, to ϵM^* additive accuracy for appropriate choices of $\mathbf{A}$, b, and c yields an ϵ-approximate maximum cardinality matching. Intuitively, $\mathbf{A}$ and b penalize violations of the matching constraints, and c is a multiple of the all-ones vector capturing the objective of maximizing the matching size. However, ℓ_1 regression objectives do not necessarily have unique minimizers: as such the output of directly minimizing these objectives is difficult to characterize beyond (approximate) optimality. This induces difficulty in using solutions to such problems directly in dynamic graph algorithms.

Our first key observation (building upon intuition from congestion balancing [73]) is that, beyond enabling faster runtime guarantees, regularization provides more robust solutions which are resilient to edge deletions in dynamic applications. We show that if

$$x^*_\epsilon := \min_{x \in M^* \cdot \Delta^m} -c^\top x + \|\mathbf{A}x - b\|_1 + \epsilon H(x) \tag{6.5}$$

is the solution to the *regularized* box-simplex formulation of bipartite matching, then x^*_ϵ enjoys favorable stability properties allowing us to argue about its size under deletions.

The stability of solutions to (6.5) is fairly intuitive: the entropy regularizer $H(x)$ encourages the objective to spread the matching uniformly among edges, when all else is held equal. For example, when G is a complete bipartite graph on $2n$ vertices, standard linear programming relaxations of matching do not favor either of (i) an integral perfect matching, and (ii) a fractional matching spreading mass evenly across many edges, over the other. However, using (i) as our approximate matching on a dynamic graph undergoing deletions is substantially more unstable; an adaptive adversary can remove edges corresponding to our matching, forcing $\Omega(n)$ recomputations. On the other hand, no edge deletions can cause this type of instability for strategy (ii): as each edge receives weight $\frac{1}{n}$ in the fractional matching, the only way to reduce the fractional matching size by ϵn is to remove $O(\epsilon n^2)$ edges: thus $O(\epsilon^{-1})$ recomputations are (intuitively) sufficient for maintaining an ϵ-approximate maximum matching. This distinction underlies the use of *high-accuracy* solvers in our reduction; indeed, while they obtain large matching values in an original graph, approximate solutions may not carry the same types of dynamic matching value stability. We note similar intuition motivated the approach in [73].

To make this argument more rigorous, consider using x^*_ϵ as our approximate matching for a

number of iterations corresponding to edge deletions, until its size restricted to the smaller graph has decreased by a factor of $1 - O(\epsilon)$. By using strong convexity of (6.5) in the ℓ_1 norm, we argue that whenever the objective value of x^*_ϵ has worsened, the maximum matching size itself must have gone down by a (potentially much smaller) amount. A tighter characterization of this strong convexity argument shows that we only need to recompute a solution to slight variants of (6.5) roughly $\widetilde{O}(\epsilon^{-2})$ times throughout the life of the algorithm. Combined with accelerated $\widetilde{O}(\frac{m}{\epsilon})$-time solvers for regularized box-simplex games (which are slight modifications of (6.5)), this strategy yields an overall runtime of $\widetilde{O}(\frac{m}{\epsilon^3})$, improving upon the recent state-of-the-art decremental result of [73].

We formalize these ideas in Section 6.3, where we demonstrate that a range of regularization strategies (see Definition 23) such as (6.5) are amenable to this reduction. Roughly, as long as our regularized objective is "at least as strongly convex" as the entropic regularizer, and closely approximates the matching value in the static setting, then it can be used in our DDBM algorithm. Combining this framework with solvers for regularized matching problems, we give three different results. The first two obtain amortized update times of roughly $\widetilde{O}(\epsilon^{-3})$, in Theorems 38 and 39 via box-simplex games and matrix scaling, respectively (though the latter holds only for dense graphs). We give an informal statement of the former here.

Theorem 32 (informal, see Theorem 38)**.** *Let $G = (V, E)$ be bipartite, $|V| = n$, $|E| = m$, and $\epsilon \geq \text{poly}(m^{-1})$. There is a deterministic algorithm maintaining an ϵ-approximate matching in a dynamic bipartite graph with adversarial edge deletions running in time $O(m \log^5 m \cdot \epsilon^{-3})$.*

We note that our algorithm (deterministically) returns a *fractional matching*. There is a black-box reduction from dynamic fractional matching maintenance to dynamic integral matching maintenance contained in [527], but to our knowledge this reduction is bottlenecked at an amortized $\widetilde{O}(\epsilon^{-4})$ runtime (see e.g. Appendix A.2, [73]). Improving this reduction is a key open problem.

High-accuracy solvers for regularized box-simplex games. To use our DDBM framework, we give a new algorithm for solving regularized box-simplex games of the form:

$$\min_{x \in \Delta^m} \max_{y \in [0,1]^n} f_{\mu,\epsilon}(x, y) := y^\top \mathbf{A}^\top x + c^\top x - b^\top y + \mu H(x) - \frac{\epsilon}{2} \left(y^2\right)^\top |\mathbf{A}|^\top x, \tag{6.6}$$

where ϵ and $\mu = \Omega(\epsilon)$ are regularization parameters and y^2, $|\mathbf{A}|$ denote entrywise operations. The regularization terms $H(x)$ and $(y^2)^\top |\mathbf{A}|^\top x$ in (6.6) are parts of a primal-dual regularizer proposed in [293] (and a variation of a similar regularizer of [483]) used in state-of-the-art algorithms for approximately solving (unregularized) box-simplex games. This choice of regularization enjoys favorable regularization properties over the joint box-simplex domain, and thereby sidesteps the infamous ℓ_∞-strong convexity barrier that has limited previous attempts at accelerated algorithms for this problem. Under relatively mild restrictions on problem parameters (see discussion at the start of Section 6.4), we develop a *high accuracy solver* for (6.6), stated informally here.

Theorem 33 (informal, see Theorem 41)**.** *Given an instance of* (6.6)*, with* $\mu = \Omega(\epsilon)$*,* $\|\mathbf{A}\|_\infty \leq 1$*, and* $\sigma \geq \text{poly}(m^{-1})$ *Algorithm 23 returns* x *with* $\max_{y\in[0,1]^n} f_{\mu,\epsilon}(x,y) - f_{\mu,\epsilon}(x^\star,y^\star) \leq \sigma$ *in time* $\widetilde{O}(\text{nnz}(\mathbf{A}) \cdot \frac{1}{\sqrt{\mu\epsilon}})$ *where* $(x^\star,y^\star)$ *is the optimizer of* (6.6)*.*

Our solver follows recent developments in solving unregularized box-simplex games. We analyze an approximate extragradient algorithm based on the mirror prox method of [415], and prove that iterates of the regularized problem (6.6) enjoy multiplicative stability properties previously shown for the iterates of mirror prox on the unregularized problem [147]. Leveraging these tools, we also show the regularizer-operator pair satisfies technical conditions known as *relative Lipschitzness* and *strong monotonicity*, thus enabling a similar convergence analysis as in [147]. This yields an efficient algorithm for solving (6.6).

Roughly, when the scale of the problem (defined in terms of the matrix operator norm $\|\mathbf{A}\|_\infty$ and appropriate norms of b and c) is bounded,[3] our algorithm for computing a high-precision optimizer to (6.6) runs in $\widetilde{O}(\frac{1}{\sqrt{\mu\epsilon}})$ iterations, each bottlenecked by a matrix-vector product through $\mathbf{A}$. When $\mu \approx \epsilon$, the optimizer of the regularized variant is an $O(\epsilon)$-approximate solution to the unregularized problem (6.1), and hence Theorem 41 recovers state-of-the-art runtimes (scaling as $\widetilde{O}(\epsilon^{-1})$) for box-simplex games up to logarithmic factors. We achieve our improved dependence on μ in Theorem 41 by trading off the scales of the primal and dual domains. This type of argument is well-known for *separable regularizers* [111], but a key technical novelty of our paper is demonstrating a similar analysis holds for non-separable regularizers compatible with box-simplex games e.g. the one from [293], which has not previously been done. To our knowledge, Theorem 41 is the first result for solving general regularized box-simplex games to high accuracy in nearly-linear time. We develop our box-simplex algorithm and prove Theorem 41 in Section 6.4.

Improved rates for the Sinkhorn distance objective. We apply our accelerated solver for (6.6) in computing approximations to the Sinkhorn distance objective (6.3), a fundamental algorithmic problem in the practice of machine learning, at a faster rate. It is well-known that solving the *regularized* Sinkhorn problem (6.3) with μ scaling much larger than the target accuracy ϵ enjoys favorable properties in practice [151] (compared to its unregularized counterpart, the standard OT distance). In [24], the authors show that Sinkhorn iteration studied in prior work solves (6.3) to additive accuracy ϵ at an unaccelerated rate of $\widetilde{O}(\frac{1}{\mu\epsilon})$. For completeness we provide a proof of this result (up to logarithmic factors) in Appendix E.3.3.

As a straightforward application of the solver we develop for (6.6), we demonstrate that we can attain an accelerated rate of $\widetilde{O}(\frac{1}{\sqrt{\mu\epsilon}})$ for approximating (6.3) to additive accuracy ϵ via a first-order method. More specifically, the following result is based on reducing the "explicitly constrained" Sinkhorn objective (6.3) to a "soft constrained" regression variant of the form (6.4), where our box-simplex game solver is applicable. We now state our first result on improved rates for approximating Sinkhorn distance objectives.

[3] Our runtimes straightforwardly extend to depend appropriately on these norms in a scale-invariant way.

Theorem 34 (informal, see Theorem 85). *Let $\mu \in [\Omega(\epsilon), O(\frac{\|c\|_\infty}{\log m})]$ in (6.3) corresponding to a complete bipartite graph with m edges. There is an algorithm based on the regularized box-simplex game solver of Theorem 41 which obtains an ϵ-approximate minimizer to (6.3) in time $\widetilde{O}(m \cdot \frac{\|c\|_\infty}{\sqrt{\mu\epsilon}})$.*

By leveraging the particular structure of the Sinkhorn distance and its connection to a primitive in scientific computing and theoretical computer science known as *matrix scaling* [367, 145, 552], we give a further-improved solver for (6.3) in Theorem 86. This solver has a nearly-linear runtime scaling as $\widetilde{O}(\frac{1}{\mu})$, which is a high-precision solver for the original Sinkhorn objective. Our high-precision Sinkhorn solver applies powerful second-order optimization tools from [145] based on the *box-constrained Newton's method* for matrix scaling, yielding our second result on improved Sinkhorn distance approximation rates.

Theorem 35 (informal, see Theorem 86). *Let $\mu, \epsilon = O(\|c\|_\infty)$ in (6.3) corresponding to a complete bipartite graph with m edges. There is an algorithm based on the matrix scaling solver of [145] which obtains an ϵ-approximate minimizer to (6.3) in time $\widetilde{O}(m \cdot \frac{\|c\|_\infty}{\mu})$.*

We present both Theorems 85 and 86 because they follow from somewhat incomparable solver frameworks. While the runtime of Theorem 85 is dominated by that of Theorem 86, it is a direct application of a more general solver (Theorem 41), which also applies to regularized regression or box-simplex objectives where the optimum does not have a characterization as a matrix scaling. Moreover, the algorithm of Theorem 86 is a second-order method which leverages recent advances in solving Laplacian systems, and hence may be less practical than its counterpart in Theorem 85. Finally, we note that due to subtle parameterization differences for our DDBM applications, the DDBM runtime attained by using our box-simplex solver within our reduction framework is more favorable on sparse graphs ($m \ll n^2$), compared to that obtained by the matrix scaling solver.

Since our results on optimizing Sinkhorn distances follow from machinery developed in this paper and [145], we defer them to Appendix E.3. For consistency with the optimal transport literature and ease of presentation, we state our results in Appendix E.3 for instances of (6.3) corresponding to complete bipartite graphs. However, our algorithms based on Theorem 41 extend naturally to sparse graphs, as do the matrix scaling algorithms of [145] as discussed in that work. We provide applications of these subroutines to the DDBM problem on sparse graphs in Section 6.3 and Appendix E.1.

A recent advancement. Subsequent to the original submission of this paper, a breakthrough result of [124] provided an algorithm which computes high-accuracy solutions to a variety of graph-based optimization objectives in almost-linear time. One of the many applications of [124] is faster algorithms for computing Sinkhorn distances. Using the algorithm designed in [124], in place of [145], yields a speedup of Theorem 86, removing the polynomial dependence on μ^{-1} at the cost of an overhead of $m^{o(1)}$.

Theorem 36 (informal, see Lemma 271). *For $\epsilon = \Omega(m^{-3})$ in (6.3) corresponding to a bipartite*

graph with m edges, there is an algorithm that obtains an ϵ-approximate minimizer to (6.3) *in time* $m^{1+o(1)}$.

Further, we show how to use [124] to obtain improved running times for DDBM. By plugging in this Sinkhorn distance computation algorithm into our DDBM framework (and showing it is compatible with the form of our subproblems), we obtain an improved (randomized) DDBM solver in terms of the ϵ^{-1} dependence at the cost of a $m^{o(1)}$ overhead.

Theorem 37 (informal, see Theorem 40)**.** *Let $G = (V, E)$ be bipartite, $|V| = n$, $|E| = m$, and $\epsilon \in [\Omega(m^{-3}, 1)$. There is a randomized algorithm with success probability $1 - n^{-\Omega(1)}$ maintaining an ϵ-approximate maximum matching in an (adaptive) decremental stream running in time $m^{1+o(1)}\epsilon^{-2}$.*

We believe this result highlights the versatility and utility of our framework for solving DDBM. Given the varied technical machinery involved (and differing runtimes and use of randomness) our Sinkhorn algorithms, i.e. our new regularized box-simplex solver (see Section 6.4), [145], and [124], we provide statements of each solver instantiated by these different machinery in Section 6.4.3. However, due to the subsequent nature of [124] with respect to our work, and because DDBM running times using [145] are no better than those achieved in our paper using our regularized box-simplex solver, the proofs of our DDBM solver based on [124] are deferred to Appendix E.1.4.

6.1.2 Prior work

Dynamic matching. Dynamic graph algorithms are an active area of research in the theoretical computer science, see e.g. [264, 204, 73, 320, 230, 247, 5, 262, 245, 248, 408, 246] and references therein. These algorithms have been developed under various dynamic graph models, including the *oblivious adversary* model where the updates to the graph are fixed in advance (i.e. do not depend on randomness used by the algorithm), and the *adaptive adversary* model in which updates are allowed to respond to the algorithm, potentially adversarially. We focus on surveying deterministic dynamic matching algorithms, which perform equally well under oblivious and adaptive updates; for a more in-depth discussion and corresponding developments in other settings, see [527].

Many variants of the particular dynamic problem of maintaining matchings in bipartite graphs have been studied, such as the *fully dynamic* [259], *incremental* [258, 250], and *decremental* [73] cases. However, known conditional hardness results [270, 328] suggest that attaining a polylogarithmic update time for maintaining a ϵ-approximate fully dynamic matching may be unattainable even for constant ϵ, prompting the study of restricted variants. The works most relevant to our paper are those of [258], which provides a $\widetilde{O}(\epsilon^{-4})$ amortized update time algorithm for computing an ϵ-approximate matching in incremental bipartite case, and [73], which achieves a similar $\widetilde{O}(\epsilon^{-4})$ update time for decremental bipartite matching. Our main DDBM results, stated in Theorems 38 and 39, improve upon [73] by roughly a factor of ϵ^{-1} in the decremental setting.

Box-simplex games. Box-simplex games, as well as ℓ_1 and ℓ_∞ regression, are equivalent to linear programs in full generality [349], have widespread utility, and hence have been studied extensively by the continuous optimization community. Here we focus on discussing *near-linear time* approximation algorithms, i.e. algorithms which run in time near-linear in the sparsity of the constraint matrix, potentially depending inverse polynomially on the desired accuracy. Interior point methods solve these problems with polylogarithmic dependence on accuracy, but are second-order and often encounter polynomial runtime overhead in the dimension (though there are exceptions, e.g. [516] and references therein).

A sequence of early works e.g. [415, 419, 420] on primal-dual optimization developed first-order methods for solving games of the form (6.1). These works either directly operated on the objective (6.1) as a minimax problem, or optimized a smooth approximation to the objective recast as a convex optimization problem. While these techniques obtained iteration complexities near-linear in the sparsity of the constraint matrix $\mathbf{A}$, they either incurred an (unaccelerated) ϵ^{-2} dependence on the accuracy ϵ, or achieved an ϵ^{-1} rate of convergence at the cost of additional dimension-dependent factors. This was due to the notorious "ℓ_∞ strong convexity barrier" (see Appendix A, [486]), which bottlenecked classical acceleration analyses over an ℓ_∞-constrained domain. [483] overcame this barrier by utilizing the primal-dual structure of (6.1) through a technique called "area convexity", obtaining a $\widetilde{O}(\epsilon^{-1})$-iteration algorithm. Since then, [147] demonstrated that fine-grained analyses of the classical algorithms of [415, 420] also obtain comparable rates for solving (6.1). Finally, we mention that area convexity has found applications in optimal transport and positive linear programming [293, 91].

Sinkhorn distances. Since [151] proposed Sinkhorn distances for machine learning applications, a flurry of work has aimed at developing algorithms with faster runtimes for (6.3). A line of work by [25, 208, 365] has analyzed the theoretical guarantees of the classical Sinkhorn matrix scaling algorithm for this problem, due to the characterization of its solution as a diagonal rescaling of a fixed matrix. These algorithms obtain rates scaling roughly as $\widetilde{O}(\|c\|_\infty^2 \epsilon^{-2})$ for solving (6.3) to additive accuracy ϵ. Perhaps surprisingly, to our knowledge no guarantees for solving (6.3) which improve as the regularization parameter μ grows are currently stated in the literature, a shortcoming addressed by this work. Finally, we remark that Sinkhorn iteration has also received extensive treatment from the theoretical computer science community, e.g. [367], due to connections with algebraic complexity; see [233] for a recent overview of these connections.

6.2 Preliminaries

General notation. We denote $[n] := \{1, 2, \ldots, n\}$ and let $\mathbf{0}$ and $\mathbf{1}$ denote the all-0 and all-1 vectors. Given $v \in \mathbb{R}^d$, v_i or $[v]_i$ denotes the i^{th} entry of v, and for any subset $E \subseteq [d]$ we use v_E or $[v]_E$ to denote the vector in $\mathbb{R}^d$ zeroing out v on entries outside of E. We use $([v]_i)_+ = \max([v]_i, 0)$ to

denote the operation truncating negative entries. We use $v \circ w$ to denote elementwise multiplication between any $v, w \in \mathbb{R}^d$. Given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, we use $\mathbf{A}_{ij}$ to denote its $(i,j)^{\text{th}}$ entry, and denote its i^{th} row and j^{th} column by $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ respectively; its nonzero entry count is $\text{nnz}(\mathbf{A})$. We use $\mathbf{diag}\,(v)$ to denote the diagonal matrix where $[\mathbf{diag}\,(v)]_{ii} = v_i$, for each i. Given two quantities M and M', for any $c > 1$ we say M is a c-approximation to M' if it satisfies $\frac{1}{c}M' \leq M \leq cM'$. For $\epsilon \ll 1$, we say M is an ϵ(-multiplicative)-approximation of M' if $(1-\epsilon)M' \leq M \leq (1+\epsilon)M'$. Throughout the paper, we use $|\mathbf{A}|$ to denote taking the elementwise absolute value of a matrix $\mathbf{A}$, and v^2 to denote the elementwise squaring of a vector v when clear from context.

Norms. $\|\cdot\|_p$ denotes the ℓ_p norm of a vector or corresponding operator norm of a matrix. In particular, $\|\mathbf{A}\|_\infty = \max_i \|\mathbf{A}_{i:}\|_1$. We use $\|\cdot\|$ interchangeably with $\|\cdot\|_2$. We use Δ^m to denote an m-dimensional simplex, i.e. $x \in \Delta^m \iff x \in \mathbb{R}^d_{\geq 0}, \|x\|_1 = 1$.

Graphs. A graph $G = (V, E)$ has vertices V and edges E; we abbreviate $n := |V|$ and $m := |E|$ whenever the graph is clear from context. For bipartite graphs, $V = L \cup R$ denotes the bipartition. We let $\mathbf{B} \in \{0,1\}^{E \times V}$ be the (unsigned edge-vertex) incidence matrix with $\mathbf{B}_{ev} = 1$ if v is an endpoint of e and $\mathbf{B}_{ev} = 0$ otherwise.

Bregman divergence. Given any convex distance generating function (DGF) $q(x)$, we use $V^q_{x'}(x) = q(x) - q(x') - \langle \nabla q(x'), x - x' \rangle \geq 0$ as its induced Bregman divergence. When the DGF is clear from context, we abbreviate $V := V^q$. By definition, V satisfies

$$\langle -\nabla V_{x'}(x), x - u \rangle = V_{x'}(u) - V_x(u) - V_{x'}(x) \text{ for any } x, x', u. \tag{6.7}$$

Computational model. We use the standard word RAM model, where one can perform each basic arithmetic operations on $O(\log n)$-bit words in constant time.

6.3 Dynamic decremental bipartite matching

Here we provide a reduction from maintaining an approximately maximum matching in a decremental bipartite graph to solving regularized matching problems to sufficiently high precision. In Section 6.3.1 we give this framework and then, in Section 6.3.2, we provide various instantiations of our framework based on different solvers, to demonstrate its versatility.

6.3.1 DDBM framework

Here we provide our general framework for solving DDBM, which assumes that for bipartite $G = (V, E)$ and approximate matching value M there is a canonical regularized matching problem with properties given in Definition 23; we later provide multiple such examples. Throughout this section, $\text{MCM}(E)$ denotes the size of the maximum cardinality matching on edge set E; the vertex set V is fixed throughout, so we omit it in definitions.

Definition 23 (Canonical regularized objective)**.** *Let $G = (V, E_0)$ be a bipartite graph and $M \geq 0$ be an 8-approximation of $\mathrm{MCM}(E_0)$. For all $E \subseteq E_0$ with $\mathrm{MCM}(E) \geq \frac{M}{8}$, let $f_{M,E} : \mathbb{R}^E_{\geq 0} \to \mathbb{R}$,*

$$\nu^E := \min_{x \in \mathbb{R}^E_{\geq 0}} f_{M,E}(x) \text{ , and } x^E := \mathrm{argmin}_{x \in \mathbb{R}^E_{\geq 0}} f_{M,E}(x). \tag{6.8}$$

*We call the set of $\{f_{M,E}\}_{\mathrm{MCM}(E) \geq \frac{M}{8}}$ a family of (ϵ, β)-*canonical regularized objectives *(CROs) for $G(E_0)$ and M if the following four properties hold.*

1. *For all $E \subseteq E_0$ with $\mathrm{MCM}(E) \geq \frac{M}{8}$, $-\nu^E$ is an $\frac{\epsilon}{8}$-approximation of $\mathrm{MCM}(E)$.*
2. *For all $E \subseteq E_0$ with $\mathrm{MCM}(E) \geq \frac{M}{8}$, $f_{M,E}$ is equivalent to f_{M,E_0} with the extra constraint that $x_{E_0 \setminus E}$ is fixed to 0 entrywise.*
3. *For any $E' \subseteq E \subseteq E_0$ with $\mathrm{MCM}(E) \geq \frac{M}{8}$ and $\mathrm{MCM}(E') \geq \frac{M}{8}$,*

$$f_{M,E'}(x^{E'}) - f_{M,E}(x^E) \geq \beta V^H_{x^E}(x^{E'}) \text{ where } H(x) := \sum_e x_e \log x_e \tag{6.9}$$

4. *For any $x \in \mathbb{R}^E_{\geq 0}$ such that $8Mx$ is a feasible matching on (V, E),*

$$8M \|x_E\|_1 - \frac{\epsilon}{128} M \leq -f_{M,E}(x) \leq 8M \|x_E\|_1 + \frac{\epsilon}{128} M. \tag{6.10}$$

We further define the following notion of a canonical solver for a given CRO, which solves the CRO to sufficiently high accuracy, and rounds the approximate solution to feasibility.

Definition 24 (Canonical solver)**.** *For (ϵ, β)-CROs $\{f_{M,E}\}_{\mathrm{MCM}(E) \geq \frac{M}{8}}$, we call $\mathcal{A}$ an $(\epsilon, \mathcal{T})$-*canonical solver *if it has subroutines* Solve *and* Round *running in $O(\mathcal{T})$ time, satisfying:*

1. Solve *finds an approximate solution $\hat{x}^E$ of $f_{M,E}$ satisfying*

$$\left(1 + \frac{\epsilon}{8}\right) \nu^E \leq f_{M,E}(\hat{x}^E) \leq \left(1 - \frac{\epsilon}{8}\right) \nu^E. \tag{6.11a}$$

$$\left\|\hat{x}^E - x^E\right\|_1 \leq \frac{\epsilon}{1100}. \tag{6.11b}$$

2. Round *takes $\hat{x}^E$ and returns $\tilde{x}^E$ where $8M\tilde{x}^E$ is a feasible matching for $G(E)$, and:*

$$\left(1 + \frac{\epsilon}{8}\right) \nu^E \leq f_{M,E}(\tilde{x}^E) \leq \left(1 - \frac{\epsilon}{8}\right) \nu^E. \tag{6.12a}$$

$$\tilde{x}^E \leq \hat{x}^E \text{ monotonically.} \tag{6.12b}$$

Our DDBM framework, Algorithm 20, uses CRO solvers satisfying the approximation guarantees of Definition 24 to dynamically maintain an approximately maximum matching. We state its correctness and runtime in Proposition 18, and defer a proof to Appendix E.1.1.

Algorithm 20: DecMatching$(\epsilon, G = (V, E))$

Input: $\epsilon \in (0, \frac{1}{8})$, graph $G = (V, E)$;
Parameters: Family of CROs $\{f_{M,E}\}_{E \text{ is MP}}$, an $(\epsilon, \mathcal{T})$-canonical solver $(\mathsf{Solve}, \mathsf{Round})$;
Compute M with $\frac{1}{2}\text{MCM}(E) \le M \le \text{MCM}(E)$, via the greedy algorithm;
$\hat{x}^E \leftarrow \mathsf{Solve}(f_{M,E})$;
$\tilde{x}^E \leftarrow \mathsf{Round}(\hat{x}^E)$;
$M_{\text{est}} \leftarrow M$;
while $M_{\text{est}} > \frac{1}{4}M$ **do**
  $E_{\text{del}} \leftarrow \emptyset$;
  while *edge e is deleted and* $\|\tilde{x}^E_{E_{\text{del}}}\|_1 \le \frac{\epsilon}{8}\left\|\tilde{x}^E\right\|_1$ **do**
    ; ▷ recompute whenever the deleted approximate matching size reaches a factor $\Theta(\epsilon)$
    $E_{\text{del}} \leftarrow E_{\text{del}} \cup \{e\}$;
  $E \leftarrow E \setminus E_{\text{del}}$, $E_{\text{del}} \leftarrow \emptyset$;
  $\hat{x}^E \leftarrow \mathsf{Solve}(f_{M,E})$; ▷ find high-accuracy minimizer of $F_{M,E}$ satisfying (6.11a) and (6.11b)
  $\tilde{x}^E \leftarrow \mathsf{Round}(\hat{x}^E)$; ▷ round to feasible matching satisfying (6.12a) and (6.12b)
  Compute M_{est} with $\frac{1}{2}\text{MCM}(E) \le M_{\text{est}} \le \text{MCM}(E)$, via the greedy algorithm

In the following, we let E_0 be the original graph's edge set, and $E_1, E_2, \ldots, E_K$ be the sequence of edge sets recomputed in Line 11, before termination for E_{K+1} on Line 7.

Proposition 18. *Let $\epsilon \in (0, 1)$ and $M \ge 0$. Given a family of (ϵ, β)-CROs $\{f_{M,E}\}$ for $G = (V, E_0)$, and an $(\epsilon, \mathcal{T})$-canonical solver for the family, Algorithm 20 satisfies the following.*

1. *When $M_{\text{est}} > \frac{1}{4}M$ on Line 7, where M_{est} estimates $\text{MCM}(E_k)$: at any point in the loop of Lines 9 to 10, $8M\tilde{x}^{E_k}_{E_k \setminus E_{\text{del}}}$ is an ϵ-approximate matching of $G(V, E_k \setminus E_{\text{del}})$.*
2. *When $M_{\text{est}} \le \frac{1}{4}M$ on Line 7, where M_{est} estimates $\text{MCM}(E)$: $\text{MCM}(E) \le \frac{1}{2}\text{MCM}(E_0)$.*

The runtime of the algorithm is $O(m + (\mathcal{T} + m) \cdot \frac{M}{\beta\epsilon})$.

Proof sketch. We summarize proofs of the two properties, and our overall runtime bound.

Matching approximation properties. By the greedy matching guarantee in Line 7, it holds that for any E_k (the edge set recomputed in the k^{th} iteration of Line 11 before termination), its true matching size $\text{MCM}(E_k)$ must be no smaller than $\frac{M}{4}$. Consequently, we can use the CRO family to approximate the true matching size up to $O(\epsilon)$ multiplicative factors, and by the guarantee (6.12a), this implies $8M\tilde{x}^{E_k}_{E_k \setminus E_{\text{del}}}$ is an $O(\epsilon)$ approximation of the true matching size. Also, our algorithm's termination condition and the guarantee on M_{est} immediately imply $\text{MCM}(E_{K+1}) \le \frac{1}{2}\text{MCM}(E_0)$.

Iteration bound. We use a potential argument. Given $E_{k+1} \subset E_k$, corresponding to consecutive

edge sets requiring recomputation, we use the following inequalities:

$$\begin{aligned} f_{M,E_{k+1}}\left(x^{E_{k+1}}\right) - f_{M,E_k}\left(x^{E_k}\right) &\overset{(i)}{\geq} \beta V^H_{x^{E_k}}\left(x^{E_{k+1}}\right) \\ &\overset{(ii)}{\geq} \beta \sum_{i\in E_{\text{del}}} \left([x^{E_{k+1}}]_i \log[x^{E_{k+1}}]_i - [x^{E_k}]_i \log[x^{E_k}]_i - \left(1+\log[x^{E_k}]_i\right)\cdot\left([x^{E_{k+1}}]_i - [x^{E_k}]_i\right)\right) \\ &\overset{(iii)}{=} \beta \sum_{i\in E_{\text{del}}} [x^{E_k}]_i \overset{(iv)}{\geq} \beta\left(\left\|\hat{x}^{E_k}_{E_{\text{del}}}\right\|_1 - \left\|\hat{x}^{E_k} - x^{E_k}\right\|_1\right) \overset{(v)}{\geq} \beta\left(\left\|\tilde{x}^{E_k}_{E_{\text{del}}}\right\|_1 - \left\|\hat{x}^{E_k} - x^{E_k}\right\|_1\right), \end{aligned} \tag{6.13}$$

where (i) uses the third property in (6.9), (ii) uses convexity of the scalar function $c\log c$, (iii) uses that $x^{E_{k+1}}_{E_{\text{del}}}$ is 0 entrywise, (iv) uses the triangle inequality, and (v) uses the monotonicity property (6.12b). Moreover, between recomputations we have that the ℓ_1-norm of deleted edges satisfies $\left\|\tilde{x}^{E_k}_{E_{\text{del}}}\right\|_1 = \Omega(\epsilon)$, and our solver guarantees $\left\|\hat{x}^{E_k} - x^{E_k}\right\|_1 = O(\epsilon)$. Since the overall function decrease before termination is $O(M)$ given the stopping criterion in Line 7, the algorithm terminates after $O(\frac{M}{\beta\epsilon})$ recomputations. □

Using this generic DDBM framework, we obtain improved decremental matching algorithms by defining families of CROs $f_{M,E(G)}$ with associated $(\epsilon,\mathcal{T})$-canonical solvers satisfying Definition 24. In Appendix E.1.2, we give a regularized primal-dual construction of $f_{M,E}$, and adapt the solver of Section 6.4 to develop a canonical solver for the family (specifically, as the subroutine Solve). Similarly, in Appendix E.1.3, we show how to construct an appropriate family of $f_{M,E}$ using Sinkhorn distances, and apply the matrix scaling method presented in Appendix E.3.2 (based on work of [145]) to appropriately instantiate Solve.

While both algorithms, as stated, only maintain an approximate fractional matchings, this fractional matching can be rounded at any point to an explicit integral matching via e.g. the cycle-canceling procedure of Proposition 3 in [39] in time $O(m\log m)$, or dynamically (albeit at amortized cost $\widetilde{O}(\epsilon^{-4})$ using [527]). Moreover, our algorithm based on the regularized box-simplex solver (Theorem 38) is deterministic, and both work against an adaptive adversary. Repeatedly applying Proposition 18, we obtain the following overall claim.

Corollary 17. *Let $G = (V, E(G))$ be bipartite, and suppose for any subgraph $(V, E_0 \subseteq E(G))$, we are given a family of (ϵ,β)-CROs and an $(\epsilon,\mathcal{T})$-canonical solver for the family. There is a deterministic algorithm maintaining a fractional ϵ-approximate matching in a dynamic bipartite graph with adversarial edge deletions, running in time $O\left(m\log^3 n + (\mathcal{T}+m)\cdot\frac{M}{\beta\epsilon}\cdot\log n\right)$.*

Proof. It suffices to repeatedly apply Proposition 18 until we can safely conclude $\text{MCM}(E) = 0$, which by the second property can only happen $O(\log n)$ times. □

6.3.2 DDBM solvers

In this section, we demonstrate the versatility of the DDBM framework in Section 6.3.1 by instantiating it with different classes of CRO families, and applying different canonical solvers on these families. By using regularized box-simplex game solvers developed in this paper (see Section 6.4), we give an $\widetilde{O}(m\epsilon^{-3})$ time algorithm for maintaining a ϵ-multiplicatively approximate fractional maximum matching in a m-edge bipartite graph undergoing a sequence of edge deletions, improving upon the previous best running time of $\widetilde{O}(m\epsilon^{-4})$ [73]. We also use our framework to obtain different decremental matching algorithms with runtime $\widetilde{O}(n^2\epsilon^{-3})$ and $O(m^{1+o(1)}\epsilon^{-2})$, building on recent algorithmic developments in the literature on matrix scaling. The former method uses box-constrained Newton's method solvers for matrix scaling problems in [145] (these ideas are also used in Appendix E.3.2), and the latter uses a recent almost-linear time high-accuracy Sinkhorn-objective solver in [124], a byproduct of their breakthrough maximum flow solver. We defer readers to corresponding sections in Appendix E.1 for omitted proofs.

Given a bipartite graph initialized at $G = (V, E_0)$ with unsigned incidence matrix $\mathbf{B} \in \{0,1\}^{E\times V}$; we denote $n := |V|$ and $m := |E_0|$. The first family of CROs one can consider is the regularized box-simplex game objective in form:

$$\begin{aligned}\min_{(x,\xi)\in\Delta^{E+1}} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y) &:= -\mathbf{1}_E^\top(8Mx) - y^\top\left(8M\mathbf{B}^\top x - \mathbf{1}\right) + \gamma^{\mathsf{x}} H(x,\xi) + \gamma^{\mathsf{y}}\left(y^2\right)^\top \mathbf{B}^\top x,\\ \text{where } \gamma^{\mathsf{x}} = \widetilde{\Theta}\left(\epsilon M\right),\ \gamma^{\mathsf{y}} &= \Theta\left(\epsilon M\right),\ \text{ and}\\ f_{M,E}(x) &:= \min_{\xi|(x,\xi)\in\Delta^{E+1}} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y)\end{aligned} \tag{6.14}$$

We prove this is a family of $(\epsilon, \gamma^{\mathsf{x}})$-CROs (Lemma 267). The canonical solver for this family uses RemoveOverflow (Algorithm 4, [39]) as Round and uses the regularized box-simplex games developed later in this paper (see Section 6.4) as Solve, which finds an ϵ-approximate solution of (6.14) in time $\widetilde{O}(\frac{m}{\epsilon})$. Combining all these components with the DDBM framework in Corollary 17 leads to the following DDBM solver based on regularized box-simplex games.

Theorem 38. *Let $G = (V, E)$ be bipartite and let $\epsilon \in [\Omega(m^{-3}), 1)$. There is a deterministic algorithm for the DDBM problem which maintains an ϵ-approximate matching, based on solving regularized box-simplex games, running in time $O\left(m\epsilon^{-3}\log^5 n\right)$.*

Our second CRO family is the following regularized Sinkhorn distance objective:

$$\begin{aligned}\min_{\binom{x}{x^{\mathsf{dum}}}\in\mathbb{R}^{\widetilde{E}}_{\geq 0}\ |\ 2|R|\widetilde{\mathbf{B}}^\top\binom{x}{x^{\mathsf{dum}}}=d} f^{\mathsf{sink}}_{M,E}(x^{\mathsf{tot}}) &:= 2|R|\mathbf{1}_E^\top x + \gamma H\left(x, x^{\mathsf{dum}}\right) \text{ where } \gamma = \widetilde{\Theta}\left(\epsilon M\right),\\ f^{\mathsf{sink}}_{M,E}(x) &:= \min_{x^{\mathsf{dum}}\in\mathbb{R}^{E\setminus E_0}_{\geq 0}} f^{\mathsf{sink}}_{M,E}(x, x^{\mathsf{dum}}),\end{aligned} \tag{6.15}$$

where we extend graph $G = (V, E)$ to a balanced bipartite graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ by introducing dummy

vertices and edges. The extended graph allows us to write the inequality constraint $\mathbf{B}^\top x = \mathbf{1}_V$ equivalently as the linear constraint $2|R|\widetilde{\mathbf{B}}^\top \begin{pmatrix} x \\ x^{\mathsf{dum}} \end{pmatrix} = d$ for some defined $d \in \mathbb{R}^{\widetilde{V}}$ as some properly-extended vector of $\mathbf{1}_V$. This allows us to apply known matrix scaling solver to such approximating Sinkhorn distance objective in literature.

We prove this is a family of (ϵ, γ)-CROs (Lemma 269). The canonical solver for this family uses truncation to E as Round and uses the matrix scaling solver from [145] based on box-constrained Newton's method as Solve, which finds an ϵ-approximate solution of (6.15) in time $\widetilde{O}(n^2/\epsilon)$. Combining all these components with the DDBM framework in Corollary 17 leads to the following DDBM solver based on approximating Sinkhorn distances.

Theorem 39. *Let $G = (V, E)$ be bipartite and $\epsilon \in [\Omega(m^{-3}), 1)$. There is a randomized algorithm for the DDBM problem which maintains an ϵ-approximate matching with probability $1 - n^{-\Omega(1)}$, based on matrix scaling solver of [145], running in time $\widetilde{O}\left(n^2\epsilon^{-3}\right)$.*

Alternatively, for the same (ϵ, γ)-CRO family as in (6.15), one can use the same Round procedure and the recent high-accuracy almost-linear time graph flow problems solver of [124] for Solve as a canonical solver. Since this new solver can find high-accuracy solutions of entropic-regularized problems of the form (6.15) within a runtime of $(|E_0| + O(|V|))^{1+o(1)} = m^{1+o(1)}$, this gives a third DDBM solver, which yields and improved an dependence on ϵ^{-1}.

Theorem 40. *Let $G = (V, E)$ be bipartite and $\epsilon \in [\Omega(m^{-3}), 1)$. There is a randomized algorithm for the DDBM problem which maintains an ϵ-approximate matching with probability $1 - n^{-\Omega(1)}$, based on the Sinkhorn objective solver of [124], running in time $m^{1+o(1)}\epsilon^{-2}$.*

6.4 Regularized box-simplex games

In this section, we develop a high-accuracy solver for regularized box-simplex games:

$$\begin{aligned} \min_{x \in \Delta^m} \max_{y \in [0,1]^n} f_{\mu,\epsilon}(x, y) &:= y^\top \mathbf{A}^\top x + c^\top x - b^\top y + \mu H(x) - \frac{\epsilon}{2}\left(y^2\right)^\top |\mathbf{A}|^\top x, \\ \text{where } \; H(x) &:= \sum_{i \in [m]} x_i \log x_i \text{ is the standard entropic regularizer,} \end{aligned} \tag{6.16}$$

where we recall absolute values and squaring act entrywise.

For ease of presentation, we make the following assumptions for some $\delta > 0$.

1. Upper bounds on entries: $\|\mathbf{A}\|_\infty \le 1$, $\|b\|_\infty \le B_{\max}$, $\|c\|_\infty \le C_{\max}$. For simplicity, we assume $B_{\max} \ge C_{\max} \ge 1$; else, $C_{\max} \leftarrow \max(1, C_{\max})$ and $B_{\max} \leftarrow \max(C_{\max}, B_{\max})$.
2. Lower bounds on matrix column entries: $\max_i |\mathbf{A}_{ij}| \ge \delta$ for every $j \in [n]$.

We defer the detailed arguments of why these assumptions are without loss of generality to Appendix E.2. Our algorithm acts on the induced (monotone) gradient operator of the regularized box-simplex objective (6.16), namely $(\nabla_x f_{\mu,\epsilon}(x,y), -\nabla_y f_{\mu,\epsilon}(x,y))$, defined as

$$g_{\mu,\epsilon}(x,y) := \left(\mathbf{A}y + c + \mu(\mathbf{1} + \log(x)) - \frac{\epsilon}{2}\left|\mathbf{A}\right|(y^2), -\mathbf{A}^\top x + b + \epsilon \cdot \mathbf{diag}\,(y)\left|\mathbf{A}\right|^\top x\right). \tag{6.17}$$

Further, it uses the following joint (non-separable) regularizer of

$$r_{\mu,\epsilon}(x,y) := \rho \sum_{i\in[m]} x_i \log x_i + \frac{1}{\rho} x^\top \left|\mathbf{A}\right|\left(y^2\right) \quad \text{where} \quad \rho = \sqrt{\frac{2\mu}{\epsilon}}, \tag{6.18}$$

variants of which have been used in [483, 293, 39, 147]. When clear from context, we drop subscripts and refer to these as operator g and regularizer r. Our method is the first high-accuracy near-linear time solver for the regularized problem (6.16), yielding an $O(\varepsilon)$-approximate solution with a runtime scaling polylogarithmically in problem parameters and ε. We utilize a variant of the mirror prox (extragradient) [415] method for strongly monotone objectives, which appeared in [111, 147] for regularized saddle point problems with separable regularizers.

In Section 6.4.1, we present high-level ideas of our algorithm, which uses a mirror prox outer loop (Algorithm 21) and an alternating minimization inner loop (Algorithm 22) to implement outer loop steps; we also provide convergence guarantees. In Section 6.4.2, we state useful properties of the regularizer (6.18), and discuss a technical detail ensuring iterate stability in our method. In Section 6.4.3, we provide our full algorithm for regularized box-simplex games, Algorithm 23 and give guarantees in Theorem 41. Omitted proofs are in Appendix E.2.

6.4.1 Algorithmic framework

In this section, we give the algorithmic framework we use to develop our high-precision solver, which combines an outer loop inspired by mirror prox [415] with a custom inner loop for implementing each iteration. We first define an approximate solution for a proximal oracle.

Definition 25 (Approximate proximal oracle solution)**.** *Given a convex function f over domain $\mathcal{Z}$ and $\varepsilon \geq 0$, we say $z' \in \mathcal{Z}$ is a* ε-approximate solution *for a proximal oracle if z' satisfies $\langle \nabla f(z'), z' - z\rangle \leq \varepsilon$. We denote this approximation property by $z' \leftarrow_\varepsilon \operatorname{argmin}_{z\in\mathcal{Z}} f$.*

We employ such approximate solutions as the proximal oracle within our "outer loop" method. Our outer loop is a variant of mirror prox (Algorithm 21) which builds upon both the mirror prox type method in [483] for solving unregularized box-simplex games and the high-accuracy mirror prox solver developed in [111, 147] for bilinear saddle-point problems on geometries admitting separable regularizers. We first give a high-level overview of the analysis, which requires bounds on two

properties. First, suppose g is ν-strongly monotone with respect to regularizer r, i.e.

$$\text{for any } w, z \in \mathcal{Z}, \ \langle g(w) - g(z), w - z \rangle \geq \nu \langle \nabla r(w) - \nabla r(z), w - z \rangle . \tag{6.19}$$

Further, suppose it is α-*relatively Lipschitz* with respect to r and Algorithm 23 (see Definition 1 of [147]), i.e. for any consecutive iterates z_{k-1}, $z_{k-1/2}$, z_k of our algorithm,[4]

$$\left\langle g(z_{k-1/2}) - g(z_{k-1})), z_{k-1/2} - z_k \right\rangle \leq \alpha \Big(V_{z_{k-1/2}}(z_k) + V_{z_{k-1}}(z_{k-1/2}) \Big). \tag{6.20}$$

With these assumptions, we show that the strongly monotone mirror prox step makes progress by decreasing the divergence to optimal solution $V_{z_k}(z^\star)$ by a factor of $\frac{\alpha}{\alpha+\nu}$: this implies $\widetilde{O}(\frac{\alpha}{\nu})$ iterations suffice for finding a high-accuracy solution. We provide the formal convergence guarantee of MirrorProx in Proposition 19, which also accommodates the error of each approximate proximal step used in the algorithm. This convergence guarantee is generic and does not rely on the concrete structure of g, r in our box-simplex problem.

Algorithm 21: MirrorProx()

Input: $\frac{\varepsilon}{2}$-approximate proximal oracle, operator and regularizer pair (g, r) such that g is ν-strongly monotone and α-relatively Lipschitz with respect to r;
Parameters: Number of iterations K;
Output: Point z_K with $V_{z_K}(z^\star) \leq (\frac{\alpha}{\nu+\alpha})^K \Theta + \frac{\varepsilon}{\nu}$;
$z_0 \leftarrow \operatorname{argmin}_{z \in \mathcal{Z}} r(z)$;
for $k = 1, \ldots, K$ **do**
  $z_{k-1/2} \leftarrow_{\varepsilon/2} \operatorname{argmin}_{z \in \mathcal{Z}} \left\{ \langle g\left(z_{k-1}\right), z \rangle + \alpha V_{z_{k-1}}(z) \right\}$;
  $z_k \leftarrow_{\varepsilon/2} \operatorname{argmin}_{z \in \mathcal{Z}} \left\{ \left\langle g\left(z_{k-1/2}\right), z \right\rangle + \alpha V_{z_{k-1}}(z) + \nu V_{z_{k-1/2}}(z) \right\}$;
return z_K

Proposition 19 (Convergence of Algorithm 21)**.** *Given regularizer r with range at most Θ, suppose g is ν-strongly-monotone with respect to r (see* (6.19)*), and is α-relatively-Lipschitz with respect to r (see* (6.20)*). Let z_K be the output of Algorithm 21. Then, $V^r_{z_K}(z^\star) \leq \left(\frac{\alpha}{\nu+\alpha}\right)^K \Theta + \frac{\varepsilon}{\nu}$.*

Given the somewhat complicated nature of our joint regularizer, we cannot solve the proximal problems required by Algorithm 21 in closed form. Instead, we implement each proximal step to the desired accuracy by using an alternating minimization scheme, similarly to the implementation of approximate proximal steps in [483, 293].

To analyze our algorithm, we use a generic progress guarantee for alternating minimization from [293] to solve each subproblem, stated below.

[4] This property (i.e. relative Lipschitzness restricted to iterates of the algorithm) was referred to as "local relative Lipschitzness" in [147], but we drop the term "local" for simplicity.

Algorithm 22: $\mathsf{AltMin}(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \mathbf{A}, \theta, T, x_{\text{init}}, y_{\text{init}})$

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}_{\geq 0}$, $\gamma^{\mathsf{x}} \in \mathbb{R}^m$, $\gamma^{\mathsf{y}} \in \mathbb{R}^n$, $T \in \mathbb{N}$, $\theta > 0$, $x_{\text{init}} \in \Delta^m$, $y_{\text{init}} \in [0,1]^n$;
Output: Approximate minimizer to $\langle (\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}), z \rangle + \theta r(z)$ for $r(z)$ in (6.18);
$x^{(0)} \leftarrow x_{\text{init}}$, $y^{(0)} \leftarrow y_{\text{init}}$;
for $0 \leq t \leq T$ **do**
  $x^{(t+1)} \leftarrow \text{argmin}_{x \in \Delta^m} \left\{ \langle \gamma^{\mathsf{x}}, x \rangle + \theta r(x, y^{(t)}) \right\}$;
  $y^{(t+1)} \leftarrow \text{argmin}_{y \in [0,1]^n} \left\{ \langle \gamma^{\mathsf{y}}, y \rangle + \theta r(x^{(t+1)}, y) \right\}$;
Return: $(x^{(T+1)}, y^{(T)})$;

Lemma 81 (Alternating minimization progress, Lemma 5 and Lemma 7, [293])**.** *Let $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be jointly convex, $\theta > 0$, and γ^x and γ^y be linear operators on $\mathcal{X}, \mathcal{Y}$. Define*

$$x_{\mathsf{OPT}}, y_{\mathsf{OPT}} = \text{argmin}_{x \in \mathcal{X}} \text{argmin}_{y \in \mathcal{Y}} f(x,y) := \langle \gamma^x, x \rangle + \langle \gamma^y, y \rangle + \theta r(x,y). \tag{6.21}$$

Suppose $f(x,y)$ is twice-differentiable and satisfies: for all $x' \geq \frac{1}{2}x$ entrywise, $x', x \in \mathcal{X}$ and $y', y \in \mathcal{Y}$, $\nabla^2 f(x', y') \succeq \frac{1}{\kappa} f(x,y)$. Then the iterates of Algorithm 22 satisfy

$$f(x^{(t+2)}, y^{(t+1)}) - f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}}) \leq \left(1 - \frac{1}{2\kappa}\right) \left(f(x^{(t+1)}, y^{(t)}) - f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}}) \right).$$

Combining this lemma with the structure of our regularizer (6.18), we obtain the following guarantees, showing Algorithm 22 finds a $\frac{\varepsilon}{2}$-approximate solution to the proximal oracle.

Corollary 18 (Convergence of Algorithm 22)**.** *Let $\delta, \varepsilon \in (0,1)$, $\rho \geq 1$. Suppose we are given $\gamma \in \mathcal{Z}_* = \mathcal{X}_* \times \mathcal{Y}_*$ with $\max(\|\gamma^{\mathsf{x}}\|_\infty, \|\gamma^{\mathsf{y}}\|_1) \leq B$, and define the proximal subproblem solution*

$$x_{\mathsf{OPT}}, y_{\mathsf{OPT}} = \text{argmin}_{x \in \Delta^m} \text{argmin}_{y \in [0,1]^n} f(x,y) := \langle \gamma^x, x \rangle + \langle \gamma^y, y \rangle + \theta r(x,y) \quad \text{for some } \theta > 0.$$

If the Hessian condition in Lemma 81 holds with a constant $\kappa > 0$, and all simplex iterates x of Algorithm 22 satisfy $x \geq \delta$ elementwise, then the algorithm finds a $\frac{\varepsilon}{2}$-approximate solution to the proximal oracle within $T = O\left(\log\left(\frac{\rho(B+mn\theta)^2}{\delta\varepsilon\theta}\right)\right)$ iterations.

6.4.2 Helper lemmas

Before providing our full method and analysis, here we list a few helper lemmas, which we rely on heavily in our later development. The first characterizes a useful property of $r_{\mu,\epsilon}$, showing that its Hessian is locally well-approximated by a diagonal matrix, which induces appropriate local norms for the blocks $x \in \mathcal{X}$, $y \in \mathcal{Y}$. We use this to prove the "strong monotonicity" (Lemma 85) and "relative Lipschitzness" (Lemma 86) bounds required in Section 6.4.3.

Lemma 82 (Bounds on regularizer). *Suppose* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *has* $\|\mathbf{A}\|_\infty \le 1$. *For any* $z = (x, y) \in \Delta^m \times [0,1]^n$, $r = r_{\mu,\epsilon}$ *defined as in* (6.18), *and* $\bar{x} \in \mathbb{R}^m_{>0}$, $\langle x, \mathbf{A}y\rangle \le \|x\|_{\mathbf{diag}\left(\frac{1}{\bar{x}}\right)} \|y\|_{\mathbf{diag}\left(|\mathbf{A}|^\top \bar{x}\right)}$. *Further, if* $\rho \ge 3$, *the matrix*

$$\mathbf{D}(x) := \begin{pmatrix} \frac{\rho}{2}\mathbf{diag}\left(\frac{1}{x}\right) & \mathbf{0} \\ \mathbf{0} & \frac{1}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x\right) \end{pmatrix} \tag{6.22}$$

satisfies the following relationship with the Hessian matrix of $r(z)$:

$$\mathbf{D}(x) \preceq \nabla^2 r(z) = \begin{pmatrix} \rho \cdot \mathbf{diag}\left(\frac{1}{x}\right) & \frac{2}{\rho}\mathbf{A}\mathbf{diag}\left(y\right) \\ \frac{2}{\rho}\mathbf{diag}\left(y\right)\mathbf{A}^\top & \frac{2}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x\right) \end{pmatrix} \preceq 4\mathbf{D}(x). \tag{6.23}$$

We also introduce the following notion of a padding oracle (cf. Definition 2 of [112]), which helps us control the multiplicative stability of iterates when running our algorithm.

Definition 26. *Given* $\delta > 0$, *and any* $\bar{z} = (\bar{x}, y) \in \Delta^m \times [0,1]^n$, *a padding oracle* $\mathcal{O}_\delta$ *returns* $z = (x, y)$ *by setting* $\hat{x}_i = \max(\bar{x}_i, \delta)$ *coordinate-wise and letting* $x = \frac{\hat{x}}{\|\hat{x}\|_1}$.

This padding oracle has two merits which we exploit. First, the error incurred due to padding is small proportional to the padding size δ, which finds usage in proving the correctness of our main algorithm, Algorithm 23 (see Proposition 20).

Lemma 83 (Error of padding, cf. Lemma 6, [112]). *For* $\delta > 0$ *and* $\bar{z} = (\bar{x}, y) \in \Delta^m \times [0,1]^n$ *let* $z = (x, y) \in \Delta^m \times [0,1]^n$ *where* $x = \mathcal{O}_\delta(\bar{x})$ *(Definition 26), then for* r *in* (6.18), *and any* $w \in \mathcal{Z} = \Delta^m \times [0,1]^n$, $V_z^r(w) - V_{\bar{z}}^r(w) \le \left(\rho + \frac{8}{\rho}\right) m\delta$.

Second, padding ensures that the iterates of our algorithm satisfy $x = \Omega(\delta)$ entrywise, i.e. no entries of our simplex iterates x are too small. This helps ensure the stability of iterates throughout one call of Algorithm 22, formally through the next lemma.

Lemma 84 (Iterate stability in Algorithm 22). *Suppose* $\epsilon \le 1$, $\rho \ge 6$, *and* $\alpha \ge \frac{36}{\rho}(\mu \log \frac{4}{\delta} + 3C_{\max})$. *Let* (x_k, y_k) *denote blocks of* z_k, *the* k^{th} *iterate of Algorithm 21. In any iteration* k *of Algorithm 21, calling Algorithm 22 to implement Line 6, if* $x_{k-1} \ge \frac{\delta}{2}$ *entrywise,* $x^{(t+1)} \in x_{k-1} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]$, *for all* $t \in [T]$. *Calling Algorithm 22 to implement Line 7, if* $x_{k-1/2} \ge \frac{\delta}{4}$ *entrywise,* $x^{(t+1)} \in x_{k-1}^{\frac{\alpha}{\alpha+\nu}} \circ x_{k-1/2}^{\frac{\nu}{\alpha+\nu}} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]$ *for all* $t \in [T]$.

6.4.3 Regularized box-simplex solver and its guarantees

We give our full high-accuracy regularized box-simplex game solver as Algorithm 23, which combines Algorithm 21, Algorithm 22, and a padding step to ensure stability.

In order to analyze the convergence of Algorithm 23, we begin by observing that the operator in (6.17) satisfies strong monotonicity with respect to our regularizer (6.18).

Algorithm 23: RegularizedBS$(\mathbf{A}, b, c, \epsilon, \mu, \varepsilon)$

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, accuracy $\varepsilon \in (m^{-10}, 1)$, $72\epsilon \le \mu \le 1$;
Output: Approximate solution pair (x, y) to (6.16);
Global: $\delta \leftarrow \frac{\epsilon\varepsilon^2}{m^2}$, $\rho \leftarrow \sqrt{\frac{2\mu}{\epsilon}}$, $\nu \leftarrow \frac{1}{2}\sqrt{\frac{\mu\epsilon}{2}}$, $\alpha \leftarrow 18C_{\max} + 32\sqrt{\frac{\mu\epsilon}{2}}\log\frac{4}{\delta}$;
Global: $T \leftarrow O\left(\log \frac{mnB_{\max}\alpha\rho}{\delta\varepsilon}\right)$, $K \leftarrow O\left(\frac{\alpha}{\nu}\log\left(\frac{\nu\log m}{\varepsilon}\right)\right)$ for appropriate constants;
$(x_0, y_0) \leftarrow (\frac{1}{m}\cdot\mathbf{1}_m, \mathbf{0}_n)$;
for $k = 1$ *to* K **do**
  $(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}) \leftarrow$ GradBS$(x_{k-1}, y_{k-1}, x_{k-1}, y_{k-1}, 0)$;
  $(x_{k-\frac{1}{2}}, y_{k-\frac{1}{2}}) \leftarrow$ AltminBS$(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \alpha, x_{k-1}, y_{k-1})$;
  $(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}) \leftarrow$ GradBS$(x_{k-\frac{1}{2}}, y_{k-\frac{1}{2}}, x_{k-1}, y_{k-1}, \nu)$;
  $(x^{(T+1)}, y^{(T)}) \leftarrow$ AltminBS$(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \alpha+\nu, x_{k-\frac{1}{2}}, y_{k-\frac{1}{2}})$;
  $x_k \leftarrow \frac{1}{\left\|\max\left(x^{(T+1)}, \delta\right)\right\|_1}\cdot\max\left(x^{(T+1)}, \delta\right)$, $y_k \leftarrow y^{(T)}$; ▷ Implement padding $\mathcal{O}_\delta(x^{(T+1)})$
function GradBS(x, y, x_0, y_0, Θ)
  $g^{\mathsf{x}} \leftarrow \mathbf{A}y + c + \mu(\mathbf{1} + \log(x)) - \frac{\epsilon}{2}|\mathbf{A}|(y^2)$;
  $g^{\mathsf{y}} \leftarrow -\mathbf{A}^\top x + b + \epsilon\mathbf{diag}(y)|\mathbf{A}|^\top x$;
  $g_r^{\mathsf{x}} \leftarrow -\alpha\rho(1+\log x_0) - \frac{\alpha}{\rho}|\mathbf{A}|y_0^2 - \Theta\rho(1+\log x) - \frac{\Theta}{\rho}|\mathbf{A}|y^2$;
  $g_r^{\mathsf{y}} \leftarrow -\frac{2\alpha}{\rho}\mathbf{diag}(y_0)|\mathbf{A}|^\top x_0 - \frac{2\Theta}{\rho}\mathbf{diag}(y)|\mathbf{A}|^\top x)$;
  return $(g^{\mathsf{x}} + g_r^{\mathsf{x}}, g^{\mathsf{y}} + g_r^{\mathsf{y}})$
function AltminBS$(\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}}, \theta, x^{(0)}, y^{(0)})$ ▷ Implement approximate proximal oracle via AltMin
  for $0 \le t \le T$ **do**
    $x^{(t+1)} \leftarrow \frac{1}{\left\|\exp\left(-\frac{1}{\theta\rho}\gamma^{\mathsf{x}} - \frac{1}{\rho^2}|\mathbf{A}|\left(y^{(t)}\right)^2\right)\right\|_1}\cdot\exp\left(-\frac{1}{\theta\rho}\gamma^{\mathsf{x}} - \frac{1}{\rho^2}|\mathbf{A}|\left(y^{(t)}\right)^2\right)$;
    $y^{(t+1)} \leftarrow \mathrm{med}\left(0, 1, -\frac{\rho}{2\theta}\cdot\frac{\gamma^{\mathsf{y}}}{|\mathbf{A}|^\top x^{(t+1)}}\right)$;
  Return: $(x^{(T+1)}, y^{(T)})$;

Lemma 85 (Strong monotonicity)**.** *Let $\mu \ge \frac{\epsilon}{2}$ and $\rho := \sqrt{\frac{2\mu}{\epsilon}}$. The gradient operator $g_{\mu,\epsilon}$* (6.17) *is $\nu := \frac{1}{2}\sqrt{\frac{\mu\epsilon}{2}}$-strongly monotone (see* (6.19)*) with respect to $r_{\mu,\epsilon}$ defined in* (6.18)*.*

Next, we show iterate stability through each loop of alternating minimization (i.e. from Line 7 to Line 8, and Line 9 to Line 10 respectively), via Lemma 84.

Corollary 19 (Iterate stability in Algorithm 23)**.** *Assume the same parameter bounds as Lemma 84, and that $\delta \in (0, m^{-1})$. In the k^{th} outer loop of Algorithm 23, $x_{k-1} \ge \frac{\delta}{2}$ entrywise. Further, for all iterates $x^{(t+1)}$ computed in Line 7 to Line 8 and x_{OPT} as defined in* (6.21) *with $\theta = \alpha$, $\frac{1}{2}x_{k-1} \le x^{(t+1)}, x_{\mathsf{OPT}} \le 2x_{k-1}$, and $x^{(t+1)}, x_{\mathsf{OPT}} \ge \frac{\delta}{4}$, entrywise. Similarly, for all iterates $x^{(t+1)}$ computed in Line 9 to Line 10 and x_{OPT} as defined in* (6.21) *with $\theta = \alpha+\nu$, $\frac{1}{2}x_{k-1/2} \le x^{(t+1)}, x_{\mathsf{OPT}} \le 2x_{k-1/2}$ and $x^{(t+1)}, x_{\mathsf{OPT}} \ge \frac{\delta}{4}$, entrywise.*

Under iterate stability, our next step is to prove that our operator $g_{\mu,\epsilon}$ is relatively Lipschitz with respect to our regularizer $r_{\mu,\epsilon}$ (as defined in (6.20)).

Lemma 86 (Relative Lipschitzness)**.** *Assume the same parameter bounds as in Lemma 84. In the k^{th} outer loop of Algorithm 23, let $\bar{z}_k \leftarrow (x^{(T+1)}, y^{(T)})$ from Line 10 be z_k before the padding operation. Then, $x_{k-1/2}, \bar{x}_k \in [\frac{1}{2}x_{k-1}, 2x_{k-1}]$ elementwise and*

$$\left\langle g(z_{k-1/2}) - g(z_{k-1}), z_{k-1/2} - \bar{z}_k \right\rangle \leq \alpha \left(V_{z_{k-1}}(z_{k-1/2}) + V_{z_{k-1/2}}(\bar{z}_k) \right) \quad \text{for} \quad \alpha = 4 + 32\sqrt{\frac{\mu\epsilon}{2}}.$$

Next, we give a convergence guarantee on the inner loops (from Line 7 to Line 8, and Line 9 to Line 10) in Algorithm 23, as an immediate consequence of Corollary 18.

Corollary 20 (Inner loop convergence in Algorithm 23)**.** *Assume the same parameter bounds as in Lemma 84. For γ defined in Line 7, suppose for an appropriate constant $T = \Omega\left(\log \frac{mnB_{\max}\alpha\rho}{\delta\varepsilon}\right)$. Then, for all k iterate $z_{k-1/2} = (x_{k-1/2}, y_{k-1/2})$ of Line 8 satisfies*

$$\left\langle \nabla g(z_{k-1}) + \alpha \nabla V_{z_{k-1}}(z_{k-1/2}), z_{k-1/2} - w \right\rangle \leq \frac{\nu\varepsilon}{4}, \text{ for all } w \in \mathcal{Z}.$$

Similarly, for γ defined in Line 9, iterate $\bar{z}_k = (x_{(T+1)}, y_{(T)})$ of Line 10 satisfies

$$\left\langle \nabla g(z_{k-1}) + \alpha \nabla V_{z_{k-1}}(\bar{z}_k) + \nu \nabla V_{z_{k-1/2}}(\bar{z}_k), \bar{z}_k - w \right\rangle \leq \frac{\nu\varepsilon}{4}, \text{ for all } w \in \mathcal{Z}.$$

We now analyze the progress made by each outer loop of Algorithm 23. The proof is very similar to that of Proposition 19; the only difference is controlling the extra error incurred in the padding step of Line 11, which we bound via Lemma 83.

Proposition 20 (Convergence of Algorithm 23)**.** *Assume the same parameter bounds as in Lemma 84, and that $\delta \leq \frac{\varepsilon}{4\rho\alpha m}$. Algorithm 23 returns z_K satisfying $V^r_{z_K}(z^\star) \leq \frac{3\varepsilon}{\nu}$, letting (for an appropriate constant) $K = \Omega(\frac{\alpha}{\nu}\log(\frac{\nu \log m}{\varepsilon}))$.*

We are now ready to prove the main theorem of this section, which gives a complete convergence guarantee of Algorithm 23 by combining our previous claims.

Theorem 41 (Regularized box-simplex solver)**.** *Given regularized box-simplex game* (6.16) *with $72\epsilon \leq \mu \leq 1$ and optimizer $(x^\star, y^\star)$, and letting $\varepsilon \in (m^{-10}, 1)$,* RegularizedBS *(Algorithm 23) returns x^K satisfying $\left\|x^K - x^\star\right\|_1 \leq \frac{\varepsilon}{C_{\max}\log^2 m}$ and $\max_{y\in[0,1]^n} f_{\mu,\epsilon}(x^K, y) - f_{\mu,\epsilon}(x^\star, y^\star) \leq \varepsilon$. The total runtime of the algorithm is $O(\mathrm{nnz}(\mathbf{A}) \cdot (\frac{C_{\max}}{\sqrt{\mu\epsilon}} + \log(\frac{m}{\sigma\epsilon})) \cdot \log(\frac{C_{\max}\log m}{\varepsilon}) \log(\frac{mnB_{\max}}{\varepsilon}))$.*

As a corollary, we obtain an approximate solver for regularized box-simplex games in the following form (which in particular does not include a quadratic regularizer):

$$\min_{x\in\Delta^m} \max_{y\in[0,1]^n} f_\mu(x,y) = y^\top \mathbf{A}^\top x + c^\top x - b^\top y + \mu H(x), \quad \text{where} \quad H(x) := \sum_{i\in[m]} x_i \log x_i. \tag{6.24}$$

Corollary 21 (Half-regularized approximate solver)**.** *Given regularized box-simplex game* (6.24) *with regularization parameters* $72\epsilon \leq \mu \leq 1$ *and optimizer* $(x^\star, y^\star)$*, and letting* $\epsilon \in (m^{-10}, 1)$*, Algorithm 23 with* $\varepsilon \leftarrow \frac{\epsilon}{2}$ *returns* x^K *satisfying* $\max_{y\in[0,1]^n} f_\mu(x^K, y) - f_\mu(x^\star, y^\star) \leq \epsilon$*. The total runtime of the algorithm is* $O\left(\mathrm{nnz}(\mathbf{A}) \cdot (\frac{C_{\max}}{\sqrt{\mu\epsilon}} + \log\left(\frac{m}{\epsilon}\right)) \cdot \log\left(\frac{C_{\max}\log m}{\epsilon}\right) \log\left(\frac{mnB_{\max}}{\epsilon}\right)\right)$.

Part II

Semidefinite Programming and High-Dimensional Statistics

Chapter 7

Matrix Multiplicative Weights and Friends

This chapter is based in part on [108, 289, 180, 287], with Yair Carmon, Ilias Diakonikolas, John C. Duchi, Arun Jambulapati, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, Christopher Musco, and Aaron Sidford.

7.1 Introduction

In this chapter, we present several related results on variants of the matrix multiplicative weights method [534, 34], a meta-algorithm for regret minimization over subsets of the positive semidefinite cone. Using these tools, we develop faster solvers for various approximate formulations of semidefinite programming, as well as new algorithms for applications in high-dimensional statistics in Chapters 8 and 9. Further exposition on the various relationships between the tools developed in this chapter, as well as their implications, can be found in Chapter 1.

7.1.1 Sketching matrix multiplicative weights

Consider the problem of online learning over the spectrahedron $\mathcal{\Delta}_n$, the set of $n \times n$ symmetric positive semidefinite matrices with unit trace. At every time step t, a player chooses action $\mathbf{X}_t \in \mathcal{\Delta}_n$, an adversary supplies symmetric gain matrix $\mathbf{G}_t$, and the player earns reward $\langle \mathbf{G}_t, \mathbf{X}_t \rangle := \mathrm{tr}(\mathbf{G}_t \mathbf{X}_t)$. We seek to minimize the regret with respect to the best single action (in hindsight),

$$\sup_{\mathbf{X} \in \mathcal{\Delta}_n} \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X} \rangle - \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle = \lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t \right) - \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle. \tag{7.1}$$

[534, 535] solve this problem using the matrix exponentiated gradient algorithm [511], also known as matrix multiplicative weights (MMW). It is given by

$$\mathbf{X}_t = \mathsf{P}^{\mathrm{mw}}\left(\eta \sum_{i=1}^{t-1} \mathbf{G}_i\right), \quad \text{where} \quad \mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) := \frac{e^{\mathbf{Y}}}{\operatorname{tr} e^{\mathbf{Y}}}, \tag{7.2}$$

and $\eta > 0$ is a step size parameter. If the operator norm $\|\mathbf{G}_t\|_\infty \leq 1$ for every t, using the MMW strategy (7.2) with $\eta = \sqrt{2\log(n)/T}$ guarantees that the regret (7.1) is bounded by $\sqrt{2\log(n)T}$; this guarantee is minimax optimal up to a constant [33].

Unlike standard (vector) multiplicative weights, MMW is computational expensive to implement in the high-dimensional setting $n \gg 1$. This is due ot the high cost of computing matrix exponentials; currently they require an eigen-decomposition which costs $\Theta(n^3)$ with practical general-purpose methods and $\Omega(n^\omega)$ in theory [433]. This difficulty has led a number of researchers to consider a rank-k sketch of P^{mw} of the form

$$\mathsf{P}_{\mathbf{U}}(\mathbf{Y}) := \frac{e^{\mathbf{Y}/2}\mathbf{U}\mathbf{U}^T e^{\mathbf{Y}/2}}{\langle e^{\mathbf{Y}}, \mathbf{U}\mathbf{U}^T\rangle}, \quad \text{where} \quad \mathbf{U} \in \mathbb{R}^{n\times k} \tag{7.3}$$

and the elements of $\mathbf{U}$ are i.i.d. standard Gaussian. For $k \ll n$, $\mathsf{P}_{\mathbf{U}}$ is much cheaper than P^{mw} to compute, since its computation requires only k products of the form $e^{\mathbf{A}}b$ which can be evaluated efficiently via iterative methods (see Section 7.2.3). Since we play rank-deficient matrices, an adversary with knowledge of $\mathbf{X}_t$ may choose the gain $\mathbf{G}_t$ to be in its nullspace, incurring regret linear in T. To rule such an adversary out, we assume that $\mathbf{G}_t$ and $\mathbf{X}_t$ must be chosen simultaneously. We formalize this as

Assumption 5. *Conditionally on $\mathbf{X}_1, \mathbf{G}_1, \ldots, \mathbf{X}_{t-1}, \mathbf{G}_{t-1}$, the gain $\mathbf{G}_t$ is independent of $\mathbf{X}_t$.*

This assumption is standard in the literature on adversarial bandit problems [99] where it is similarly unavoidable. While it comes at significant loss of generality, Assumption 5 holds in two important applications, as described below.

The challenge of bias Assumption 5 allows us to write

$$\mathbb{E}\left[\left\langle \mathbf{G}_t, \mathsf{P}_{U_t}\left(\eta\sum_{i=1}^{t-1}\mathbf{G}_i\right)\right\rangle | \{\mathbf{G}_i\}_{i=1}^t\right] = \left\langle G_t, \mathbb{E}_U \mathsf{P}_U\left(\eta\sum_{i=1}^{t-1}\mathbf{G}_i\right)\right\rangle.$$

However, even though U satisfies $\mathbb{E}_U UU^T = I$, we have $\mathbb{E}_U \mathsf{P}_U(\mathbf{Y}) \neq \mathsf{P}^{\mathrm{mw}}(\mathbf{Y})$ for general $\mathbf{Y}$. Therefore, the guarantees of MMW do not immediately apply to actions chosen according to the sketch (7.3), even in expectation. A common solution in the literature [34, 442, 19] is to pick $k = \widetilde{O}\left(1/\epsilon^2\right)$ such that, by the Johnson-Lindenstrauss lemma, $\mathsf{P}_U(\mathbf{Y})$ approximates $\mathsf{P}^{\mathrm{mw}}(\mathbf{Y})$ to within multiplicative error ϵ. This makes the MMW guarantees applicable again, but requires

considerable computation per step, that will match the cost of full matrix exponentiation for sufficiently small ϵ. [304, 20] prove regret guarantees for sketches of fixed rank $k \leq 3$ with forms different from (7.3); we discuss their approaches in detail later in this section.

Our approach. In this work we use the sketch (7.3) with $k = 1$, playing the rank-1 matrix $\mathbf{X}_t = \mathsf{P}_{u_t}(\eta \sum_{i=1}^{t-1} \mathbf{G}_i)$ where $\mathsf{P}_u(\mathbf{Y}) = vv^T/(v^T v)$ for $v = e^{\mathbf{Y}/2}u$ and $u_t \in \mathbb{R}^n$ standard Gaussian. Instead of viewing P_u as a biased estimator of P^{mw}, we define the deterministic function

$$\bar{\mathsf{P}}(\mathbf{Y}) := \mathbb{E}_u \mathsf{P}_u(\mathbf{Y}),$$

and view P_u as an unbiased estimator for $\bar{\mathsf{P}}$. Our primary contribution is in showing that

$$\bar{\mathsf{P}} \text{ is nearly as good a mirror projection as } \mathsf{P}^{\text{mw}}.$$

More precisely, we show that replacing P^{mw} with $\bar{\mathsf{P}}$ leaves the regret bounds almost unchanged; if $\|\mathbf{G}_t\|_\infty \leq 1$ for every t, the actions $\bar{\mathbf{X}}_t = \bar{\mathsf{P}}(\eta \sum_{i=1}^{t-1} \mathbf{G}_i)$ guarantee (with properly tuned η) regret of at most $\sqrt{6 \log(4n)T}$, worse than MMW by only a factor of roughly $\sqrt{3}$. To prove this, we establish that $\bar{\mathsf{P}}$ possesses the geometric properties necessary for mirror descent analysis: it is Lipschitz continuous and its associated Bregman divergence is appropriately bounded. Since P_u is—by definition—an unbiased estimator of $\bar{\mathsf{P}}$, we immediately obtain (thanks to Assumption 5) that $X_t = \mathsf{P}_{u_t}(\eta \sum_{i=1}^{t-1} \mathbf{G}_i)$ satisfies the same regret bound in expectation. High-probability bounds follow immediately via martingale concentration.

Application to online PCA. As our sketched actions are of the form $\mathbf{X}_t = x_t x_t^T$, the regret they incur is $\lambda_{\max}\big(\sum_{t=1}^T \mathbf{G}_t\big) - \sum_{t=1}^T x_t^T \mathbf{G}_t x_t$. Therefore, the vectors x_t can be viewed as streaming approximations of the principal component[1] of the cumulative matrix $\sum_{i=1}^{t-1} \mathbf{G}_i$. This online counterpart of the classical principal component analysis problem is the topic of a number of prior works [534, 232, 20]. Our sketch offers regret bounds that are optimal up to constants, with computational cost per step as low as any known alternative, and overall computational cost better than any in the literature by a factor of at least $\log^5 n$. Our regret bounds hold for gains $\mathbf{G}_t$ of any rank or sparsity, and our computational scheme (Section 7.2.3) naturally leverages low rank and/or sparsity in the gains.

Application to semidefinite programming (SDP). Any feasibility-form SDP is reducible to the matrix saddle-point game $\max_{\mathbf{X} \in \Delta_n} \min_{y \in \sigma_m} \langle \sum_{i=1}^m y_i \mathbf{A}_i, \mathbf{X} \rangle$, where σ_m is the simplex in $\mathbb{R}^m$ and $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathbb{R}^{n \times n}$ are symmetric matrices. A simple procedure for approximating a saddle-point (Nash equilibrium) for this game is to have each player perform online learning, where the max-player observes gains $\mathbf{G}_t = \sum_{i=1}^m [y_t]_i \mathbf{A}_i$ and the min-player observes costs $[c_t]_i = \langle \mathbf{A}_i, \mathbf{X}_t \rangle$. Using standard/matrix multiplicative weights for the min/max players, respectively, we may produce approximate solutions with additive error ϵ in $O(\log(nm)/\epsilon^2)$ iterations, with each iteration costing

[1] For this reason we consider gain-maximization rather than loss-minimization, which is generally more conventional.

$O(n^3)$ time, due to the MMW computation. In Section 7.2.4 we show that by replacing MMW with our sketch we guarantee ϵ error in a similar number of iterations, but with each iteration costing $\widetilde{O}\left(N/\sqrt{\epsilon}\right)$, where N is the problem description size, which is often significantly smaller than n^2. This guarantee matches the state-of-the-art in a number of settings.

Related work

MMW appears in a large body of work spanning optimization, theoretical computer science, and machine learning [415, 534, 33]. Here, we focus on works that, like us, attempt to relieve the computational burden of computing the matrix exponential, while preserving the MMW regret guarantees. To our knowledge, the first proposal along these lines is due to [34], who apply MMW with a Johnson-Lindenstrauss sketch to semidefinite relaxations of combinatorial problems. Subsequent works on positive semidefinite programming adopted this technique [442, 19]. To achieve ϵ-accurate solutions, these works require roughly ϵ^{-2} matrix exponential vector products per mirror projection.

[50] apply the accelerated mirror-prox scheme of [415] to matrix saddle-point problems and approximate P^{mw} using the rank-k sketch (7.3). Instead of appealing to the JL lemma, they absorb the bias and variance of this approximation directly into the algorithm's error estimates. This enables a more parsimonious choice of k; to attain additive error ϵ, they require $k = \widetilde{O}\left(\epsilon^{-1}\right)$. See Section 7.2.4 for additional discussion of the performance of this method.

A different line of work, called Follow the Perturbed Leader (FTPL) [304], eschews matrix exponentiation, and instead produces rank-1 actions $\mathbf{X}_t = x_t x_t^T$, where x_t is an approximate top eigenvector of a random perturbation of $\sum_{i=1}^{t-1} \mathbf{G}_i$. While a single eigenvector computation has roughly the same cost as a single matrix-exponential vector product, the regret bounds for FTPL—and hence also the total work—scale polynomially in the problem dimension n: [232] bound the regret by $\widetilde{O}\left(\sqrt{nT}\right)$ and [211] improve the bound to $\widetilde{O}\left(\sqrt{n^{1/2}T}\right)$ for gains of rank 1. In contrast, the regret of MMW and its sketches depends on n only logarithmically.

[20] give the first fixed-rank sketch with MMW-like regret, proposing a scheme called Follow the Compressed Leader (FTCL). Their approach is based on replacing the MMW mirror projection (7.2) with the projection corresponding to $\ell_{1-1/q}$ regularization, given by $\mathsf{P}^{\text{q-reg}}(\mathbf{Y}) := (c(\mathbf{Y})I - \mathbf{Y})^{-q}$ where $c(\mathbf{Y})$ is the unique $c \in \mathbb{R}$ such that $cI - \mathbf{Y} \succ 0$ and $\operatorname{tr}[(cI - \mathbf{Y})^{-q}] = 1$. They use a sketch of $\mathsf{P}^{\text{q-reg}}$ similar in spirit to (7.3) and prove that $k = 3$ suffices to obtain regret bounds within a polylogarithmic factor of MMW, with q chosen to be roughly $\log n$.

The basis of the FTCL proof strategy is a potential argument used to derive regret bounds for the exact $\mathsf{P}^{\text{q-reg}}$. Their analysis consists of carefully tracing this argument, and accounting for the errors caused by sketching in each step of the way. In comparison, we believe our analysis is more transparent; rather than control multiple series expansion error terms, we establish three simple geometric properties of our projection $\bar{\mathsf{P}}$. We also provide tighter bounds; to guarantee ϵ average regret, FTCL requires a factor of $\Omega(\log^5(n/\epsilon))$ more online learning steps than our method. The

per-step computational cost of our method is similar to that of FTCL, with better polylogarithmic dependence on n. On a practical note, the computational scheme we describe in Section 7.2.3 is significantly simpler to implement than the one proposed for FTCL.

7.1.2 Ky Fan matrix multiplicative weights

We give a Ky Fan k-norm (sum of k largest eigenvalues) generalization of MMW, which typically gives operator norm guarantees. We analyze our algorithm and show that it is tolerant to the error guarantees of approximate k-PCA procedures such as simultaneous power iteration [404]. Crucial to our tightest runtime bounds are strengthenings of the analysis of a similar procedure found in [136] in several places, which save multiple k factors in our guarantees and may be of independent interest; we now highlight a few here.[2]

The main idea of our Ky Fan MMW regret guarantee is to bound the cost of actions $\{\mathbf{Y}_t\}_{t\geq 0}$ against a sequence of positive semidefinite "gain matrices" $\{\mathbf{G}_t\}_{t\geq 0}$ as measured by inner products. The actions $\{\mathbf{Y}_t\}_{t\geq 0}$ are given by the algorithm (depending on the gain matrices), and live in

$$\mathcal{Y} := \{\mathbf{Y} \in \mathbb{R}^{d\times d} \mid \mathbf{0} \preceq \mathbf{Y} \preceq \mathbf{I}, \mathrm{Tr}(\mathbf{Y}) = k\}.$$

The reason for this choice of action set, the "k-Fantope," is because it satisfies

$$\sup_{\mathbf{U}\in\mathcal{Y}} \langle \mathbf{U}, \mathbf{G}\rangle = \|\mathbf{G}\|_k,$$

where $\|\cdot\|_k$ is the Ky Fan k-norm, so the best action in hindsight captures this norm. Ultimately, our filtering scheme requires matrix-vector query access to each $\mathbf{Y}_t$, which are defined by *Bregman projections* onto the set $\mathcal{Y}$. It was shown in [136] that the natural choice of projection, induced by a regularizer $r(\mathbf{Y})$ chosen to be matrix entropy, is a truncated exponential, where truncation occurs on the top-k eigenspace. The bottleneck cost of iterations is computing this space.

To this end, we show new guarantees on the performance of approximate k-PCA, which allow for their use in this process. One example is that we show roughly $\frac{1}{\epsilon}$ iterations of simultaneous power iteration on a positive semidefinite matrix $\mathbf{S} \in \mathbb{R}^{d\times d}$, resulting in approximate eigenvectors $\mathbf{V} \in \mathbb{R}^{d\times k}$, are enough to guarantee (cf. Proposition 24)

$$(1-\epsilon)\mathbf{S} \preceq \mathbf{PSP} + (\mathbf{I}-\mathbf{P})\,\mathbf{S}\,(\mathbf{I}-\mathbf{P}) \preceq (1+\epsilon)\mathbf{S}, \text{ where } \mathbf{P} := \mathbf{V}\mathbf{V}^\top.$$

This improves a similar analysis in [136], which showed an approximation factor of $1 \pm k\epsilon$.

The main other technical piece required by our MMW algorithm is a refined divergence bound

[2] We believe that similar wins following from our tighter analysis apply to the algorithm of [136].

of the form (cf. Lemma 88 for a formal statement)

$$V_{\mathbf{S}}^{r^*}(\mathbf{S}+\eta\mathbf{G}) \leq \|\eta\mathbf{G}\|_{\mathrm{op}} \langle \eta\mathbf{G}, \mathbf{Y}\rangle, \text{ where } \mathbf{Y} := \nabla r^*(\mathbf{S}) \in \mathcal{Y},$$
$$\text{a strengthening of } V_{\mathbf{S}}^{r^*}(\mathbf{S}+\eta\mathbf{G}) \leq k\|\eta\mathbf{G}\|_{\mathrm{op}}^2.$$

Here, V^{r^*} is the Bregman divergence in the convex conjugate of r. The latter bound follows easily from strong convexity of r (and hence smoothness of its dual); we require the former strengthening so that we can use the action matrices $\{\mathbf{Y}_t\}_{t\geq 0}$ to define scores, to decrease inner products.[3] In particular, the weaker bound above has no dependence on $\mathbf{Y}$, so without the stronger bound it is unclear how to use the MMW update structure to downweight.

We prove our refined divergence bound by adapting arguments from previous literature [108, 285] on using Hessian formulae of spectral functions to prove divergence bounds, whenever the conjugate r^* is twice-differentiable, and applying the Alexandrov theorem. Finally, up to (non-dominant) approximation error terms, our Ky Fan MMW procedure's main guarantee can be stated as: given a sequence of positive semidefinite matrices $\{\mathbf{G}_t\}_{t\geq 0}$, let step size $\eta > 0$ satisfy $\eta\mathbf{G}_t \preceq \mathbf{I}$ for all t. The procedure plays a sequence $\{\mathbf{Y}_t\}_{t\geq 0} \in \mathcal{Y}$, so that for any $T \in \mathbb{N}$,

$$\left\|\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{G}_t\right\|_k \leq \frac{2}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle + \frac{k\log d}{\eta T}. \tag{7.4}$$

7.1.3 Matrix dictionary recovery SDPs

Positive semidefinite programs are a wide family of structured SDPs with numerous applications [241, 36, 281, 353, 133, 131, 135, 136], which contain "pure packing" SDPs and "pure covering" SDPs as special cases. We use the term positive semidefinite programming in this paper to refer to the fully general mixed packing-covering SDP problem, which is parameterized by "packing" and "covering" matrices $\{\mathbf{P}_i\}_{i\in[n]}, \mathbf{P}, \{\mathbf{C}_i\}_{i\in[n]}, \mathbf{C} \in \mathbb{S}^d_{\geq 0}$, and asks to find the smallest $\mu > 0$ such that there exists $w \in \mathbb{R}^n_{\geq 0}$ with

$$\sum_{i\in[n]} w_i\mathbf{P}_i \preceq \mu\mathbf{P}, \ \sum_{i\in[n]} w_i\mathbf{C}_i \succeq \mathbf{C}. \tag{7.5}$$

By redefining $\mathbf{P}_i \leftarrow \frac{1}{\mu}\mathbf{P}^{-\frac{1}{2}}\mathbf{P}_i\mathbf{P}^{-\frac{1}{2}}$ and $\mathbf{C}_i \leftarrow \mathbf{C}^{-\frac{1}{2}}\mathbf{C}_i\mathbf{C}^{-\frac{1}{2}}$ for all $i \in [n]$ and a given $\mu > 0$, the optimization problem in (7.5) is equivalent to testing whether there exists $w \in \mathbb{R}^n_{\geq 0}$ such that

$$\sum_{i\in[n]} w_i\mathbf{P}_i \preceq \sum_{i\in[n]} w_i\mathbf{C}_i. \tag{7.6}$$

The above formulation (7.6) was studied by [283, 285], and an important open problem in the field of structured convex programming is designing a "width-independent" solver for testing the

[3] It is a strengthening since $\nabla r^*(\mathbf{S}) \in \mathcal{Y}$, so we can apply a matrix Hölder's inequality and use $\mathrm{Tr}(\mathbf{Y}) = k$, $\forall \mathbf{Y} \in \mathcal{Y}$.

feasibility of (7.6) up to a $1+\epsilon$ factor (namely, testing whether (7.6) is approximately feasible with an iteration count polynomial in ϵ^{-1} and polylogarithmic in other problem parameters), or solving for the optimal μ in (7.5) to this approximation factor. Up to this point, such width-independent solvers have remained elusive beyond the case of pure packing SDPs [19, 442, 289].

This work develops an efficient algorithm for solving specializations of (7.5), (7.6) where the packing and covering matrices $\{\mathbf{P}_i\}_{i\in[n]}, \{\mathbf{C}_i\}_{i\in[n]}$, as well as the constraints $\mathbf{P}$, $\mathbf{C}$, are multiples of each other. In particular, we give efficient algorithms for the following special case of (7.5), (7.6). Given a set of matrices (a "matrix dictionary") $\{\mathbf{M}_i\}_{i\in[n]} \in \mathbb{S}^d_{\geq 0}$, a constraint matrix $\mathbf{B}$, and a tolerance $\epsilon \in (0,1)$, such that there exists a feasible set of weights $w^\star \in \mathbb{R}^n_{\geq 0}$ with

$$\mathbf{B} \preceq \sum_{i\in[n]} w_i^\star \mathbf{M}_i \preceq \kappa \mathbf{B}, \tag{7.7}$$

for some unknown $\kappa \geq 1$, we wish to return a set of weights $w \in \mathbb{R}^n_{\geq 0}$ such that

$$\mathbf{B} \preceq \sum_{i\in[n]} w_i \mathbf{M}_i \preceq (1+\epsilon)\kappa \mathbf{B}. \tag{7.8}$$

While this "matrix dictionary recovery" problem is a restricted case of (7.5), as we demonstrate, it is already expressive enough to capture many interesting applications.

Our results concerning the problem (7.7), (7.8) will assume the family $\{\mathbf{M}_i\}_{i\in[n]}$ is "simple" in the following two senses: First, we assume we have an explicit factorization of each $\mathbf{M}_i$ as

$$\mathbf{M}_i = \mathbf{V}_i \mathbf{V}_i^\top, \ \mathbf{V}_i \in \mathbb{R}^{d\times m}. \tag{7.9}$$

In our applications $m = 1$, but our solver also handles the general case. Second, recalling the shorthand $\mathcal{M}(w) := \sum_{i\in[n]} w_i \mathbf{M}_i$, we assume that we can approximately solve systems in $\mathcal{M}(w)+\lambda\mathbf{I}$ for any $w \in \mathbb{R}^n_{\geq 0}$, $\lambda \geq 0$. Concretely, we assume for any $\epsilon > 0$, there is a linear operator $\widetilde{\mathcal{M}}_{w,\lambda,\epsilon}$ which we can compute and apply in $\mathcal{T}^{\text{sol}}_{\mathcal{M}} \cdot \log \frac{1}{\epsilon}$ time[4], and

$$\left\| \widetilde{\mathcal{M}}_{w,\lambda,\epsilon} v - (\mathcal{M}(w)+\lambda\mathbf{I})^{-1} v \right\|_2 \leq \epsilon \left\| (\mathcal{M}(w)+\lambda\mathbf{I})^{-1} v \right\|_2 \text{ for all } v \in \mathbb{R}^d. \tag{7.10}$$

Under these assumptions, we prove the following main claims in Section 7.4. We remark that we did not try to heavily optimize logarithmic factors and the dependence on ϵ^{-1}, as many of these factors are inherited from subroutines from prior work.

Theorem 42. *Given matrices $\{\mathbf{M}_i\}_{i\in[n]}$ with explicit factorizations (7.9), such that (7.7) is feasible for $\mathbf{B} = \mathbf{I}$ and some $\kappa \geq 1$, we can return weights $w \in \mathbb{R}^n_{\geq 0}$ satisfying (7.8) with probability $\geq 1-\delta$*

[4]We use this notation because, if $\mathcal{T}^{\text{sol}}_{\mathcal{M}}$ is the complexity of solving the system to constant error $c < 1$, then we can use an iterative refinement procedure to solve the system to accuracy ϵ in time $\mathcal{T}^{\text{sol}}_{\mathcal{M}} \cdot \log \frac{1}{\epsilon}$ for any $\epsilon > 0$.

in time

$$O\left(\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]})\cdot\kappa^{1.5}\cdot\frac{\log^6(\frac{mnd\kappa}{\delta\epsilon})}{\epsilon^8}\right).$$

Here $\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]})$ denotes the computational complexity of multiplying an aribtrary vector by *all* matrices in the set $\{\mathbf{V}_i\}_{i\in[n]}$.

Theorem 43. *Given matrices $\{\mathbf{M}_i\}_{i\in[n]}$ with explicit factorizations* (7.9)*, such that* (7.7) *is feasible for some $\kappa \geq 1$ and we can solve linear systems in linear combinations of $\{\mathbf{M}_i\}_{i\in[n]}$ to ϵ relative accuracy in the sense of* (7.10) *in $\mathcal{T}_{\mathcal{M}}^{sol}\cdot\log\frac{1}{\epsilon}$ time, and $\mathbf{B}$ satisfying*

$$\mathbf{B}\preceq\mathcal{M}(\mathbf{1})\preceq\alpha\mathbf{B},\ \mathbf{I}\preceq\mathbf{B}\preceq\beta\mathbf{I}, \tag{7.11}$$

we can return weights $w\in\mathbb{R}^n_{\geq 0}$ satisfying (7.8) *with probability $\geq 1-\delta$ in time*

$$O\left(\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]}\cup\{\mathbf{B}\}\right)+\mathcal{T}_{\mathcal{M}}^{sol}\right)\cdot\kappa^2\cdot\frac{\log^{11}(\frac{mnd\kappa\beta}{\delta\epsilon})}{\epsilon^8}\cdot\log\frac{\alpha}{\epsilon}\right).$$

The first condition in (10.32) is no more general than assuming we have a "warm start" reweighting $w_0\in\mathbb{R}^n_{\geq 0}$ (not necessarily $\mathbf{1}$) satisfying $\mathbf{B}\preceq\sum_{i\in[n]}[w_0]_i\mathbf{M}_i\preceq\alpha\mathbf{B}$, by exploiting scale invariance of the problem and setting $\mathbf{M}_i\leftarrow[w_0]_i\mathbf{M}_i$. The second bound in (10.32) is equivalent to $\kappa(\mathbf{B})\leq\beta$ up to constant factors, since given a bound β, we can use the power method (cf. Fact 40) to shift the scale of $\mathbf{B}$ so it is spectrally larger than $\mathbf{I}$ spectrally. The operation requires just a logarithmic number of matrix vector multiplications with $\mathbf{B}$, which does not impact the runtime in Theorem 43.

Our algorithm for Theorem 42 is based on matrix multiplicative weights (MMW) [534, 34, 33], a popular first-order meta-algorithm for approximately solving SDPs, with carefully chosen gain matrices formed by using packing SDP solvers as a black box. In this sense, it is an efficient reduction from structured mixed positive instances of the form (7.7), (7.8) to pure packing instances. We note similar ideas were previously used in [353] (repurposed in [133]) for solving graph-structured matrix dictionary recovery problems. Our Theorems 42 and 43 improve upon these results both in generality (prior works only handled $\kappa=1+\epsilon$ where ϵ is sufficiently small) and efficiency (our reduction calls a packing solver $\approx\log d$ times for constant ϵ,κ, while [353] used $\approx\log^2 d$ calls). Moreover, our method is a natural application of MMW, and is arguably simpler than prior work. This simplicity is useful in diagonal scaling applications, as it allows us to obtain a tighter characterization of our dependence on κ, the primary quantity of interest.

Our algorithm for proving Theorem 43 is new, and based on combining Theorem 42 with a multi-level iterative preconditioning scheme we refer to as a "homotopy method." In particular, our algorithm for Theorem 42 recursively calls Theorem 42 and preconditioned linear system solvers as black boxes, to provide near-optimal reweightings $\mathcal{M}(w)$ which approximate $\mathbf{B}+\lambda\mathbf{I}$ for various values of λ. We then combine our access to linear system solvers in $\mathcal{M}(w)$ with efficient rational

approximations to various matrix functions, yielding our overall algorithm. This homotopy method framework is reminiscent of techniques used by other recent works in the literature on numerical linear algebra and structured continuous optimization, such as [362, 311, 100, 8].

7.1.4 Schatten packing SDPs

We consider a natural generalization of packing semidefinite programs (SDPs) which we call *Schatten packing*. Given symmetric positive semidefinite $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and parameter $p \geq 1$, a Schatten packing SDP asks to solve the optimization problem

$$\min \left\| \sum_{i \in [n]} w_i \mathbf{A}_i \right\|_p \text{ subject to } w \in \Delta^n. \tag{7.12}$$

Here, $\|\mathbf{M}\|_p$ is the Schatten-p norm of matrix $\mathbf{M}$ and Δ^n is the probability simplex. When $p = \infty$, (7.12) is the well-studied (standard) packing SDP objective [282, 19, 442], which asks to find the most spectrally bounded convex combination of packing matrices. For smaller p, the objective encourages combinations more (spectrally) uniformly distributed over directions.

The specialization of (7.12) to diagonal matrices is a smooth generalization of packing linear programs, previously studied in the context of fair resource allocation [385, 191]. For the ℓ_∞ case of (7.12), packing SDPs have the desirable property of admitting "width-independent" approximation algorithms via exploiting positivity structure. Specifically, width-independent solvers obtain multiplicative approximations with runtimes independent or logarithmically dependent on size parameters of the problem. This is a strengthening of additive notions of approximation typically used for approximate semidefinite programming. Our work gives the first width-independent solver for Schatten packing. We state an informal guarantee here.

Theorem 44. *Let $\{\mathbf{A}_i\}_{i \in [n]} \in \mathbb{S}^d_{\geq 0}$, and $\epsilon > 0$. There is an algorithm taking $O(\frac{p \log(\frac{nd}{\epsilon})}{\epsilon})$ iterations, returning a $1 + \epsilon$ multiplicative approximation to the problem* (7.12)*. For odd p, each iteration can be implemented in time nearly-linear in the number of nonzeros amongst all $\{\mathbf{A}_i\}_{i \in [n]}$.*

We remark that by setting p appropriately, Theorem 44 also results in the state-of-the-art approximate (ℓ_∞) packing SDP solver. It will be used in our development of Theorems 42 and 43.

Previous work

Semidefinite programming (SDP) and its linear programming specialization are fundamental computational tasks, with myriad applications in learning, operations research, and computer science. Though general-purpose polynomial time algorithms exist for SDPs ([424]), in practical settings in high dimensions, approximations depending linearly on input size and polynomially on error ϵ are sometimes desirable. To this end, approximation algorithms based on entropic mirror descent have

been intensely studied [534, 34, 232, 20, 108], obtaining ϵ additive approximations to the objective with runtimes depending polynomially on ρ/ϵ, where ρ is the "width", the largest spectral norm of a constraint.

For structured SDPs, stronger guarantees can be obtained in terms of width. Specifically, several algorithms developed for packing SDPs ((7.12) with $p = \infty$) yield $(1+\epsilon)$-*multiplicative* approximations to the objective, with *logarithmic* dependence on width [282, 442, 19, 285]. As ρ upper bounds objective value in this setting, in the worst case runtimes of width-dependent solvers yielding $\epsilon\rho$-additive approximations have similar dependences as width-independent counterparts. Width-independent solvers simultaneously yield stronger multiplicative bounds at all scales of objective value, making them desirable in suitable applications. In particular, ℓ_∞ packing SDPs have found great utility in robust statistics algorithm design [133, 131, 132, 171].

Beyond ℓ_∞ packing, width-independent guarantees in the SDP literature are few and far between; to our knowledge, other than the covering and mixed solvers of [285], ours is the first such guarantee for a broader family of objectives[5]. Our method complements analogous ℓ_p extensions in the width-dependent setting, e.g. [21], as well as width-independent solvers for ℓ_p packing linear programs [385, 191]. We highlight the fair packing solvers of [385, 191], motivated by problems in equitable resource allocation, which further solved ℓ_p packing variants for $p \notin [1, \infty)$. We find analogous problems in semidefinite settings interesting, and defer to future work.

7.2 A rank-1 sketch for matrix multiplicative weights

We present our main contribution in Section 7.2.1: regret bounds for our rank-1 randomized projections P_u and their proof via the geometry of $\bar{\mathsf{P}}$. In Section 7.2.3 we describe how to compute $\mathbf{X}_t$ in $\widetilde{O}\left(\sqrt{\eta t}\right)$ matrix-vector products using the Lanczos method. In Section 7.2.4 we present in detail the application of our sketching scheme to semidefinite programming, as described above. We conclude the section in Section 7.2.5 by discussing a number of possible extensions of our results along with the challenges they present.

7.2.1 Main result

In this section, we state and prove our main result: regret bounds for a rank-1 sketch of the matrix multiplicative weights method. Let us recall our sketch. At time step t, having observed gain matrices $\mathbf{G}_1, \ldots, \mathbf{G}_{t-1} \in S_n$, we independently draw[6] $u_t \sim \mathsf{Uni}(\mathbb{S}^{n-1})$ and play the rank-1 matrix

$$\mathbf{X}_t := \mathsf{P}_{u_t}\left(\eta \sum_{i=1}^{t-1} \mathbf{G}_i\right), \text{ where } \mathsf{P}_u(\mathbf{Y}) := \frac{e^{\mathbf{Y}/2} u u^T e^{\mathbf{Y}/2}}{u^T e^{\mathbf{Y}} u} = \frac{vv^T}{v^T v} \text{ for } v = e^{\mathbf{Y}/2} u. \tag{7.13}$$

[5] In concurrent and independent work, [136] develops width-independent solvers for Ky-Fan packing objectives, a different notion of generalization than the Schatten packing objectives we consider.

[6] Since P_u is invariant to scaling of u, it has the same distribution for u standard Gaussian or uniform on a sphere.

We call $\mathsf{P}_u : S_n \to \Delta_n$ the *randomized mirror projection*. The key computational consideration is that we can evaluate $\mathsf{P}_u(\mathbf{Y})$ efficiently, while on the analytic side, we show that the update (7.13) defines *on average* an efficient mirror descent procedure. The regret bounds for $\mathbf{X}_t$ then follow.

Expected regret bounds

The focus of our analysis is the *average mirror projection*

$$\bar{\mathsf{P}}(\mathbf{Y}) := \mathbb{E}_u \mathsf{P}_u(\mathbf{Y}) \text{ and action sequence } \bar{\mathbf{X}}_t := \bar{\mathsf{P}}\left(\eta \sum_{i=1}^{t-1} \mathbf{G}_i\right), \tag{7.14}$$

where $\mathbb{E}_u$ denotes expectation w.r.t. to $u \sim \mathsf{Uni}(\mathbb{S}^{n-1})$. As we show in Section 7.2.2 to come, $\bar{\mathsf{P}}$ is the gradient of the function

$$\bar{\mathsf{p}}(\mathbf{Y}) := \mathbb{E}_u \log\left(\left\langle e^{\mathbf{Y}}, uu^T\right\rangle\right) = \mathbb{E}_u \log\left(u^T e^{\mathbf{Y}} u\right),$$

which we also show[7] is a convex spectral function [356]. As a consequence, we can write the average action $\bar{\mathbf{X}}_t$ in the familiar dual averaging [422] or Follow the Regularized Leader [268] form

$$\bar{\mathbf{X}}_t = \mathrm{argmax}_{\mathbf{X} \in \Delta_n} \left\{ \eta \sum_{i=1}^{t-1} \langle \mathbf{G}_i, \mathbf{X} \rangle - \bar{\mathsf{r}}(\mathbf{X}) \right\}$$

where $\bar{\mathsf{r}}(\mathbf{X}) = \sup_{\mathbf{Y} \in S_n} \{\langle \mathbf{Y}, \mathbf{X} \rangle - \bar{\mathsf{p}}(\mathbf{Y})\}$ is the convex conjugate of $\bar{\mathsf{p}}$. In this standard approach, the regularizer $\bar{\mathsf{r}}$ defines the scheme, and regret analysis proceeds by showing that $\bar{\mathsf{r}}$ is strongly convex and has bounded range. The former property is equivalent to the smoothness of $\bar{\mathsf{p}}$.

In contrast, our starting point is the definition (7.14) of the projection $\bar{\mathsf{P}}$, and we find it more convenient to argue about $\bar{\mathsf{P}}$ and $\bar{\mathsf{p}}$ directly. Toward that end, for any $\mathbf{Y}, \mathbf{Y}' \in S_n$ we let

$$\bar{V}_{\mathbf{Y}}(\mathbf{Y}') := \bar{\mathsf{p}}(\mathbf{Y}') - \bar{\mathsf{p}}(\mathbf{Y}) - \left\langle \mathbf{Y}' - \mathbf{Y}, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle \tag{7.15}$$

denote the Bregman divergence induced by $\bar{\mathsf{p}}$. We show that $\bar{V}_{\mathbf{Y}}(\cdot)$ has the properties—analogous to those arising from duality in analyses of dual averaging [422]—necessary to establish our regret bounds.

Proposition 21. *The projection $\bar{\mathsf{P}}$ and divergence $\bar{V}$ satisfy*

1. *Smoothness: for every $\mathbf{Y}, \mathbf{D} \in S_n$, $\bar{V}_{\mathbf{Y}}(\mathbf{Y} + \mathbf{D}) \leq \frac{3}{2} \|\mathbf{D}\|_\infty^2$.*
2. *Refined smoothness for positive shifts: for every $\mathbf{Y}, \mathbf{D} \in S_n$ such that $\mathbf{D} \succeq 0$ and $\|\mathbf{D}\|_\infty \leq \frac{1}{6}$, $\bar{V}_{\mathbf{Y}}(\mathbf{Y} + \mathbf{D}) \leq 3 \|\mathbf{D}\|_\infty \left\langle \mathbf{D}, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle$.*

[7] For *fixed* $u \in \mathbb{R}^n$, however, $\mathsf{P}_u \neq \nabla \log\left(u^T e^{\mathbf{Y}} u\right)$ and we do not know if it is the gradient of any other function. Moreover, $\mathbf{Y} \mapsto \log\left(u^T e^{\mathbf{Y}} u\right)$ is not convex.

3. Diameter bound: for every $\mathbf{Y}, \mathbf{Y}' \in S_n$, $\bar{V}_{\mathbf{Y}}(0) - \bar{V}_{\mathbf{Y}}(\mathbf{Y}') \le \log(4n)$.

4. Surjectivity: for every $\mathbf{X} \in \operatorname{relint} \Delta_n$ *there exists* $\mathbf{Y} \in S_n$ *such that* $\bar{\mathsf{P}}(\mathbf{Y}) = \mathbf{X}$.

We return to Proposition 21 and prove it in Section 7.2.2. The proposition gives the following regret bounds for the averaged actions $\bar{\mathbf{X}}_t$.

Theorem 45. *Let* $\mathbf{G}_1, \dots, \mathbf{G}_T$ be any *sequence of gain matrices in* S_n *and let* $\bar{\mathbf{X}}_t = \bar{\mathsf{P}}(\eta \sum_{i=1}^{t-1} \mathbf{G}_i)$ *as in Eq.* (7.14). *Then, for every* $T \in \mathbb{N}$,

$$\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle \le \frac{\log(4n)}{\eta} + \frac{3\eta}{2} \cdot \sum_{t=1}^{T} \|\mathbf{G}_t\|_\infty^2 . \tag{7.16}$$

If additionally $0 \preceq \mathbf{G}_t \preceq I$ *for every* t *and* $\eta \le \frac{1}{6}$,

$$\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle \le \frac{\log(4n)}{\eta} + 3\eta \cdot \lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right). \tag{7.17}$$

We prove Theorem 45 in Appendix F.1.1. The proof is essentially the standard dual averaging telescoping argument [422], which we perform using only the properties in Proposition 21. Indeed, matrix multiplicative weights satisfies a version of Proposition 21 with slightly smaller constant factors, and its regret bounds follow similarly.

The projection $\bar{\mathsf{P}}$ is no easier to compute than the matrix multiplicative weights projection. However, P_u is easily computed and is unbiased for $\bar{\mathsf{P}}$. Consequently—under Assumption 5—the sketch P_u inherits the regret guarantees in Theorem 45. To argue this formally, we define the σ-fields

$$\mathcal{F}_t := \sigma(\mathbf{G}_1, \mathbf{X}_1, \dots, \mathbf{G}_t \mathbf{X}_1, \mathbf{G}_{t+1}),$$

so that $\mathbf{G}_t \in \mathcal{F}_{t-1}$ and $\bar{\mathbf{X}}_t \in \mathcal{F}_{t-1}$, while, under Assumption 5, $\mathbb{E}[\mathbf{X}_1 \mid \mathcal{F}_{t-1}] = \bar{\mathbf{X}}_t$ because $u_t \sim \mathsf{Uni}(\mathbb{S}^{n-1})$, independent of $\mathcal{F}_{t-1}$. Consequently, we have the following

Corollary 22. *Let* $\mathbf{G}_1, \dots, \mathbf{G}_T$ *be symmetric gain matrices satisfying Assumption 5 and let* $\mathbf{X}_t$ *be generated according to Eq.* (7.13). *Then*

$$\mathbb{E}\left[\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \mathbf{X}_t \right\rangle\right] \le \frac{\log(4n)}{\eta} + \frac{3\eta}{2} \cdot \sum_{t=1}^{T} \mathbb{E}\left[\, \|\mathbf{G}_t\|_\infty^2 \,\right].$$

If additionally $0 \preceq \mathbf{G}_t \preceq I$ *for every* t *and* $\eta \le \frac{1}{6}$,

$$\mathbb{E}\left[\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \mathbf{X}_t \right\rangle\right] \le \frac{\log(4n)}{\eta} + 3\eta \cdot \mathbb{E}\left[\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right)\right].$$

Proof. Using $\mathbf{G}_t \in \mathcal{F}_{t-1}$ and $\mathbb{E}[\mathbf{X}_t \mid \mathcal{F}_{t-1}] = \bar{\mathbf{X}}_t$, we have $\mathbb{E}\langle \mathbf{G}_t, \mathbf{X}_t\rangle = \mathbb{E}\left[\mathbb{E}[\langle \mathbf{G}_t, \mathbf{X}_t\rangle \mid \mathcal{F}_{t-1}]\right] = \mathbb{E}\left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t\right\rangle$, and so the result is immediate from taking expectation in Theorem 45. □

It is instructive to compare these guarantees to those for the full (non-approximate) matrix multiplicative weights algorithm. Let

$$\mathcal{R}[T] := \mathbb{E}\left[\lambda_{\max}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{G}_t\right) - \frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{G}_t, \mathbf{X}_t\rangle\right]$$

denote the expected *average* regret at time T. If $\|\mathbf{G}_t\|_\infty \le 1$ for every t, the bound (7.16) along with Corollary 22 imply, for $\eta = (2\log(4n)/(3T))^{1/2}$,

$$\mathcal{R}[T] \le \sqrt{\frac{6\log(4n)}{T}}, \quad \text{i.e. } \mathcal{R}[T] \le \epsilon \text{ for } T \ge \frac{6\log(4n)}{\epsilon^2}.$$

In contrast, the matrix multiplicative weights procedure (7.2) guarantees average regret below ϵ in $2\log(n)/\epsilon^2$ steps, so our guarantee is worse by a factor of roughly 3.

The bound (7.17) guarantees smaller *relative* average regret when we additionally assume $0 \preceq \mathbf{G}_t \preceq I$ for every t and an a-priori upper bound of the form $\lambda^\star := \lambda_{\max}(\frac{1}{T}\sum_{t=1}^{T}\mathbf{G}_t) \ge \lambda_0$. Here, a judicious choice of η guarantees $\mathcal{R}[T]/\lambda^\star \le \varepsilon$ for $T \ge 12\log(4n)/(\lambda_0\varepsilon^2)$. Again, this is slower than the corresponding guarantee for matrix multiplicative weights by a factor of roughly 3. Relative regret bounds of the form (7.17) are useful in several application of multiplicative weights and its matrix variant [33], e.g. width-independent solvers for linear and positive semidefinite programs [442].

High-probability regret bounds

Using standard martingale convergence arguments [114, 416], we can provide high-probability convergence guarantees for our algorithm. Indeed, we have already observed in Corollary 22 that $\mathbb{E}\left[\langle \mathbf{G}_t, \mathbf{X}_t\rangle \mid \mathcal{F}_{t-1}\right] = \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t\right\rangle$ and therefore $\left\langle \mathbf{G}_t, \mathbf{X}_t - \bar{\mathbf{X}}_t\right\rangle$ is a martingale difference sequence adapted to the filtration $\mathcal{F}_t$. As $|\langle \mathbf{G}_t, \mathbf{X}_t\rangle| \le \|\mathbf{G}_t\|_\infty \|\mathbf{X}_t\|_1 = \|\mathbf{G}_t\|_\infty$, the martingale $\sum_{i=1}^{t}\left\langle \mathbf{G}_i, \mathbf{X}_t i - \bar{\mathbf{X}}_i\right\rangle$ has bounded differences whenever $\|\mathbf{G}_t\|_\infty$ is bounded, so that the next theorem is an immediate consequence of the Azuma-Hoeffding inequality and its multiplicative variant [20]

Corollary 23. *Let $\mathbf{G}_1, \ldots, \mathbf{G}_T$ be symmetric gain matrices satisfying Assumption 5 and let $\mathbf{X}_t$ be generated according to Eq.* (7.13)*. If $\|\mathbf{G}_t\|_\infty \le 1$ for every t, then for every $T \in \mathbb{N}$ and $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right) - \sum_{t=1}^{T}\langle \mathbf{G}_t, \mathbf{X}_t\rangle \le \frac{\log(4n)}{\eta} + \frac{3\eta}{2}T + \sqrt{2T\log\tfrac{1}{\delta}}. \tag{7.18}$$

If additionally $0 \preceq \mathbf{G}_t \preceq I$ *for every* t *and* $\eta \leq \frac{1}{6}$*, then with probability at least* $1-\delta$*,*

$$\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right) - \sum_{t=1}^{T}\langle \mathbf{G}_t, \mathbf{X}_t\rangle \leq \frac{\log(4n/\delta)}{\eta} + 4\eta\,\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right). \tag{7.19}$$

We give the proof of Corollary 23 in Appendix F.1.2.

Our development uses Assumption 5 only through its consequence $\mathbb{E}[\mathbf{X}_t t \mid \mathcal{F}_{t-1}] = \bar{\mathbf{X}}_t$. Therefore, our results apply to any adversary that produces gains with such martingale structure, a weaker requirement than Assumption 5.

7.2.2 Analyzing the average mirror projection

In this section we outline the proof of Proposition 21, which constitutes the core technical contribution of our work. Our general strategy is to relate the average mirror projection to the multiplicative weights projection, which satisfies a version of Proposition 21. Our principal mathematical tool is the theory of convex, twice-differentiable spectral functions [356, 357].

We begin with the vector log-sum-exp, or softmax, function

$$\mathrm{lse}(v) := \log\left(\sum_{j=1}^{n} e^{v_j}\right) \text{ and its gradient } \nabla\mathrm{lse}(v) = \frac{e^v}{\mathbf{1}^T e^v},$$

where we write e^v for $\exp(\cdot)$ applied elementwise to v and $\mathbf{1}$ for the all-ones vector. Note that $\nabla\mathrm{lse} : \mathbb{R}^n \to \sigma_n$ is the mirror projection associated with (vector) multiplicative weights. Let $\mathbf{Y} \in S_n$ have eigen-decomposition $\mathbf{Y} = \mathbf{Q}\,\mathrm{diag}(\lambda)\mathbf{Q}^T$. The matrix softmax function is

$$\mathsf{p}^{\mathrm{mw}}(\mathbf{Y}) := \log\mathrm{tr}\, e^{\mathbf{Y}} = \mathrm{lse}(\lambda) \text{ and } \mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) = \nabla\mathsf{p}^{\mathrm{mw}}(\mathbf{Y}) = \frac{e^{\mathbf{Y}}}{\mathrm{tr}\, e^{\mathbf{Y}}} = Q\,\mathrm{diag}(\nabla\mathrm{lse}(\lambda))Q^T$$

is the matrix multiplicative weights mirror projection.

We now connect the function $\bar{\mathsf{p}}(\mathbf{Y}) = \mathbb{E}_u[\log\mathrm{tr}(e^{\mathbf{Y}}uu^T)]$ and the projection $\bar{\mathsf{P}}(\mathbf{Y}) = \mathbb{E}_u \frac{e^{\mathbf{Y}/2}uu^Te^{\mathbf{Y}/2}}{u^Te^{\mathbf{Y}}u}$ to their counterparts $\mathsf{p}^{\mathrm{mw}}, \mathsf{P}^{\mathrm{mw}}$ and lse.

Lemma 87. *Let* $\mathbf{Y} \in S_n$ *have eigen-decomposition* $\mathbf{Y} = \mathbf{Q}\,\mathrm{diag}(\lambda)\mathbf{Q}^T$*. Let* $w \in \sigma_n$ *be drawn from a* Dirichlet$(\frac{1}{2},\ldots,\frac{1}{2})$ *distribution. Then*

$$\bar{\mathsf{p}}(\mathbf{Y}) = \mathbb{E}_w\left[\mathrm{lse}(\lambda + \log w)\right] = \mathbb{E}_w\mathsf{p}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q}\,\mathrm{diag}(\log w)\mathbf{Q}^T) \tag{7.20}$$

where log *is applied elementwise. The function* $\bar{\mathsf{p}}$ *is convex and its gradient is*

$$\bar{\mathsf{P}}(\mathbf{Y}) = \nabla\bar{\mathsf{p}}(\mathbf{Y}) = \mathbf{Q}\,\mathrm{diag}\left(\mathbb{E}_w[\nabla\mathrm{lse}(\lambda+\log w)]\right)\mathbf{Q}^T = \mathbb{E}_w\mathsf{P}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q}\,\mathrm{diag}(\log w)\mathbf{Q}^T). \tag{7.21}$$

Proof. Let u be uniformly distributed over the unit sphere in $\mathbb{R}^n$ and note that u and $\mathbf{Q}^Tu$ are

identically distributed. Therefore, for $\Lambda = \operatorname{diag}(\lambda)$,

$$\bar{\mathsf{p}}(\mathbf{Y}) = \mathbb{E}_u \log\left(u^T e^{\mathbf{Y}} u\right) = \mathbb{E}_u \log\left((\mathbf{Q}^T u)^T e^{\Lambda} (\mathbf{Q}^T u)\right) = \mathbb{E}_u \log\left(u^T e^{\Lambda} u\right) = \bar{\mathsf{p}}(\Lambda).$$

Further, a vector w with coordinates[8] $w_i = u_i^2$ has a Dirichlet$(\frac{1}{2}, \ldots, \frac{1}{2})$ distribution. Hence,

$$\bar{\mathsf{p}}(\Lambda) = \mathbb{E}_u \log\left(\sum_{i=1}^n u_i^2 e^{\lambda_i}\right) = \mathbb{E}_w \log\left(\sum_{i=1}^n e^{\lambda_i + \log w_i}\right) = \mathbb{E}_w \operatorname{lse}(\lambda + \log w),$$

establishing the identity (7.20).

Evidently, $\bar{\mathsf{p}}(\mathbf{Y})$ is a spectral function—a permutation-invariant function of the eigenvalues of $\mathbf{Y}$. Moreover, since lse is convex, $\lambda \mapsto \mathbb{E}_w \operatorname{lse}(\lambda + \log w)$ is also convex, and Corollary 2.4, [356] shows that $\bar{\mathsf{p}}$ is convex. Moreover, Corollary 3.2, [356] gives

$$\nabla \bar{\mathsf{p}}(\mathbf{Y}) = \mathbf{Q} \operatorname{diag}(\nabla \mathbb{E}_w[\operatorname{lse}(\lambda + \log w)]) \mathbf{Q}^T = \mathbb{E}_w \mathsf{P}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q} \log(w) \mathbf{Q}^T).$$

It remains to show that $\bar{\mathsf{P}}(\mathbf{Y}) = \nabla \bar{\mathsf{p}}(\mathbf{Y})$. Here we again use the rotational symmetry of u to write

$$\bar{\mathsf{P}}(\mathbf{Y}) = \mathbb{E}_u \frac{e^{\mathbf{Y}/2} u u^T e^{\mathbf{Y}/2}}{u^T e^{\mathbf{Y}} u} = \mathbf{Q} \left(\mathbb{E}_u \frac{e^{\Lambda/2} (\mathbf{Q}^T u)(\mathbf{Q}^T u)^T e^{\Lambda/2}}{(\mathbf{Q}^T u)^T e^{\Lambda} (\mathbf{Q}^T u)}\right) \mathbf{Q}^T = \mathbf{Q}\, \bar{\mathsf{P}}(\Lambda) \mathbf{Q}^T.$$

Moreover,

$$\bar{\mathsf{P}}(\Lambda)_{ij} = \mathbb{E}_u \frac{u_i u_j e^{(\lambda_i + \lambda_j)/2}}{\sum_{k=1}^n u_k^2 e^{\lambda_k}} \overset{(\star)}{=} \mathbb{E}_u \frac{u_i^2 e^{\lambda_i} \mathbb{I}_{\{i=j\}}}{\sum_{k=1}^n u_k^2 e^{\lambda_k}} = \mathbb{E}_w \nabla_i \operatorname{lse}(\lambda + \log w) \mathbb{I}_{\{i=j\}}$$

where the equality $(\star)$ above follows because u_i has a symmetric distribution, even conditional on $u_j, j \neq i$, so $\mathbb{E}\left[u_i u_j \mid u_1^2, \ldots, u_n^2, u_j\right] = 0$ for $i \neq j$. □

Lemma 87 is all we need in order to prove parts 3 and 4 of Proposition 21.

Proof. **(Proposition 21, parts 3 and 4)** We first observe the following simple lower bound on $\bar{\mathsf{p}}$, immediate from identity (7.20) in Lemma 87,

$$\bar{\mathsf{p}}(\mathbf{Y}) = \mathbb{E}_w \log\Big(\sum_{i=1}^n e^{\lambda_i(\mathbf{Y}) + \log w_i}\Big) \geq \lambda_{\max}(\mathbf{Y}) + E_{w_1} \log w_1 \geq \lambda_{\max}(\mathbf{Y}) - \log(4n), \tag{7.22}$$

where $\mathbb{E}_{w_1} \log w_1 \geq -\log(4n)$ comes from noting that $w_1 \sim \operatorname{Beta}(\frac{1}{2}, \frac{n-1}{2})$ (see Lemma 286 in Appendix F.1.5). For matrices $\mathbf{Y} \in S_n$ and $\mathbf{X}_t \in \Delta_n$,

$$\langle \mathbf{Y}, \mathbf{X}_t \rangle = \langle \mathbf{Y} - \lambda_{\min}(\mathbf{Y}) I, \mathbf{X}_t \rangle + \lambda_{\min}(\mathbf{Y}) \operatorname{tr} \mathbf{X}_t \leq \|\mathbf{Y} - \lambda_{\min}(\mathbf{Y}) I\|_\infty \|\mathbf{X}_t\|_1 + \lambda_{\min}(\mathbf{Y}) \operatorname{tr} \mathbf{X}_t = \lambda_{\max}(\mathbf{Y}),$$

[8] The letter w naturally denotes a vector of 'weights' in the simplex. Here, it is also double-u.

where the final equality is due to $\|\mathbf{Y} - \lambda_{\min}(\mathbf{Y})\mathbf{I}\|_\infty = \lambda_{\max}(\mathbf{Y}) - \lambda_{\min}(\mathbf{Y})$ for every $\mathbf{Y} \in S_n$ and $\|\mathbf{X}_t\|_1 = \operatorname{tr} \mathbf{X}_t = 1$ for every $\mathbf{X}_t \in \Delta_n$.

$$\langle \mathbf{Y}, \mathbf{X}_t \rangle - \bar{\mathsf{p}}(\mathbf{Y}) \leq \log(4n) \tag{7.23}$$

for every $\mathbf{Y} \in S_n$ and $\mathbf{X}_t \in \Delta_n$. Part 3 follows since

$$\bar{V}_{\mathbf{Y}}(0) - \bar{V}_{\mathbf{Y}}(\mathbf{Y}') = \bar{\mathsf{p}}(0) + \left\langle \mathbf{Y}', \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle - \bar{\mathsf{p}}(\mathbf{Y}') \leq \bar{\mathsf{p}}(0) + \log(4n) = \log(4n),$$

where we used the bound (7.23) with $\mathbf{X}_t = \bar{\mathsf{P}}(\mathbf{Y})$ and the fact that $\bar{\mathsf{p}}(0) = \mathbb{E}_w \log(\mathbf{1}^T w) = 0$.

To show Part 4, let $\bar{\mathsf{r}}(\mathbf{X}_t) := \sup_{\mathbf{Y} \in S_n} \{\langle \mathbf{Y}, \mathbf{X}_t \rangle - \bar{\mathsf{p}}(\mathbf{Y})\}$ be the convex conjugate of $\bar{\mathsf{p}}$. Eq. (7.23) implies that $\bar{\mathsf{r}}(\mathbf{X}_t) < \infty$ for all $\mathbf{X}_t \in \Delta_n$, and therefore $\operatorname{relint} \Delta_n \subseteq \operatorname{relint} \operatorname{dom} \bar{\mathsf{r}}$. Every convex function has nonempty subdifferential on the relative interior of its domain as shown in Theorem X.1.4.2 of [271], and thus for $X \in \operatorname{relint} \Delta_n$ there exists $\mathbf{Y} \in \partial \bar{\mathsf{r}}(\mathbf{X}_t)$. By definition of $\bar{\mathsf{r}}$, any such $\mathbf{Y}$ satisfies $\mathbf{X}_t = \nabla \bar{\mathsf{p}}(\mathbf{Y}) = \bar{\mathsf{P}}(\mathbf{Y})$, as required. □

Proving parts 1 and 2 requires second order information on $\bar{\mathsf{p}}$. For twice differentiable function f, we denote $\nabla^2 f(\mathbf{A})[\mathbf{B}, \mathbf{B}] = \frac{\partial^2}{\partial t^2} f(\mathbf{A} + t\mathbf{B})|_{t=0}$. It is easy to verify that, for every $\lambda, \delta \in \mathbb{R}^n$,

$$\delta^T \nabla^2 \operatorname{lse}(\lambda) \delta = \nabla^2 \operatorname{lse}(\lambda)[\delta, \delta] \leq (\delta^2)^T \nabla \operatorname{lse}(\lambda),$$

where $[\delta^2]_i = \delta_i^2$; this concisely captures the pertinent second order structure of the multiplicative weights mirror projection. [421] shows that this property extends to the matrix case.

For any $\mathbf{Y}, \mathbf{D} \in S_n$, $\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y})[\mathbf{D}, \mathbf{D}] \leq \left\langle \mathbf{D}^2, \mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) \right\rangle$. In Appendix F.1.3 we explain how to find this result in [421], as it is not explicit there. In view of Lemma 87, it is natural to hope that $\nabla^2 \bar{\mathsf{p}}$ and $\nabla^2 \mathsf{p}^{\mathrm{mw}}$ are also related via simple expectation. Unfortunately, this fails; we can, however, derive a bound.

For any $\mathbf{Y}, \mathbf{D} \in S_n$, orthogonal eigenbasis $\mathbf{Q}$ for Y, and $w \sim \operatorname{Dirichlet}(\frac{1}{2}, \ldots, \frac{1}{2})$,

$$\nabla^2 \bar{\mathsf{p}}(\mathbf{Y})[\mathbf{D}, \mathbf{D}] \leq 3 \cdot \mathbb{E}_w \nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q} \operatorname{diag}(\log w) \mathbf{Q}^T)[\mathbf{D}, \mathbf{D}] \tag{7.24}$$

$$\leq 3 \left\langle \mathbf{D}^2, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle. \tag{7.25}$$

Our proof of Lemma 7.2.2 is technical; we sketch it here briefly and give it in full Appendix F.1.4. The key ingredient in the proof is a formula for the Hessian of spectral functions [357]. Using the spectral characterization (7.20), the formula gives that

$$\nabla^2 \bar{\mathsf{p}}(\mathbf{Y})[\mathbf{D}, \mathbf{D}] = \operatorname{diag}(\tilde{\mathbf{D}})^T \left[\mathbb{E}_w \nabla^2 \operatorname{lse}(\lambda + \log w)\right] \operatorname{diag}(\tilde{\mathbf{D}}) + \left\langle \mathbb{E}_w A^w(\lambda), \tilde{\mathbf{D}} \circ \tilde{\mathbf{D}} \right\rangle.$$

where $\tilde{\mathbf{D}} = \mathbf{Q}^T \mathbf{D} \mathbf{Q}$, $\operatorname{diag}(\tilde{\mathbf{D}}) \in \mathbb{R}^n$ is the vector containing the diagonal entries of $\tilde{\mathbf{D}}$, $\mathbf{A} \circ \mathbf{B}$ denotes

elementwise multiplication of $\mathbf{A}$ and $\mathbf{B}$, and $\mathbf{A}^w_{ij}(\lambda) := \frac{\nabla_i \mathrm{lse}(\lambda+\log(w)) - \nabla_j \mathrm{lse}(\lambda+\log(w))}{\lambda_i - \lambda_j}\mathbb{I}_{\{i\neq j\}}$. With the shorthand $\mathbf{Y}_{\{w\}} := \mathbf{Y} + \mathbf{Q}\,\mathrm{diag}(\log w)\mathbf{Q}^T$, we use the formula of [357] again to express $\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})$ as

$$\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})[\mathbf{D},\mathbf{D}] = \mathrm{diag}(\tilde{\mathbf{D}})^T \left[\nabla^2 \mathrm{lse}(\lambda + \log w)\right] \mathrm{diag}(\tilde{\mathbf{D}}) + \left\langle \mathbf{A}^{\mathbf{1}}(\lambda + \log w), \tilde{\mathbf{D}} \circ \tilde{\mathbf{D}} \right\rangle,$$

where $\mathbf{A}^{\mathbf{1}} = \mathbf{A}^{\tilde{w}}$ evaluated at $\tilde{w} = \mathbf{1}$. The bulk of the proof is dedicated to establishing the entry-wise bounds

$$\mathbb{E}_w \mathbf{A}^w_{ij}(\lambda) \leq \mathbb{E}_w \left[\left(1 + \frac{\tanh\left(\frac{\lambda_i - \lambda_j}{2}\right) \left| \log \frac{w_i}{w_j} \right|}{\lambda_i - \lambda_j} \right) \mathbf{A}^{\mathbf{1}}_{ij}(\lambda + \log w) \right] \leq 3 \cdot \mathbb{E}_w \mathbf{A}^{\mathbf{1}}_{ij}(\lambda + \log w).$$

The first inequality follows from pointwise analysis of a symmetrized version of $\mathbf{A}^w_{ij}$. The second inequality follows from piecewise monotonicity of $\mathbf{A}^w_{ij}$ as a function of $\log \frac{w_i}{w_j} \sim \mathrm{logit}\,\mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$, combined with tight exponential tail bounds for the latter. Substituting the bound on $\mathbb{E}_w \mathbf{A}^w_{ij}(\lambda)$ into the expression for $\nabla^2 \bar{\mathsf{p}}(\mathbf{Y})$ and comparing with $\mathbb{E}_w \nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})$ yields the desired result (7.24). Applying Lemma 7.2.2 and recalling the identity (7.21) yields

$$\mathbb{E}_w \nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q}\ \mathrm{diag}(\log w)\mathbf{Q}^T)[\mathbf{D},\mathbf{D}] \leq \left\langle \mathbf{D}^2, \mathbb{E}_w \mathsf{P}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q}\,\mathrm{diag}(\log w)\mathbf{Q}^T) \right\rangle = \left\langle \mathbf{D}^2, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle,$$

establishing the final bound (7.25).

The bound (7.25) gives the remaining parts of Proposition 21.

Proof. **(Proposition 21, parts 1 and 2)** Fix $\mathbf{Y}, \mathbf{D} \in S_n$ and let $p(t) := \bar{\mathsf{p}}(\mathbf{Y} + t\mathbf{D})$. The Bregman divergence (7.15) admits the integral form

$$\begin{aligned} \bar{V}_{\mathbf{Y}}(\mathbf{Y} + \mathbf{D}) = p(1) - p(0) - p'(0) &= \int_0^1 (p'(t) - p'(0))dt = \int_0^1 \int_0^t p''(\tau) d\tau dt \\ &= \int_0^1 \int_0^t \nabla^2 \bar{\mathsf{p}}(\mathbf{Y} + \tau\mathbf{D})[\mathbf{D},\mathbf{D}] d\tau dt. \end{aligned} \tag{7.26}$$

Note that since $\bar{\mathsf{P}}(\mathbf{Y}) \in \Delta_n$ for every $\mathbf{Y} \in S_n$, $\left\langle \mathbf{D}^2, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle \leq \|\mathbf{D}^2\|_\infty \|\bar{\mathsf{P}}(\mathbf{Y})\|_1 = \|\mathbf{D}\|^2_\infty$. Therefore, the bound (7.25) gives

$$\nabla^2 \bar{\mathsf{p}}(\mathbf{Y} + \tau\mathbf{D})[\mathbf{D},\mathbf{D}] \leq 3\,\|\mathbf{D}\|^2_\infty.$$

Substituting back into (7.26) and using $\int_0^1 \int_0^t d\tau dt = \frac{1}{2}$ gives Proposition 21.1.

When $\mathbf{D} \succeq 0$, we have

$$\left\langle \mathbf{D}^2, \bar{\mathsf{P}}(\mathbf{Y}) \right\rangle = \left\langle \mathbf{D}, \mathbf{D}^{1/2} \bar{\mathsf{P}}(\mathbf{Y}) \mathbf{D}^{1/2} \right\rangle \leq \|\mathbf{D}\|_\infty \|\mathbf{D}^{1/2} \bar{\mathsf{P}}(\mathbf{Y}) \mathbf{D}^{1/2}\|_1 = \|\mathbf{D}\|_\infty \left\langle \mathbf{D}, \nabla \bar{\mathsf{p}}(\mathbf{Y}) \right\rangle.$$

Plugging the bound above into the bound (7.25) and substituting back into (7.26) gives

$$\bar{V}_{\mathbf{Y}}(\mathbf{Y}+\mathbf{D}) \leq 3\left\|\mathbf{D}\right\|_{\infty} \int_0^1 \int_0^t \langle \mathbf{D}, \nabla \bar{\mathsf{p}}(\mathbf{Y}+\tau\mathbf{D})\rangle d\tau dt. \tag{7.27}$$

Moreover,

$$\int_0^t \langle \mathbf{D}, \nabla \bar{\mathsf{p}}(\mathbf{Y}+\tau\mathbf{D})\rangle d\tau = \int_0^t p'(\tau)d\tau = p(t) - p(0) = \bar{V}_{\mathbf{Y}}(\mathbf{Y}+t\mathbf{D}) + \left\langle t\mathbf{D}, \bar{\mathsf{P}}(\mathbf{Y})\right\rangle, \tag{7.28}$$

where the final equality uses the definition (7.15) of the Bregman divergence. Note also that $v(t) := \bar{V}_{\mathbf{Y}}(\mathbf{Y}+t\mathbf{D})$ is increasing for $t \geq 0$ due to convexity of $\bar{\mathsf{p}}$; $tv'(t) = \langle t\mathbf{D}, \nabla\bar{\mathsf{p}}(\mathbf{Y}+t\mathbf{D}) - \nabla\bar{\mathsf{p}}(\mathbf{Y})\rangle \geq 0$. Therefore, the equality (7.28) implies $\int_0^t \langle \mathbf{D}, \nabla\bar{\mathsf{p}}(\mathbf{Y}+\tau\mathbf{D})\rangle d\tau \leq \bar{V}_{\mathbf{Y}}(\mathbf{Y}+\mathbf{D}) + t\cdot\left\langle \mathbf{D}, \bar{\mathsf{P}}(\mathbf{Y})\right\rangle$ for every $0 \leq t \leq 1$. Substituting this back into (7.27) and rearranging gives

$$\left(1 - 3\left\|\mathbf{D}\right\|_{\infty}\right)\bar{V}_{\mathbf{Y}}(\mathbf{Y}+\mathbf{D}) \leq \frac{3}{2}\left\|\mathbf{D}\right\|_{\infty}\left\langle \mathbf{D}, \bar{\mathsf{P}}(\mathbf{Y})\right\rangle.$$

establishing part 2 of the proposition, as $1 - 3\left\|\mathbf{D}\right\|_{\infty} \geq \frac{1}{2}$ by assumption. □

7.2.3 Efficient computation of matrix exponential-vector products

The main burden in computing the randomized mirror projections (7.13) lies in computing $e^{\mathbf{A}}b$ for $\mathbf{A} \in S_n$ and $b \in \mathbb{R}^n$. Matrix exponential-vector products have widespread use in solutions of differential equations [465, 272], and also appear as core components in a number of theoretical algorithms [34, 429, 291]. Following a large body of literature [399], we approximate $e^{\mathbf{A}}b$ via the classic Lanczos method [339], an iterative process for computing $f(\mathbf{A})b$ for general real functions f applied to matrix $\mathbf{A}$. The Lanczos approximation enjoys strong convergence guarantees upon which we base our analysis [466]. It is also eminently practical: the only tunable parameter is the number of iterations, and each iteration accesses $\mathbf{A}$ via a single matrix-vector product.

Let $\widetilde{\exp}_k(\mathbf{A}, b)$ be the result of k iterations of the Lanczos method for approximating $e^{\mathbf{A}}b$. We provide a precise description of the method in Appendix F.1.6. Let

$$\tilde{\mathbf{X}}_{t;k} = \widetilde{\mathsf{P}}_{u_t;k}\left(\eta\sum_{i=1}^{t-1}\mathbf{G}_i\right), \quad \text{where } \widetilde{\mathsf{P}}_{u;k}(\mathbf{Y}) = \frac{vv^T}{v^Tv} \text{ for } v = \widetilde{\exp}_k(\mathbf{Y}/2, u) \tag{7.29}$$

denote the *approximate* randomized mirror projection. Using the Lanczos method to compute full eigen-decompositions has well-documented numerical stability issues [392]. In contrast, the approximation (7.29) appears to be numerically stable. To provide a theoretical basis for this observation, we exhibit error bounds under finite floating point precision, leveraging the results of [405], which in turn build on [199, 200]. To account for computational cost, we denote by $\mathrm{mv}(\mathbf{Y})$ the cost of multiplying matrix $\mathbf{Y}$ by any vector.

Proposition 22. *Let $\epsilon, \delta \in (0,1)$ and $\mathbf{Y} \in S_n$, and set $M := \max\{\|\mathbf{A}\|_{\mathrm{op}}, \log(\frac{n}{\epsilon\delta}), 1\}$. Let u be uniformly distributed on the unit sphere in $\mathbb{R}^n$ and independent of $\mathbf{Y}$. If the number of Lanczos iterations k satisfies $k \geq \Theta(1)\sqrt{M \log(\frac{nM}{\epsilon\delta})}$ then the approximation (7.29) satisfies*

$$\|\mathsf{P}_u(\mathbf{Y}) - \widetilde{\mathsf{P}}_{u;k}(\mathbf{Y})\|_1 \leq \epsilon \text{ with probability } \geq 1-\delta \text{ over } u \sim \mathsf{Uni}(\mathbb{S}^{n-1})$$

when implemented using floating point operations with $\mathbf{B} = \Theta(1)\log\frac{nM}{\epsilon\delta}$ bits of precision. The time to compute $\widetilde{\mathsf{P}}_{u;k}(\mathbf{Y})$ is $O(\mathrm{mv}(\mathbf{Y})k + k^2\mathbf{B})$.

We prove Proposition 22 in Appendix F.1.6 and describe here the main ingredients in the proof. First, we show by calculation that

$$\|\mathsf{P}_u(\mathbf{Y}) - \widetilde{\mathsf{P}}_{u;k}(\mathbf{Y})\|_1 \leq \sqrt{8}\frac{\left\|e^{\mathbf{Y}/2}u - \widetilde{\exp}_k(\mathbf{Y}/2, u)\right\|_2}{\left\|e^{\mathbf{Y}/2}u\right\|_2}.$$

Therefore, a multiplicative error guarantee for $\widetilde{\exp}_k$ would imply our result. Unfortunately, for such a guarantee to hold for *all* vectors u we must have $k = \Omega(\|\mathbf{Y}\|_\infty)$ [429]. We circumvent that by using the randomness of u to argue that w.h.p. $\|e^{\mathbf{Y}/2}u\|_2 \gtrsim \frac{1}{\sqrt{n}}e^{\lambda_{\max}(\mathbf{Y}/2)}\|u\|_2$. This allows us to use existing additive error guarantees for $\widetilde{\exp}_k$ to obtain our result.

We connect the approximation to regret in the following corollary.

Corollary 24. *Let $\mathbf{G}_1, \ldots, \mathbf{G}_T$ be symmetric gain matrices satisfying $\|\mathbf{G}_t\|_\infty \leq 1$ for every t. There exists a numerical constant $k_0 < \infty$, such that for every $T \in \mathbb{N}$ and $\delta \in (0,1)$, $\tilde{\mathbf{X}}_{t;k_t}$ defined in (7.29) with $k_t = \lceil k_0(\sqrt{1+\eta t})\log(\frac{nT}{\delta})\rceil$, and $\mathbf{X}_t$ defined in (7.13) satisfy*

$$\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \tilde{\mathbf{X}}_{t;k_t}\right\rangle \geq -1 + \sum_{t=1}^{T}\langle \mathbf{G}_t, \mathbf{X}_t\rangle \quad w.p. \geq 1-\delta/2. \tag{7.30}$$

Let $\epsilon \in (0,1]$, $T = \frac{16\log(4en/\delta)}{\epsilon^2}$ and $\eta = \sqrt{\frac{2\log(4en)}{3T}}$. If Assumption 5 holds with respect to the actions $\tilde{\mathbf{X}}_{t;k_t}$, then with probability at least $1-\delta$, $\frac{1}{T}\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right) - \frac{1}{T}\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \tilde{\mathbf{X}}_{t;k_t}\right\rangle \leq \epsilon$. Computing the actions $\tilde{\mathbf{X}}_{1;k_1}, \ldots, \tilde{\mathbf{X}}_{T;k_T}$ requires $O(\epsilon^{-2.5}\log^{2.5}(\frac{n}{\epsilon\delta}))$ matrix-vector products.

Finally, as we discuss in detail in Appendix F.1.6, computing matrix exponential-vector products (and hence P_u) reduces to solving $\widetilde{O}(1)$ linear systems. Since [20] propose to compute their sketch using a similar reduction, the running time guarantees they establish for their sketch are also valid for ours.

7.2.4 Application to semidefinite programming

Here we describe how to use our rank-1 sketch to solve semidefinite programs (SDPs). The standard SDP formulation is, given $\tilde{\mathbf{C}}, \tilde{\mathbf{A}}_1, \ldots, \tilde{\mathbf{A}}_{\tilde{m}} \in S_{\tilde{n}}$ and $\tilde{b} \in \mathbb{R}^{\tilde{m}}$,

$$\underset{\mathbf{Z} \succeq 0}{\text{minimize}} \left\langle \tilde{\mathbf{C}}, \mathbf{Z} \right\rangle \text{ subject to } \left\langle \tilde{\mathbf{A}}_i, \mathbf{Z} \right\rangle = \tilde{b}_i \quad \forall i \in [\tilde{m}].$$

A binary search over the optimum value reduces this problem to a sequence of feasibility problems. When the constraints imply $\operatorname{tr} \mathbf{Z} \leq r$ for some $r < \infty$, every intermediate feasibility problem is equivalent to deciding whether there exists $\mathbf{X}$ in the spectrahedron Δ_n s.t. $\langle \mathbf{A}_i, \mathbf{X} \rangle \leq 0$ for all $i \in [m]$, with n, m and $\mathbf{A}_i \in S_n$ constructed from $\tilde{n}, \tilde{m}, \tilde{\mathbf{A}}_i, \tilde{b}, \tilde{\mathbf{C}}$ and r. This decision problem is in turn equivalent [231] to determining the sign of

$$\mathfrak{s} = \min_{y \in \sigma_m} \max_{\mathbf{X} \in \Delta_n} \left\langle \mathcal{A}^\star y, \mathbf{X} \right\rangle, \text{ where } \mathcal{A}^\star y := \sum_{i \in [m]} [y]_i \mathbf{A}_i. \tag{7.31}$$

and σ_m is the simplex in $\mathbb{R}^m$. For every $y \in \sigma_m$ and $\mathbf{X} \in \Delta_n$, we have that

$$\min_{y' \in \sigma_m} \left\langle \mathcal{A}^\star y', \mathbf{X} \right\rangle \leq \mathfrak{s} \leq \max_{\mathbf{X}' \in \Delta_n} \left\langle \mathcal{A}^\star y, \mathbf{X}' \right\rangle.$$

Therefore, to determine $\mathfrak{s}$ to additive error ϵ, it suffices to find $y, \mathbf{X}$ with $\operatorname{Gap}(\mathbf{X}, y) \leq \epsilon$, where

$$\operatorname{Gap}(\mathbf{X}, y) := \max_{\mathbf{X}' \in \Delta_n} \left\langle \mathcal{A}^\star y, \mathbf{X}' \right\rangle - \min_{y' \in \sigma_m} \left\langle \mathcal{A}^\star y', \mathbf{X} \right\rangle = \lambda_{\max} \left(\mathcal{A}^\star y \right) - \min_{i \in [m]} \left\langle \mathbf{A}_i, \mathbf{X} \right\rangle. \tag{7.32}$$

A basic approach to solving convex-concave games such as (7.31) is to apply online learning for $\mathbf{X}$ and y simultaneously, where at each round the gains/costs to the max/min player are determined by the actions of the opposite player in the previous round. Importantly, such dynamics satisfy Assumption 5, and we use our rank-1 sketch as the online learning strategy of the (matrix) max player, and standard multiplicative weights for the (vector) min player. Algorithm 24 describes the resulting scheme. The algorithm entertains a convergence guarantee that depends on the *width parameter*

$$\omega := \max_{i \in [m]} \left\| \mathbf{X}_i \right\|_\infty$$

and has the following form.

Theorem 46. *Let $\{\mathbf{X}_t, y_t\}_{t=1}^T$ be the actions produced by Algorithm 1 and, define $\mathbf{X}_T^{\text{avg}} = \frac{1}{T} \sum_{t=1}^T \mathbf{X}_t$, $y_T^{\text{avg}} = \frac{1}{T} \sum_{t=1}^T y_t$. Then*

$$\mathbb{E} \left[\operatorname{Gap} \left(\mathbf{X}_T^{\text{avg}}, y_T^{\text{avg}} \right) \right] \leq \frac{\log(4mn)}{\eta T} + 2\eta\omega^2.$$

Proof. Recalling the definition (7.32) of the duality gap, and that $\mathbf{G}_t = \mathcal{A}^\star y_t$ and $[c_t]_i = \langle \mathbf{A}_i, \mathbf{X}_t \rangle$,

Algorithm 24: Primal-dual SDP feasibility

Let $\mathbf{G}_0 := 0$ and $c_0 := 0$
for $t = 1, \ldots, T$ **do**
  Sample vector u_t uniformly at random from the unit sphere
  Play matrix $\mathbf{X}_t := \mathsf{P}_{u_t}\big(\sum_{i=1}^{t-1} \eta \mathbf{G}_i\big)$
  Play vector $y_t := \nabla \mathrm{lse}\big(-\eta \sum_{i=1}^{t-1} c_i\big) = \frac{y_{t-1} \circ e^{-\eta c_{t-1}}}{\mathbf{1}^T (y_{t-1} \circ e^{-\eta c_{t-1}})}$.
  Form gain matrix $\mathbf{G}_t = \mathcal{A}^\star y_t = \sum_{i \in [m]} [y_t]_i \mathbf{A}_i$
  Form cost vector $[c_t]_i := \langle \mathbf{X}_t, \mathbf{A}_i \rangle$, $i \in [m]$

we have

$$\mathrm{Gap}(\mathbf{X}_T^{\mathrm{avg}}, y_T^{\mathrm{avg}}) = \frac{1}{T} \lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \frac{1}{T} \min_{i \in [m]} \left\{ \sum_{t=1}^{T} [c_t]_i \right\}.$$

Note that $y_t = \nabla \mathrm{lse}(-\eta \sum_{i=1}^{t-1} c_i)$ is a function of $\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}$. Therefore, $\mathbf{G}_t = \mathcal{A}^\star y_t$ satisfies Assumption 5 and we may use Corollary 22 to write

$$\mathbb{E}\left[\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \right] \leq \frac{\log(4n)}{\eta} + \frac{3\eta}{2} \cdot \sum_{t=1}^{T} \mathbb{E}\big[\|\mathbf{G}_t\|_\infty^2 \big] \leq \frac{\log(4n)}{\eta} + \frac{3\eta\omega^2 T}{2},$$

where in the second inequality we used $\omega = \max_{i \in [m]} \|\mathbf{A}_i\|_\infty$ and $y \in \sigma_m$ to bound $\|\mathbf{G}_t\|_\infty = \|\mathcal{A}^\star y_t\|_\infty \leq \omega \cdot \mathbf{1}^T y_t = \omega$. Similarly, we use the standard multiplicative weights regret bound [474] to write

$$\sum_{t=1}^{T} c_t^T y_t - \min_{i \in [m]} \left\{ \sum_{t=1}^{T} [c_t]_i \right\} \leq \frac{\log(m)}{\eta} + \frac{\eta}{2} \cdot \sum_{t=1}^{T} \|c_t\|_\infty^2 \leq \frac{\log(m)}{\eta} + \frac{\eta\omega^2 T}{2},$$

where the second inequality again follows from $|[c_t]_i| = |\langle \mathbf{A}_i, X_t \rangle| \leq \|\mathbf{A}_i\|_\infty \leq \omega$ since $\mathbf{X}_t \in \Delta_n$.

Finally,

$$c_t^T y_t = \sum_{i=1}^{m} [y_t]_i \langle \mathbf{A}_i, \mathbf{X}_t \rangle = \langle \mathcal{A}^\star y_t, \mathbf{X}_t \rangle = \langle \mathbf{G}_t, \mathbf{X}_t \rangle .$$

Hence, summing the two regret bounds and dividing by T gives the result. □

For $\eta = \log(4mn)/\sqrt{2\omega^2 T}$ and $T = 8 \log(4mn)\omega^2/\epsilon^2$, Theorem 46 guarantees $\mathbb{E}\left[\mathrm{Gap}(\mathbf{X}_T^{\mathrm{avg}}, y_T^{\mathrm{avg}})\right] \leq \epsilon$. A high-probability version of this guarantee follows readily via Corollary 23.

Let us now discuss the computational cost of Algorithm 24. Let $\mathrm{mv}(\mathbf{M})$ denote the time required to multiply the matrix $\mathbf{M}$ by any vector, and let $\mathrm{mv}(\mathcal{A}) := \sum_{i \in [m]} \mathrm{mv}(\mathbf{A}_i)$. Except for the computation of $\mathbf{X}_t$, every step in the for loop in Algorithm 24 takes $O(\mathrm{mv}(\mathcal{A}))$ work to execute (we may assume $\mathrm{mv}(\mathcal{A}) \geq \max\{n, m\}$ without loss of generality). Let $\mathbf{Y}_t = \eta \sum_{i=1}^{t-1} \mathbf{G}_i = \mathcal{A}^\star(\sum_{i=1}^{t-1} \eta y_i)$, and note that, with the values of η and T above, $\|\mathbf{Y}_t\|_\infty \leq \eta T \omega = \widetilde{O}(\omega/\epsilon)$ for every $t \leq T$.

Per Section 7.2.3, the computation of $\mathbf{X}_t$ costs $\widetilde{O}\left(\|\mathbf{Y}_t\|_\infty^{0.5}\ \mathrm{mv}(\mathbf{Y}_t)\right) = \widetilde{O}\left((\omega/\epsilon)^{0.5}\,\mathrm{mv}(\mathbf{Y}_t)\right)$. Writing $\mathrm{mv}(\mathcal{A}^\star) := \max_{\alpha\in\mathbb{R}^m}\{\mathrm{mv}(\mathcal{A}^\star\alpha)\} \leq \min\{\mathrm{mv}(\mathcal{A}), n^2\}$, the total computational cost of our algorithm is

$$\widetilde{O}\left(\left[(\omega/\epsilon)^{0.5}\,\mathrm{mv}(\mathcal{A}^\star) + \mathrm{mv}(\mathcal{A})\right]T\right) = \widetilde{O}\left((\omega/\epsilon)^{2.5}\,\mathrm{mv}(\mathcal{A}^\star) + (\omega/\epsilon)^2\,\mathrm{mv}(\mathcal{A})\right).$$

In many settings of interest—namely when the $\mathbf{A}_i$s have mostly non-overlapping sparsity patterns and yet the $\mathbf{Y}_t$s are sparse—we have $\mathrm{mv}(\mathcal{A}^\star) \approx \mathrm{mv}(\mathcal{A})$, so that the computational cost is dominated by the first term.

Comparison with other algorithms

Let $\mathrm{nnz}(\mathbf{M})$ denote the number of nonzero entries of matrix M, and let $\mathrm{nnz}(\mathcal{A}) := \sum_{i\in[m]}\mathrm{nnz}(\mathbf{A}_i) \geq \mathrm{mv}(\mathcal{A})$. If in Algorithm 24 we replace the randomized projection P_u with the matrix multiplicative weights projection P^{mw}, the regret bound of Theorem 46 still holds, but the overall computational cost becomes $\widetilde{O}\left((\omega/\epsilon)^2\left(n^3 + \mathrm{nnz}(\mathcal{A})\right)\right)$ due to full matrix exponentiation. [415] accelerates this scheme using extra-gradient steps, guaranteeing duality gap below ϵ in $\widetilde{O}\left(\omega/\epsilon\right)$ iterations, with each iteration involving two full matrix exponential computations. The overall computational cost of such scheme is consequently $\widetilde{O}\left((\omega/\epsilon)\left(n^3 + \mathrm{nnz}(\mathcal{A})\right)\right)$. [421] attains the same rate by using accelerated gradient descent on a smoothed version of the dual problem. Our scheme improves on this rate for sufficiently sparse problems, with $n^3/\mathrm{nnz}(\mathcal{A}) \gg (\omega/\epsilon)^{-1.5}$.

[163] applies a subgradient method to the dual problem, approximating the subgradients using the Lanczos method to compute a leading eigenvector of $\mathcal{A}^\star y$. The method solves the dual problem to accuracy ϵ with total work $\widetilde{O}\left((\omega/\epsilon)^{2.5}\,\mathrm{mv}(\mathcal{A}^\star) + (\omega/\epsilon)^2\,\mathrm{mv}(\mathcal{A})\right)$, essentially the same as us. However, it is not clear how to efficiently recover a primal solution from this method. Moreover, the surrogate duality gap [163] proposes will not always be 0 at the global optimum, whereas with our approach the true duality gap is readily computable.

[50] replace the full matrix exponentiation in the accelerated scheme of [415] with a rank-k sketch of the form (7.3), where $k = \widetilde{O}\left(\omega/\epsilon\right)$. Similarly to [415], they require $\widetilde{O}\left(\omega/\epsilon\right)$ iterations to attain duality gap below ϵ. [50] approximate matrix exponential vector products by truncating a Taylor series, costing $\widetilde{O}\left(k(\omega/\epsilon)\,\mathrm{mv}(\mathcal{A}^\star)\right) = \widetilde{O}\left((\omega/\epsilon)^2\,\mathrm{mv}(\mathcal{A}^\star)\right)$ work per iteration. With the Lanczos method, the cost improves to $\widetilde{O}\left((\omega/\epsilon)^{1.5}\,\mathrm{mv}(\mathcal{A}^\star)\right)$ work per iteration. Every step of their method also computes $\langle\mathbf{A}_i, \mathbf{X}\rangle$ for all $i \in [m]$ and a rank-k matrix $\mathbf{X} = \sum_{j=1}^k v_j v_j^T$; this costs either $k\cdot\mathrm{mv}(\mathcal{A})$ work (computing $\langle\mathbf{A}_i, v_j\rangle$ for every i, j) or $\mathrm{nnz}(\mathcal{A}) + n^2k$ (when forming $\mathbf{X}$ explicitly). The former option yields total complexity identical to our method. The latter option is preferable only when $\mathrm{nnz}(\mathcal{A}) \gg n^2 \geq \mathrm{mv}(\mathcal{A}^\star)$, and can result in an improvement over the running time of our method if $\mathrm{mv}(\mathcal{A}^\star) \ll \mathrm{nnz}(\mathcal{A})\,(\omega/\epsilon)^{-1.5} + n^2\,(\omega/\epsilon)^{-0.5}$. [50] report that $k = 1$ often gave the best result in their experiment, which is not predicted by their theory. A hypothetical explanation for this finding is that, with $k = 1$, they are essentially running Algorithm 24.

Finally, [163] and [231] propose sub-sampling based algorithms for approximate SDP feasibility

with runtimes potentially sublinear in $\mathrm{mv}(\mathcal{A}^\star)$. However, because of their significantly worse dependence on ω/ϵ, as well as dependence on Frobenius norms, we match or improve upon their runtime guarantees in a variety of settings; see [231] for a detailed comparison.

7.2.5 Discussion

We conclude with a discussion of a number of additional settings where our sketch—or some variation thereof—might be beneficial. In the first two settings we discuss, the naturally arising online learning problem involves adversaries that violate Assumption 5, demonstrating a limitation of our analysis.

Online convex optimization. In the online convex optimization problem, at every time step t the adversary provides a convex loss ℓ_t, the players pays a cost $\ell_t(\mathbf{X}_t)$ and wishes to minimize the regret $\sum_{t=1}^T \ell_t(\mathbf{X}_t) - \min_{\mathbf{X}} \sum_{t=1}^T \ell_t(\mathbf{X})$. The standard reduction to the online learning problem is to construct an adversary with gains $\mathbf{G}_t = -\nabla \ell_t(\mathbf{X}_t)$. However, even if the losses ℓ_t follow Assumption 5, the constructed gains $\mathbf{G}_t$ clearly violate it. Therefore, extensions of our results to online convex optimization will require additional work and probably depend on finer problem structure.

Positive semidefinite programming. [442] and [19] propose algorithms for solving positive (packing/covering) semidefinite programs with width independent running time, meaning that the computational cost of solving the problems to ϵ multiplicative error depends only logarithmically on the width parameter (ω in Section 7.2.4). Both algorithms rely on matrix exponentiation, which they approximate with a rank $\widetilde{O}\left(\epsilon^{-2}\right)$ sketch using the Johnson-Lindenstrauss lemma. The algorithm of [442] uses matrix multiplicative weights in essentially a black-box fashion, so one could hope to replace their high-rank sketch with our rank-1 technique. Unfortunately, the gain matrices that they construct violate Assumption 5 and so our results do not immediately apply. A rank-1 sketch for this setting remains an intriguing open problem.

Improved computational efficiency against an oblivious adversary. An oblivious adversary produces gain matrices $\mathbf{G}_1, \ldots, \mathbf{G}_T$ independent of the actions $\mathbf{X}_1, \ldots, \mathbf{X}_T$; this is a stronger version of Assumption 5. For such an adversary, if we draw $u \sim \mathsf{Uni}(\mathbb{S}^{n-1})$ and set $u_1 = u_2 = \cdots = u_T = u$, the average regret guarantee of Corollary 22 still applies, as [20] explain. In this setting, it may be possible to make the computation of $\mathbf{X}_t$ more efficient by reusing $\mathbf{X}_{t-1}$. Such savings exist in the stochastic setting (when $\mathbf{G}_t$ are i.i.d.) via Oja's algorithm [20], and would be interesting to extend to the oblivious setting.

Online k eigenvectors.[427] show that a variant of matrix multiplicative weights is also capable of learning online the top k-dimensional eigenspace, with similar regret guarantees. As our rank-1 sketch solves the $k = 1$ leading eigenvector problem, it is interesting to study whether a rank-k sketch solves the k leading eigenvectors problem.

7.3 Ky Fan matrix multiplicative weights

We give a regret guarantee for a Ky Fan matrix multiplicative weights procedure, as well as its efficient implementation. We first state a general-purpose regret bound in Section 7.3.1, using a key divergence bound shown in Section 7.3.2. We then show how to use a more fine-grained analysis of simultaneous power iteration developed in Section 7.3.3 to prove correctness and a complexity bound on our overall method (tolerant to approximation error), given in Section 7.3.4.

Throughout this entire section, all variables (unless otherwise specified) will be either d-dimensional vectors or $d \times d$ matrices, and we let $k \in [d]$ be some smaller dimensionality.

7.3.1 Regret bound

Throughout this section, we define a "dual set" and regularizer inducing dual variables as follows:

$$\mathcal{Y} := \{\mathbf{Y} \mid \mathbf{0} \preceq \mathbf{Y} \preceq \mathbf{I},\ \mathrm{Tr}(\mathbf{Y}) = k\}\,,\ r(\mathbf{Y}) := \langle \mathbf{Y}, \log \mathbf{Y}\rangle - \mathrm{Tr}(\mathbf{Y}). \tag{7.33}$$

Finally, we define the projection operator for any symmetric matrix $\mathbf{S}$,

$$\nabla r^*(\mathbf{S}) := \mathrm{argmin}_{\mathbf{Y}\in\mathcal{Y}} \{\langle -\mathbf{S}, \mathbf{Y}\rangle + r(\mathbf{Y})\}\,,\ \text{where } r^*(\mathbf{S}) := \max_{\mathbf{Y}\in\mathcal{Y}} \{\langle \mathbf{S}, \mathbf{Y}\rangle - r(\mathbf{Y})\}\,. \tag{7.34}$$

Here, we remark that it is a direct application of convex duality and the following fact (which is standard, and follows from e.g. the arguments of [542]) that ∇r^* is unique, and is the gradient of r^*, the Fenchel dual of r over the set $\mathcal{Y}$.

Fact 9. *Function r defined in* (7.33) *is $\frac{1}{k}$-strongly convex over $\mathcal{Y}$ in $\|\cdot\|_{\mathrm{tr}}$, and has range $k \log \frac{d}{k}$.*

We prove a helper lemma about the structure of ∇r^*, using its closed form derived in [136].

Fact 10 ([136], Lemma 7.3)**.** *Given symmetric matrix $\mathbf{S}$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ and corresponding eigenvectors $\{v_j\}_{j\in[d]}$, we can compute $\nabla r^*(\mathbf{S})$ as follows. Define*

$$\tau(\mathbf{S}) := \max\left\{\tau \;\middle|\; \tau > 0,\ \frac{\exp(\tau)}{\sum_{j\in[d]} \exp(\min(\tau, \lambda_j))} \leq \frac{1}{k}\right\}. \tag{7.35}$$

Then,

$$\nabla r^*(\mathbf{S}) = \sum_{j\in[d]} \frac{k \exp(\min(\tau(\mathbf{S}), \lambda_j))}{\sum_{j'\in[d]} \exp(\min(\tau(\mathbf{S}), \lambda_{j'}))} v_j v_j^\top.$$

In other words, ∇r^* exponentiates its argument and normalizes the trace to be k, with the exception of "large" coordinates which are truncated so that the resulting matrix is operator norm bounded (as in the definition of $\mathcal{Y}$). We now give a "refined regret bound" for Algorithm 25 when all gain matrices $\{\mathbf{G}_t\}_{t\geq 0}$ are positive and bounded. The bound is refined in the sense that it

Algorithm 25: KFMMW$(k, \{\mathbf{G}_t\}_{t\geq 0}, \eta)$

Input: Gain matrices $\{\mathbf{G}_t\}_{t\geq 0}$, step size $\eta > 0$;
$\mathbf{Y}_0 \leftarrow \frac{k}{d}\mathbf{I}$, $\mathbf{S}_0 \leftarrow \nabla r(\mathbf{Y}_0) = \log(\frac{k}{d})\mathbf{I}$;
for $t \geq 0$ **do**
  $\mathbf{S}_{t+1} \leftarrow \mathbf{S}_t + \eta\mathbf{G}_t$;
  $\mathbf{Y}_{t+1} \leftarrow \nabla r^*(\mathbf{S}_{t+1})$;

depends directly on the inner products $\langle \mathbf{G}_t, \mathbf{Y}_t\rangle$ rather than a looser, more standard bound such as $k\left\|\mathbf{G}_t\right\|_{\mathrm{op}}$ (cf. discussion in [21]). In proving Proposition 23, we will rely on a new bound on Bregman divergences with respect to r^*, which is stated here, and proven in the following Section 7.3.2.

Lemma 88. *For symmetric matrix $\mathbf{S}$, positive semidefinite $\mathbf{G}$, and scalar $\eta > 0$ let $\mathbf{S}' = \mathbf{S} + \eta\mathbf{G}$. Suppose that $\left\|\eta\mathbf{G}\right\|_{\mathrm{op}} \leq \frac{1}{2}$. Then,*

$$V_{\mathbf{S}}^{r^*}(\mathbf{S}') \leq \langle \eta\mathbf{G}, \nabla r^*(\mathbf{S})\rangle .$$

Proposition 23. *Suppose the input gain matrices to Algorithm 25 satisfy the bound, for all $t \geq 0$,*

$$\mathbf{0} \preceq \eta\mathbf{G}_t \preceq \frac{1}{2}\mathbf{I}.$$

Then, we have the guarantee for all $T \geq 1$, and all $\mathbf{U} \in \mathcal{Y}$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{U}\rangle \leq \frac{2}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle + \frac{k\log d}{\eta T}.$$

Proof. Fix some $\mathbf{U} \in \mathcal{Y}$ throughout this proof, and note that by Fact 9, $V_{\mathbf{Y}_0}^{r}(\mathbf{U}) \leq k\log d$ as $\mathbf{Y}_0$ minimizes r. Moreover, fix $\mathbf{\Psi} := \nabla r(\mathbf{U})$; it is a straightforward computation that the inverse mapping $\nabla r^*(\mathbf{\Psi}) = \mathbf{U}$ holds, via Fact 10. For each iteration t,

$$\begin{aligned}\langle \eta\mathbf{G}_t, \mathbf{U} - \mathbf{Y}_t\rangle &= \langle \mathbf{S}_{t+1} - \mathbf{S}_t, \nabla r^*(\mathbf{\Psi}) - \nabla r^*(\mathbf{S}_t)\rangle \\ &= V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_t) - V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_{t+1}) + V_{\mathbf{S}_t}^{r^*}(\mathbf{S}_{t+1}) \leq V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_t) - V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_{t+1}) + \langle \eta\mathbf{G}_t, \mathbf{Y}_t\rangle .\end{aligned} \tag{7.36}$$

The second equality is the well-known three-point equality of Bregman divergence and follows from expanding definitions, and in the last inequality we used Lemma 88. Telescoping (7.36) across all iterations and dividing by ηT, we arrive at the bound

$$\frac{1}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{U} - \mathbf{Y}_t\rangle \leq \frac{1}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle + \frac{V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_0)}{\eta T}.$$

The conclusion follows by rearrangement and using that (from Fact 9 and $\nabla r(\mathbf{Y}_0) = \mathbf{S}_0$)

$$\begin{aligned} V_{\mathbf{\Psi}}^{r^*}(\mathbf{S}_0) &= r^*(\mathbf{S}_0) - r^*(\mathbf{\Psi}) - \langle \mathbf{U}, \mathbf{S}_0 - \mathbf{\Psi} \rangle \\ &= (\langle \mathbf{Y}_0, \mathbf{S}_0 \rangle - r(\mathbf{Y}_0)) - (\langle \mathbf{U}, \mathbf{\Psi} \rangle - r(\mathbf{U})) - \langle \mathbf{U}, \mathbf{S}_0 - \mathbf{\Psi} \rangle \\ &= r(\mathbf{U}) - r(\mathbf{Y}_0) - \langle \nabla r(\mathbf{Y}_0), \mathbf{U} - \mathbf{Y}_0 \rangle = V_{\mathbf{Y}_0}^{r}(\mathbf{U}) \leq k \log d. \end{aligned}$$

□

In Section 7.3.4, where we will only have approximate access to the $\{\mathbf{Y}_t\}_{t \geq 0}$, we give a simple bound showing that the guarantee in Proposition 23 does not significantly deteriorate as Corollary 26.

7.3.2 Refined divergence bound

In this section, we prove Lemma 88. The proof is patterned from calculations in [108, 285] tailored towards the specific properties of the functions r, r^* in (7.33), (7.34). We define the vector variants of these functions, denoted $r_{\text{vec}} : \mathcal{Y}_{\text{vec}} \to \mathbb{R}$ and $r^*_{\text{vec}} : \mathbb{R}^d \to \mathbb{R}$, by

$$r_{\text{vec}}(y) := \langle y, \log y \rangle - \|y\|_1 \,, \; r^*_{\text{vec}}(s) := \min_{y \in \mathcal{Y}_{\text{vec}}} \{\langle -s, y \rangle + r(y)\} \,,$$

where $\mathcal{Y}_{\text{vec}}$ is the set of nonnegative vectors with ℓ_1 norm k and maximum entry bounded by 1. Here, we use $\log y$ to denote the entrywise logarithm of a vector.

Lemma 89. *For $s \in \mathbb{R}^d$, overload $\tau(s)$ to mean* (7.35) *applied to a matrix whose eigenvalues are given by s. Then, r^*_{vec} is twice-differentiable at s if and only if no coordinate of s is equal to $\tau(s)$.*

Proof. Suppose without loss throughout this proof that s is sorted so $s_1 \geq \ldots \geq s_d$; we may do this since r^*_{vec} is symmetric in its arguments. Also, define (overloading (7.35) appropriately for vectors)

$$N(s) := \sum_{j \in [d]} \exp(\min(\tau(s), s_j)) \implies [\nabla r^*_{\text{vec}}(s)]_j = \frac{k \exp(\min(\tau(s), s_j))}{N(s)}. \tag{7.37}$$

This implication is via a direct modification of the calculations leading to Fact 10 (alternatively, this follows from Corollary 3.3 of [356] since r^* is a spectral function).

Twice-differentiable case. We first prove that $r^*_{\text{vec}}(s)$ is twice-differentiable when no coordinate of s is $\tau(s)$; suppose that for some $0 \leq \ell \leq k-1$, exactly ℓ coordinates of s are (strictly) larger than $\tau(s)$.[9] If $\ell = 0$, it is clear that r^*_{vec} is twice-differentiable, so we focus on the case $\ell \neq 0$; in this case, by the definition of $\tau(s)$ (summing over indices larger and smaller than τ separately),

$$N(s) = k \exp(\tau(s)) = \ell \exp(\tau(s)) + \sum_{j \notin [\ell]} \exp(s_j) \implies \exp(\tau(s)) = \frac{\sum_{j \notin [\ell]} \exp(s_j)}{k - \ell}.$$

[9] From the definition of τ, we cannot have $\ell \geq k$ since otherwise the sum of the k largest elements is too large.

We thus compute

$$\frac{\partial}{\partial s_j}\exp(\tau(s)) = \begin{cases} 0 & j \in [\ell] \\ \frac{\exp(s_j)}{k-\ell} & j \notin [\ell] \end{cases}, \ \frac{\partial}{\partial s_j}N(s) = \begin{cases} 0 & j \in [\ell] \\ \frac{k\exp(s_j)}{k-\ell} & j \notin [\ell] \end{cases}. \tag{7.38}$$

It is then a straightforward calculation that $\nabla^2_{ij} r^*_{\text{vec}}(s)$ exists in all cases, upon differentiating coordinates of ∇r^*_{vec} as computed in (7.37). In particular,

$$\nabla^2_{ij} r^*_{\text{vec}}(s) = \begin{cases} \frac{k\exp(s_i)}{N(s)} - \frac{k\exp(s_i)^2}{N(s)^2} & i = j \notin [\ell] \\ -\frac{k\exp(s_i)\exp(s_j)}{N(s)^2} & i, j \notin [\ell], i \neq j \\ 0 & \text{otherwise} \end{cases}. \tag{7.39}$$

This also shows that all $\nabla^2_{ij} r^*_{\text{vec}}$ are continuous in a small neighborhood of s, so we conclude r^*_{vec} is twice-differentiable at s.

Non-twice-differentiable case. Next, suppose we are in the case where some coordinate $s_\ell = \tau(s)$. We claim that $\frac{\partial}{\partial s_\ell}\frac{\partial}{\partial s_\ell} r^*_{\text{vec}}(s)$ does not exist. In particular, perturbing s_ℓ in a positive direction does not affect $\tau(s)$, and thus does not affect $N(s)$ either, so the derivative from above of $\frac{\partial}{\partial s_\ell} r^*_{\text{vec}}(s)$ with respect to s_ℓ vanishes. To compute the derivative from below, suppose without loss of generality that $s_\ell \geq \tau(s)$ but $s_{\ell+1} < \tau(s)$. We handle the case where $\ell \geq 2$ here, and discuss $\ell = 1$ at the end. We first compute the effect on negatively perturbing s_ℓ on $\tau(s)$; for vanishing $\delta > 0$, let $s' = s - \delta e_\ell$. Since τ is weakly monotone in its argument, clearly $s_j \geq \tau(s) > s'_\ell$ for $j \in [\ell - 1]$, so since

$$k\exp(s'_\ell) \leq \ell\exp(s'_\ell) + (k-\ell)\exp(\tau(s)) = \ell\exp(s'_\ell) + \sum_{j \notin [\ell]}\exp(s_j) = \sum_{j \in [d]}\exp\left(\min(s'_\ell, s'_j)\right),$$

we have by the definition (7.35) that $\tau(s') \geq s'_\ell$. Next, by

$$\begin{aligned} k\exp(\tau(s')) &= (\ell-1)\exp(\tau(s')) + \exp(s'_\ell) + \sum_{j \notin [\ell]}\exp(s_j) \\ \implies \exp(\tau(s')) &= \frac{\exp(s'_\ell) + \sum_{j \notin [\ell]}\exp(s_j)}{k-(\ell-1)} = \frac{\exp(s'_\ell) + (k-\ell)\exp(\tau(s))}{k-(\ell-1)}, \end{aligned}$$

we see that $\tau(s') < \tau(s)$ since s'_ℓ decreased. It is straightforward to see from this that since

$$\left[\frac{\partial}{\partial s_\ell}\right]_- \exp(\tau(s)) = \frac{\exp(s_\ell)}{k-(\ell-1)} \implies \left[\frac{\partial}{\partial s_\ell}\right]_- \sum_{j \in [d]}\exp\left(\min(\tau(s), s_j)\right) = \frac{k\exp(s_\ell)}{k-(\ell-1)},$$

where $[\frac{\partial}{\partial s_\ell}]_-$ is the derivative from below, we have

$$\left[\frac{\partial}{\partial s_\ell}\right]_- \frac{k\exp(s_\ell)}{\sum_{j\in[d]}\exp\left(\min(\tau(s),s_j)\right)}$$
$$= \frac{k}{\left(\sum_{j\in[d]}\exp\left(\min(\tau(s),s_j)\right)\right)^2}\left(\exp(s_\ell)\sum_{j\in[d]}\exp\left(\min(\tau(s),s_j)\right) - \frac{k\exp(s_\ell)^2}{k-(\ell-1)}\right) \neq 0.$$

The last inequality is by

$$\sum_{j\in[d]}\exp\left(\min(\tau(s),s_j)\right) = k\exp(\tau(s)) \neq \frac{k}{k-(\ell-1)}\exp(\tau(s)) = \frac{k}{k-(\ell-1)}\exp(s_\ell).$$

Thus, the derivatives from above and below do not agree as desired. Finally, consider when $\ell = 1$; the above calculations imply that $\tau(s') = \infty$ (since then no element needs to be truncated). Hence,

$$\left[\frac{\partial}{\partial s_\ell}\right]_- \frac{k\exp(s_\ell)}{\sum_{j\in[d]}\exp\left(\min(\tau(s),s_j)\right)} = \frac{k}{\left(\sum_{j\in[d]}\exp\left(s_j\right)\right)^2}\left(\exp(s_\ell)\sum_{j\in[d]}\exp\left(s_j\right) - \exp(s_\ell)^2\right) \neq 0.$$

□

We next prove a bound on quadratic forms with respect to the (matrix) Hessian of r^*, at symmetric matrices $\mathbf{S}$ where the function is twice-differentiable. We crucially use formulas for the derivatives of spectral functions (permutation-invariant scalar-valued functions on symmetric matrices which depend only on the eigenvalues), from [356, 357].

Lemma 90. *Let $\mathbf{S} = \mathbf{U}^\top \mathbf{diag}(s)\,\mathbf{U}$ be a symmetric matrix with eigenvalues s sorted so that $s_1 \geq \ldots \geq s_d$, and $\mathbf{U}$ is an orthonormal basis. Then, r^* is twice-differentiable at $\mathbf{S}$ if and only if no coordinate of s equals $\tau(\mathbf{S})$. Further, when r^* is twice-differentiable at $\mathbf{S}$, for any positive semidefinite $\mathbf{G}$,*

$$\nabla^2 r^*(\mathbf{S})[\mathbf{G},\mathbf{G}] \leq \left\langle \nabla r^*(\mathbf{S}), \mathbf{G}^2\right\rangle.$$

Proof. The first claim is a direct consequence of Lemma 89 and the first part of Theorem 3.3 of [357], which states that when r^* is a spectral function of $\mathbf{S}$, it is twice-differentiable at $\mathbf{S}$ if and only if r^*_{vec} is twice-differentiable at s. Moreover, Theorem 3.3 of [357] gives the formula

$$\nabla^2 r^*(\mathbf{S})[\mathbf{G},\mathbf{G}] = \nabla^2 r^*_{\mathrm{vec}}(s)\left[\mathrm{diagvec}(\widetilde{\mathbf{G}}),\mathrm{diagvec}(\widetilde{\mathbf{G}})\right] + \left\langle \mathcal{A}, \widetilde{\mathbf{G}}\circ\widetilde{\mathbf{G}}\right\rangle,$$
$$\text{where } \widetilde{\mathbf{G}} = \mathbf{U}\mathbf{G}\mathbf{U}^\top,\ \mathcal{A}_{ij} = \begin{cases} 0 & i=j \\ \frac{\nabla_i r^*_{\mathrm{vec}}(s) - \nabla_j r^*_{\mathrm{vec}}(s)}{s_i - s_j} & i\neq j \end{cases}, \tag{7.40}$$

$\circ$ is the Hadamard (entrywise) product, and diagvec : $\mathbb{R}^{d\times d}\to\mathbb{R}^d$ returns the vector whose entries

are the diagonal of the input matrix. Here, we assume that no two entries of s are identical since it is clear that the scalar-valued Hessian is continuous at s by the formula (7.39), so Theorem 4.2 of [357] shows that $\nabla^2 r^*$ is also continuous at $\mathbf{S}$ (thus we can perturb $\mathbf{S}$ infinitesimally so the eigenvalues are unique). Now, let $\tilde{s} = \min(\tau(s), s)$ entrywise. We first have

$$\begin{aligned}\nabla^2 r^*_{\mathrm{vec}}(s)\left[\mathrm{diagvec}(\widetilde{\mathbf{G}}), \mathrm{diagvec}(\widetilde{\mathbf{G}})\right] &\leq \mathbf{diag}\left(\left\{\frac{k\exp(s_i)}{N(s)}\right\}_{s_i \leq \tau(s)}\right)\left[\mathrm{diagvec}(\widetilde{\mathbf{G}}), \mathrm{diagvec}(\widetilde{\mathbf{G}})\right] \\ &\leq \frac{k}{N(s)}\sum_{i\in[d]}\exp(\tilde{s}_i)\left(\widetilde{\mathbf{G}}_{ii}\right)^2.\end{aligned} \tag{7.41}$$

Here, we used that $\nabla^2 r^*_{\mathrm{vec}}(s)$ is a diagonal matrix minus a rank-one term, restricted to eigenvalues which are at most $\tau(s)$ as calculated in (7.39). Next, we claim that for any tuple $i \neq j \in [d]$,

$$\frac{\exp(\tilde{s}_i) - \exp(\tilde{s}_j)}{s_i - s_j} \leq \frac{\exp(\tilde{s}_i) + \exp(\tilde{s}_j)}{2}.$$

Without loss of generality assume $s_i > s_j$. This claim is obvious for any tuple where $s_i > s_j \geq \tau(s)$. For all other cases, we recall the identity $\frac{\exp(a)-\exp(b)}{a-b} \leq \frac{\exp(a)+\exp(b)}{2}$ for all $a \neq b$ (cf. Lemma B.3, [285]). Then, if $s_j \leq s_i < \tau(s)$, a direct application of this identity yields the claim; for the final case where $s_j < \tau(s) \leq s_i$, this follows from also using $s_i - s_j \geq \tilde{s}_i - \tilde{s}_j$. Continuing,

$$\begin{aligned}\left\langle \mathcal{A}, \widetilde{\mathbf{G}} \circ \widetilde{\mathbf{G}} \right\rangle &= \frac{k}{N(S)}\sum_{i\neq j\in[d]}\frac{\exp(\tilde{s}_i) - \exp(\tilde{s}_j)}{s_i - s_j}\left(\widetilde{\mathbf{G}}_{ij}\right)^2 \\ &\leq \frac{k}{N(S)}\sum_{i\neq j\in[d]}\frac{\exp(\tilde{s}_i) + \exp(\tilde{s}_j)}{2}\left(\widetilde{\mathbf{G}}_{ij}\right)^2.\end{aligned} \tag{7.42}$$

Combining (7.41) and (7.42) in the formula (7.40),

$$\begin{aligned}\nabla^2 r^*(\mathbf{S})[\mathbf{G}, \mathbf{G}] &\leq \frac{k}{N(S)}\sum_{i,j\in[d]}\frac{\exp(\tilde{s}_i) + \exp(\tilde{s}_j)}{2}\left(\widetilde{\mathbf{G}}_{ij}\right)^2 \\ &= \frac{k}{N(S)}\sum_{i,j\in[d]}\exp(\tilde{s}_i)\left(\widetilde{\mathbf{G}}_{ij}\right)^2 = \frac{k}{N(S)}\sum_{i\in[d]}\exp(\tilde{s}_i)\left(\sum_{j\in[d]}\left(\widetilde{\mathbf{G}}_{ij}\right)^2\right) \\ &= \sum_{i\in[d]}\frac{k\exp(\tilde{s}_i)}{N(S)}\left[\widetilde{\mathbf{G}}^2\right]_{ii} = \left\langle \mathbf{diag}\left(\nabla r^*_{\mathrm{vec}}(s)\right), \widetilde{\mathbf{G}}^2 \right\rangle.\end{aligned}$$

Finally, note that $\widetilde{\mathbf{G}}^2 = \mathbf{U}\mathbf{G}^2\mathbf{U}^\top$, so the last expression is equal to $\left\langle \mathbf{U}^\top \mathbf{diag}\left(\nabla r^*_{\mathrm{vec}}(s)\right)\mathbf{U}, \mathbf{G}^2 \right\rangle$ by the cyclic property of trace. We conclude by the fact that $\nabla r^*(\mathbf{S}) = \mathbf{U}^\top \mathbf{diag}\left(\nabla r^*_{\mathrm{vec}}(s)\right)\mathbf{U}$, since r^* is a spectral function, due to Corollary 3.3 of [356]. □

We conclude with the desired proof of Lemma 88.

Lemma 88. *For symmetric matrix* $\mathbf{S}$*, positive semidefinite* $\mathbf{G}$*, and scalar* $\eta > 0$ *let* $\mathbf{S}' = \mathbf{S} + \eta\mathbf{G}$*. Suppose that* $\|\eta\mathbf{G}\|_{\text{op}} \leq \frac{1}{2}$*. Then,*

$$V_{\mathbf{S}}^{r^*}(\mathbf{S}') \leq \langle \eta\mathbf{G}, \nabla r^*(\mathbf{S}) \rangle .$$

Proof. We first claim that without loss of generality, everywhere on the straight-line path from $\mathbf{S}$ to $\mathbf{S}'$ except for a measure-zero set (in $\mathbb{R}^1$), r^* is twice-differentiable. To see this, the Alexandrov theorem says that since r^* is convex, it is twice-differentiable everywhere except a measure-zero set in the space of its argument. However, by perturbing $\mathbf{S}$ and $\mathbf{S}'$ by a random matrix with eigenvalues distributed uniformly at random $\in [-\delta, \delta]$, for vanishing $\delta > 0$, with probability one the line between perturbed matrices only intersects the non-twice-differentiable set on a measure-zero set (this follows from the disintegration theorem). Thus, by continuity of V^{r^*} in both arguments (since ∇r^* is Lipschitz by Lemma 15.3 of [473], as r^* is the dual of a strongly convex function), we assume $\mathbf{S}$, $\mathbf{S}'$ have this property, so we may write

$$\begin{aligned} V_{\mathbf{S}}^{r^*}(\mathbf{S}') &= \int_0^1 \int_0^s \nabla^2 r^*(\mathbf{S}_t)[\mathbf{G}, \mathbf{G}] dt ds \\ &\leq \int_0^1 \int_0^s \left\langle \nabla r^*(\mathbf{S}_t), \eta^2 \mathbf{G}^2 \right\rangle dt ds \leq \frac{1}{2} \int_0^1 \int_0^s \langle \nabla r^*(\mathbf{S}_t), \eta\mathbf{G} \rangle \, dt ds. \end{aligned} \tag{7.43}$$

Here, for $t \in [0, 1]$ we define $\mathbf{S}_t = \mathbf{S} + t\eta\mathbf{G}$, and used Lemma 90 in the second line (almost everywhere) as well as the assumed bound on $\|\eta\mathbf{G}\|_{\text{op}}$ so that $\eta^2\mathbf{G}^2 \preceq \frac{1}{2}\eta\mathbf{G}$. Define $p(t) := r^*(\mathbf{S}_t)$ and $v(t) := V_{\mathbf{S}}^{r^*}(\mathbf{S}_t)$; then,

$$\int_0^s \langle \nabla r^*(\mathbf{S}_t), \eta\mathbf{G} \rangle \, dt = p(s) - p(0) = v(s) + \langle \nabla r^*(\mathbf{S}), s\eta\mathbf{G} \rangle \leq v(1) + \langle \nabla r^*(\mathbf{S}), s\eta\mathbf{G} \rangle .$$

In the last inequality, we used that v is increasing, which can be seen via

$$tv'(t) = \langle t\eta\mathbf{G}, \nabla r^*(\mathbf{S}_t) - \nabla r^*(\mathbf{S}) \rangle \geq 0.$$

Substituting back into (7.43),

$$V_{\mathbf{S}}^{r^*}(\mathbf{S}') \leq \frac{1}{2} \int_0^1 \left(v(1) + \langle \nabla r^*(\mathbf{S}), s\eta\mathbf{G} \rangle \right) ds \leq \frac{1}{2} v(1) + \frac{1}{2} \langle \nabla r^*(\mathbf{S}), \eta\mathbf{G} \rangle .$$

Rearranging and using that $V_{\mathbf{S}}^{r^*}(\mathbf{S}') = v(1)$ yields the desired bound. □

7.3.3 Refined k-PCA guarantees

We show a refined bound on the guarantees of simultaneous power iteration for approximately learning the top k eigenvectors of a positive semidefinite matrix (i.e. k-PCA). In particular, the

main result of this section (Proposition 24) strengthens Theorem 6.1 in [136] by a factor of k.

Algorithm 26: $\mathsf{Power}(\mathbf{A}, \lambda_{\max}, \lambda_{\min}, k, \epsilon, \delta)$

Input: Positive semidefinite $\mathbf{A} \in \mathbb{R}^{d \times d}$ with $\lambda_{\min}\mathbf{I} \preceq \mathbf{A} \preceq \lambda_{\max}\mathbf{I}$, accuracy $\epsilon \in (0,1)$, $k \in [d]$, $\delta \in (0,1)$;

$N \leftarrow \Theta\left(\frac{1}{\epsilon}\log\left(\frac{d}{\delta\epsilon} \cdot \frac{\lambda_{\max}}{\lambda_{\min}}\right)\right)$ for a sufficiently large universal constant;

$\mathbf{G} \in \mathbb{R}^{d \times k}$ entrywise $\sim \mathcal{N}(0,1)$;

Return: $\mathbf{V} \in \mathbb{R}^{d \times k}$, an orthonormal basis for the column span of $\mathbf{A}^N\mathbf{G}$;

For the remainder of this section, we will fix a particular positive semidefinite matrix $\mathbf{A} = \mathbf{U}^\top \mathbf{diag}\,(\lambda)\,\mathbf{U}$, where $\mathbf{U} \in \mathbb{R}^{d \times d}$ is orthonormal and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ are the ordered eigenvalues of $\mathbf{A}$. We will also define three sets which partition $[d]$:

$$\begin{aligned} L &:= \{j \in [d] \mid \lambda_j > (1+\tfrac{\epsilon}{4})\lambda_{k+1}\}, \\ M &:= \{j \in [d] \mid (1+\tfrac{\epsilon}{4})\lambda_{k+1} \geq \lambda_j \geq (1-\tfrac{\epsilon}{4})\lambda_{k+1}\}, \\ S &:= \{j \in [d] \mid \lambda_j < (1-\tfrac{\epsilon}{4})\lambda_{k+1}\}. \end{aligned} \tag{7.44}$$

In particular, L, M, and S are the "large", "medium", and "small" eigenvalues of $\mathbf{A}$. We first give two key structural results, which say that with high probability, the span of $\mathbf{V}$ contains essentially all the ℓ_2 mass of any vector in L, and essentially none of the ℓ_2 mass of any vector in S.

Lemma 91. *Let $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$ where $\mathbf{V}$ is the output of Algorithm 26. With probability at least $1 - \frac{\delta}{3} - \exp(-Ck)$ for a universal constant C, for all $j \in S$, $\|\mathbf{P}u_j\|_2 \leq \frac{\lambda_d^2}{\lambda_1^2} \cdot \frac{\epsilon^2}{64d^2}$, where u_j is row j of $\mathbf{U}$, and we follow notation in (7.44).*

Proof. By rotational invariance of Gaussian matrices, it suffices to consider the case where $\mathbf{A}$ is diagonal and $\mathbf{U}$ is the identity; henceforth in this lemma, u_j is the j^{th} standard basis vector. Recall that $\mathbf{P}$ is the projection onto the column span of $\mathbf{A}^N\mathbf{G}$. We explicitly compute

$$\mathbf{P} = \mathbf{A}^N\mathbf{G}\left(\mathbf{G}^\top\mathbf{A}^{2N}\mathbf{G}\right)^{-1}\mathbf{G}^\top\mathbf{A}^N \implies \|\mathbf{P}u_j\|_2^2 = u_j^\top\mathbf{A}^N\mathbf{G}\left(\mathbf{G}^\top\mathbf{A}^{2N}\mathbf{G}\right)^{-1}\mathbf{G}^\top\mathbf{A}^N u_j. \tag{7.45}$$

Here, we used that $\mathbf{P}^2 = \mathbf{P}$. Now, notice that (where $\mathbf{G}_{j:}$ is row j of $\mathbf{G}$)

$$\mathbf{G}^\top\mathbf{A}^{2N}\mathbf{G} = \sum_{j\in[d]} \lambda_j^{2N}\mathbf{G}_{j:}\mathbf{G}_{j:}^\top \succeq \lambda_k^{2N}\sum_{j\in[k]}\mathbf{G}_{j:}\mathbf{G}_{j:}^\top.$$

However, Theorem 1.1 of [463] shows that with probability $\frac{\delta}{6} + \exp(-Ck)$ for some constant C, the smallest eigenvalue of a $k \times k$ Gram matrix for independent Gaussian entries is at least $\frac{\delta}{6\sqrt{k}}$.

Assuming that this happens, we then continue to bound

$$\lambda_k^{2N} \sum_{j\in[k]} \mathbf{G}_{j:}\mathbf{G}_{j:}^\top \succeq \frac{\lambda_k^{2N}\delta}{6\sqrt{k}}\mathbf{I} \implies \|\mathbf{P}u_j\|_2^2 \le \frac{6\sqrt{k}}{\delta\lambda_k^{2N}} u_j^\top \mathbf{A}^N \mathbf{G}\mathbf{G}^\top \mathbf{A}^N u_j$$
$$= \frac{6\sqrt{k}}{\delta}\left(\frac{\lambda_j}{\lambda_k}\right)^{2N} \left[\mathbf{G}\mathbf{G}^\top\right]_{jj}$$
$$\le \frac{6\sqrt{k}}{\delta}\exp\left(-\frac{\epsilon N}{2}\right)\sum_{i\in[k]}\mathbf{G}_{ji}^2.$$

In the first implication, we combined the lower bound we just derived with (7.45). Using standard chi-squared concentration bounds (cf. Lemma 1, [340]), the probability that $\sum_{i\in[k]}\mathbf{G}_{ji}^2 \ge 2k+3\log\frac{6}{\delta}$ is no more than $\frac{\delta}{6}$. Performing a union bound, with failure probability at most $\frac{\delta}{3}+\exp(-Ck)$, we have that for sufficiently large $N = \Theta\left(\frac{1}{\epsilon}\log\left(\frac{d}{\delta\epsilon}\cdot\frac{\lambda_{\max}}{\lambda_{\min}}\right)\right)$, since $k \le d$,

$$\|\mathbf{P}u_j\|_2^2 \le \frac{12k^{1.5}+18\sqrt{k}\log\frac{6}{\delta}}{\delta}\exp\left(-\frac{\epsilon N}{2}\right) \le \frac{\lambda_{\min}^2}{\lambda_{\max}^2}\cdot\frac{\epsilon^2}{64d^2} \le \frac{\lambda_d^2}{\lambda_1^2}\cdot\frac{\epsilon^2}{64d^2}.$$

Finally, adjusting the failure probability of the chi-squared tail bound by a factor of d, the conclusion follows by union bounding over all $j \in S$. □

Lemma 92. *Let $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$ where $\mathbf{V}$ is the output of Algorithm 26. With probability at least $1-\frac{\delta}{3}-d\exp(-Ck)$ for a universal constant C, for all $j \in L$, $\|\mathbf{P}u_j\|_2^2 \ge 1-\frac{\lambda_d^2}{\lambda_1^2}\cdot\frac{\epsilon^2}{64d^2}$, where u_j is row j of $\mathbf{U}$, and we follow notation in* (7.44).

Proof. By definition of L, it is clear that $j \in [k]$. Again we consider the case where $\mathbf{A}$ is diagonal and $\mathbf{U}$ is the identity without loss of generality. Let $\mathbf{G}_{[k]:}$ be the first k rows of $\mathbf{G}$, and let

$$\widetilde{\mathbf{G}} := \mathbf{G}\left(\mathbf{G}_{[k]:}\right)^{-1}.$$

Observe that the first k rows of $\widetilde{\mathbf{G}}$ are exactly $\mathbf{I}$. Also, with probability at least $1-\frac{\delta}{6}-\exp(-Ck)$, the largest singular value of $\left(\mathbf{G}_{[k]:}\right)^{-1}$ is bounded above by $\frac{6\sqrt{k}}{\delta}$, again by Theorem 1.1 of [463]. Condition on this event for the remainder of the proof. Since in this case $\mathbf{G}_{[k]:}$ is invertible, where Span denotes column span,

$$\mathrm{Span}\left(\widetilde{\mathbf{G}}\right) = \mathrm{Span}\left(\mathbf{G}\right) \implies \mathrm{Span}\left(\mathbf{A}^N\widetilde{\mathbf{G}}\right) = \mathrm{Span}\left(\mathbf{A}^N\mathbf{G}\right).$$

Fix some $j \in L$. To show the conclusion, it suffices to show that there exists a unit vector v^* in the span of $\mathbf{A}^N\widetilde{\mathbf{G}}$ with $(\langle u_j, v^*\rangle)^2 \ge 1-\frac{\lambda_d^2}{\lambda_1^2}\cdot\frac{\epsilon^2}{64d^2}$. To see this, let $\{v_i\}_{i\in[k]}$ be any orthonormal basis

for Span $\left(\mathbf{A}^N \widetilde{\mathbf{G}}\right)$ with $v_1 = v^*$; then

$$\|\mathbf{P}u_j\|_2^2 = u_j^\top \mathbf{P} u_j = \sum_{i \in [k]} \left(\langle u_j, v_i \rangle\right)^2 \geq \left(\langle u_j, v^* \rangle\right)^2 \geq 1 - \frac{\lambda_d^2}{\lambda_1^2} \cdot \frac{\epsilon^2}{64 d^2}.$$

We will choose v^* to be the normalization of $\mathbf{A}^N \widetilde{\mathbf{G}}_{:j}$ which has unit ℓ_2 norm, where $\widetilde{\mathbf{G}}_{:j}$ is column j of $\widetilde{\mathbf{G}}$. By standard chi-squared concentration bounds, with probability at least $1 - \frac{\delta}{6}$, all rows $i \notin [k]$ of the matrix $\mathbf{G}$ have squared ℓ_2 norm at most

$$2k + 3 \log \frac{6d}{\delta}.$$

Here, we adjusted the failure probability of Lemma 1 in [340] by a factor of d and union bounded over all $i \notin [k]$. Now, this implies that for all $i \notin [k]$,

$$\left\| \left(\left(\mathbf{G}_{[k]:} \right)^{-1} \right)^\top \mathbf{G}_{i:}^\top \right\|_2^2 \leq \frac{72k^2 + 108k \log \frac{6d}{\delta}}{\delta^2} \implies \widetilde{\mathbf{G}}_{ij}^2 \leq \frac{72k^2 + 108k \log \frac{6d}{\delta}}{\delta^2} \text{ for all } j \in [k].$$

We conclude that the column vector $\widetilde{\mathbf{G}}_{:j}$ has the property that

$$\widetilde{\mathbf{G}}_{ij}^2 \begin{cases} = 1 & i = j \\ = 0 & i \neq j,\ i \in [k] \\ \leq \frac{72k^2 + 108k \log \frac{6d}{\delta}}{\delta^2} & i \notin [k]. \end{cases}$$

Here, the first two cases are by design, and the last is by our earlier derivation. Thus, by choosing $N = \Theta\left(\frac{1}{\epsilon} \log\left(\frac{d}{\delta\epsilon} \cdot \frac{\lambda_{\max}}{\lambda_{\min}}\right)\right)$ to be sufficiently large (as in the ending of the proof of Lemma 91), we see that $\mathbf{A}^N \widetilde{\mathbf{G}}_{:j}$ places all but a negligible amount of ℓ_2^2 mass on coordinate j, where we use that $\lambda_j^N \geq (1 + \frac{\epsilon}{4})^N \lambda_i^N$ for all $i \notin [k]$. □

We also give a simple helper calculation for demonstrating Loewner orderings.

Lemma 93. *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times d}$ be positive semidefinite and suppose for any fixed unit test vector $v \in \mathbb{R}^d$ and some $\epsilon \in (0, 1)$,*

$$\left| v^\top \left(\mathbf{A} - \mathbf{B} \right) v \right| \leq \epsilon v^\top \mathbf{B} v.$$

Then, $(1 - \epsilon)\mathbf{B} \preceq \mathbf{A} \preceq (1 + \epsilon)\mathbf{A}$.

Proof. The upper bound follows from

$$v^\top \mathbf{A} v \leq v^\top \mathbf{B} v + \left| v^\top \left(\mathbf{A} - \mathbf{B} \right) v \right| \leq (1 + \epsilon) v^\top \mathbf{B} v.$$

The lower bound follows similarly. □

Our main bound follows from an application of the above three results.

Proposition 24. *Let* $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$ *where* $\mathbf{V}$ *is the output of Algorithm 26. With probability at least* $1 - \frac{2\delta}{3} - 2\exp(-Ck)$ *for a universal constant* C*,*

$$(1-\epsilon)\left(\mathbf{PAP} + (\mathbf{I}-\mathbf{P})\,\mathbf{A}\,(\mathbf{I}-\mathbf{P})\right) \preceq \mathbf{A} \preceq (1+\epsilon)\left(\mathbf{PAP} + (\mathbf{I}-\mathbf{P})\,\mathbf{A}\,(\mathbf{I}-\mathbf{P})\right). \tag{7.46}$$

Proof. Condition on the conclusions of Lemmas 91 and 92 holding for the rest of this proof. We note

$$\mathbf{A} - \left(\mathbf{PAP} + (\mathbf{I}-\mathbf{P})\,\mathbf{A}\,(\mathbf{I}-\mathbf{P})\right) = \mathbf{PA}\,(\mathbf{I}-\mathbf{P}) + (\mathbf{I}-\mathbf{P})\,\mathbf{AP}.$$

Hence, applying Lemma 93, for any fixed unit test vector $v \in \mathbb{R}^d$, this proposition asks to show

$$2\left|y^\top \mathbf{A} x\right| \le \epsilon\left(x^\top \mathbf{A} x + y^\top \mathbf{A} y\right), \text{ where } x := \mathbf{P}v \text{ and } y := (\mathbf{I}-\mathbf{P})\,v.$$

Recall that $\mathbf{A} = \sum_{j\in[d]} \lambda_j u_j u_j^\top$. Letting $\tilde{x} := \mathbf{U}x$ and $\tilde{y} := \mathbf{U}y$, it suffices to show

$$\left|\sum_{j\in[d]} \lambda_j \tilde{x}_j \tilde{y}_j\right| \le \frac{\epsilon}{2}\sum_{j\in[d]} \lambda_j\left(\tilde{x}_j^2 + \tilde{y}_j^2\right). \tag{7.47}$$

Since $\langle \tilde{x}, \tilde{y}\rangle = \langle x, y\rangle = 0$ by the definition of x, y,

$$\left|\sum_{j\in[d]} \lambda_j \tilde{x}_j \tilde{y}_j\right| = \left|\sum_{j\in[d]} (\lambda_j - \lambda_{k+1})\tilde{x}_j\tilde{y}_j\right| \le \frac{\epsilon}{4}\sum_{j\in M} \lambda_{k+1}|\tilde{x}_j\tilde{y}_j| + \left|\sum_{j\notin M} (\lambda_j - \lambda_{k+1})\tilde{x}_j\tilde{y}_j\right|.$$

Here we used the definition of $j \in M$, so that $|\lambda_j - \lambda_{k+1}| \le \frac{\epsilon}{4}\lambda_{k+1}$. We first bound

$$\frac{\epsilon}{4}\sum_{j\in M} \lambda_{k+1}|\tilde{x}_j||\tilde{y}_j| \le \frac{\epsilon}{4(1-\frac{\epsilon}{4})}\sum_{j\in M} \lambda_j|\tilde{x}_j||\tilde{y}_j| \le \frac{\epsilon}{4}\sum_{j\in M} \lambda_j\left(\tilde{x}_j^2 + \tilde{y}_j^2\right) \le \frac{\epsilon}{4}\sum_{j\in[d]} \lambda_j\left(\tilde{x}_j^2 + \tilde{y}_j^2\right). \tag{7.48}$$

Moreover, by Lemma 91, for each $j \in S$, we have

$$|\tilde{x}_j| = \left|u_j^\top \mathbf{P} v\right| \le \|\mathbf{P}u_j\|_2 \|v\|_2 \le \frac{\lambda_d}{\lambda_1}\cdot\frac{\epsilon}{8d},$$

and similarly for each $j \in L$, $\tilde{y}_j \le \frac{\lambda_d}{\lambda_1}\cdot\frac{\epsilon}{8d}$ by Lemma 92. Thus, since all $|\tilde{x}_j|$ and $|\tilde{y}_j|$ are at most 1,

$$\begin{aligned}
\left|\sum_{j\in S}(\lambda_j - \lambda_{k+1})\tilde{x}_j\tilde{y}_j\right| &\le \lambda_1\sum_{j\in S}|\tilde{x}_j| \le \frac{\epsilon}{8}\lambda_d \le \frac{\epsilon}{8}\sum_{j\in[d]}\lambda_j\left(\tilde{x}_j^2 + \tilde{y}_j^2\right), \\
\left|\sum_{j\in L}(\lambda_j - \lambda_{k+1})\tilde{x}_j\tilde{y}_j\right| &\le \lambda_1\sum_{j\in L}|\tilde{y}_j| \le \frac{\epsilon}{8}\lambda_d \le \frac{\epsilon}{8}\sum_{j\in[d]}\lambda_j\left(\tilde{x}_j^2 + \tilde{y}_j^2\right).
\end{aligned} \tag{7.49}$$

Finally, combining (7.48) and (7.49), we have the desired bound (7.47). □

An unfortunate consequence of Proposition 24 is that its failure probability is exponentially related to k, rather than d. However, for sufficiently small $k = O(\log \frac{1}{\delta})$, we can use an alternative analysis of the power method due to [136] to conclude that the desired bound (7.46) holds.

Corollary 25. *There is an algorithm (either Algorithm 26 of this paper, or Algorithm 5 of [136]) which takes as input positive semidefinite $\mathbf{A} \in \mathbb{R}^{d\times d}$ with $\lambda_{\min}\mathbf{I} \preceq \mathbf{A} \preceq \lambda_{\max}\mathbf{I}$, $k \in [d]$, and accuracy parameter $\epsilon \in (0,1)$, and returns with probability at least $1-\delta$ a set of orthonormal vectors $\mathbf{V} \in \mathbb{R}^{d\times k}$ such that for $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$, (7.46) holds. The number of matrix-vector products to $\mathbf{A}$ required is*

$$O\left(\frac{k}{\epsilon}\log^2\left(\frac{d}{\delta\epsilon}\frac{\lambda_{\max}}{\lambda_{\min}}\right)\right).$$

Proof. In the case where $k \geq \frac{\log 6/\delta}{C}$ for C the universal constant in Proposition 24, the conclusion is immediate from Proposition 24. In the other case, we have that $k = O(\log \frac{1}{\delta})$. Hence, we can run Algorithm 5 of [136] with an accuracy parameter which is $O(k)$ times smaller, and use their Theorem 6.1 to obtain the desired conclusion. The iteration complexity of Algorithm 5 of [136] depends linearly on the inverse accuracy, so the bound loses an additional logarithmic factor. □

7.3.4 Implementation

In this section, we give an algorithm (Algorithm 27) which takes as input a matrix $\mathbf{S}$ and produces a matrix $\widehat{\mathbf{Y}}$ such that for some choice of input $\Delta \geq 0$, we have

$$\left\|\widehat{\mathbf{Y}} - \nabla r^*(\mathbf{S})\right\|_{\mathrm{tr}} \leq k\Delta. \tag{7.50}$$

We will use this at the end of the section to give a complete (computationally efficient) implementation of an approximate variant of Algorithm 25, and give its guarantees as Corollary 26.

Proposition 25. *With probability at least $1-\delta$, the output $\widehat{\mathbf{Y}}$ of Algorithm 27 satisfies (7.50). The complexity of Lines 3-8 of the algorithm is*

$$O\left(ndk \cdot \frac{\lambda_{\max}}{\Delta^2}\log^2\left(\frac{d}{\Delta\delta}\frac{\lambda_{\max}}{\lambda_{\min}}\right)\right).$$

Moreover, for any $\epsilon \in (0,1)$, the complexity of providing $(1\pm\epsilon)$-approximate access to quadratic forms through $\widehat{\mathbf{Y}}$ for any n fixed vectors $\{v_i\}_{i\in[n]} \subset \mathbb{R}^d$ with probability at least $1-\delta$ is

$$O\left(nd \cdot \frac{\lambda_{\max}}{\epsilon^2}\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{nd}{\delta}\right)\right).$$

Proof. We will show correctness and complexity of Algorithm 27 separately.

Algorithm 27: $\mathsf{ApproxProject}(\mathbf{S}, \lambda_{\max}, \lambda_{\min}, k, \Delta, \delta)$

1 **Input:** Positive semidefinite $\mathbf{S} = \mathbf{M}^\top \mathbf{M} \in \mathbb{R}^{d \times d}$ for some explicitly given $\mathbf{M} \in \mathbb{R}^{n \times d}$ with $\lambda_{\min}\mathbf{I} \preceq \mathbf{S} \preceq \lambda_{\max}\mathbf{I}$, $k \leq d \leq n$, accuracy $\Delta \in (0,1)$, $k \in [d]$, $\delta \in (0,1)$;

2 **Output:** $\widehat{\mathbf{Y}}$ satisfying $\left\|\widehat{\mathbf{Y}} - \nabla r^*(\mathbf{S})\right\|_{\mathrm{tr}} \leq k\Delta$ with probability $\geq 1 - \delta$;

3 $\mathbf{V} \leftarrow \mathsf{Power}(\mathbf{S}, \lambda_{\max}, \lambda_{\min}, k, \frac{\Delta}{8\lambda_{\max}}, \frac{\delta}{2})$ (or when $k \leq \frac{\log 12/\delta}{C}$, use Algorithm 5 of [136]);

4 $\{u_j\}_{j \in [k]} \leftarrow$ eigenvectors of $\mathbf{V}^\top \mathbf{M}^\top \mathbf{M} \mathbf{V} \in \mathbb{R}^{k \times k}$, left-multiplied by $\mathbf{V}$;

5 For $j \in [k]$, $\tilde{\lambda}_j \leftarrow u_j^\top \mathbf{P}\mathbf{S}\mathbf{P} u_j$ where $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$;

6 $\widehat{\mathbf{S}} \leftarrow \sum_{j \in [k]} \tilde{\lambda}_j u_j u_j^\top + (1 - \frac{\Delta}{4\lambda_{\max}})(\mathbf{I} - \mathbf{P})\mathbf{S}(\mathbf{I} - \mathbf{P})$;

7 $\widehat{T} \leftarrow (1 \pm \frac{\Delta}{8})$-approximation to $\mathrm{Tr} \exp\left((1 - \frac{\Delta}{4\lambda_{\max}})(\mathbf{I} - \mathbf{P})\mathbf{S}(\mathbf{I} - \mathbf{P})\right)$ with probability $1 - \frac{\delta}{2}$;

8 $\hat{\tau} \leftarrow$ fixed point of $k \exp(\hat{\tau}) = \sum_{j \in [k]} \exp(\min(\hat{\tau}, \tilde{\lambda}_j)) + \widehat{T}$;

9 $\widehat{\mathbf{Y}} \leftarrow$
$\frac{k}{\sum_{j \in [k]} \exp(\min(\hat{\tau}, \tilde{\lambda}_j)) + \widehat{T}} \left(\sum_{j \in [k]} \exp(\min(\hat{\tau}, \tilde{\lambda}_j)) u_j u_j^\top + \exp\left((1 - \frac{\Delta}{4\lambda_{\max}})(\mathbf{I} - \mathbf{P})\mathbf{S}(\mathbf{I} - \mathbf{P})\right)\right)$;

10 **Return:** $\widehat{\mathbf{Y}}$;

Correctness guarantee. We begin with proving correctness, which we complete in two parts. In particular, we show that the following two bounds hold:

$$\left\|\nabla r^*(\mathbf{S}) - \nabla r^*(\widehat{\mathbf{S}})\right\|_{\mathrm{tr}} \leq \frac{k\Delta}{2}, \ \left\|\widehat{\mathbf{Y}} - \nabla r^*(\widehat{\mathbf{S}})\right\|_{\mathrm{tr}} \leq \frac{k\Delta}{2}. \tag{7.51}$$

By combining the two parts of (7.51) and applying the triangle inequality, we have the desired conclusion. To show the former bound, because the convex conjugate of any $\frac{1}{k}$-strongly convex function in $\|\cdot\|_{\mathrm{tr}}$ is k-smooth in $\|\cdot\|_{\mathrm{op}}$ (cf. Lemma 15.3, [473]), and Fact 9 states that r is strongly convex, it suffices to show that

$$\left\|\widehat{\mathbf{S}} - \mathbf{S}\right\|_{\mathrm{op}} \leq \frac{\Delta}{2} \implies \left\|\nabla r^*(\widehat{\mathbf{S}}) - \nabla r^*(\mathbf{S})\right\|_{\mathrm{tr}} \leq k \left\|\widehat{\mathbf{S}} - \mathbf{S}\right\|_{\mathrm{op}} \leq \frac{k\Delta}{2}. \tag{7.52}$$

Assume first that Power was used in computing Line 3. By Proposition 24, we have that

$$\|\mathbf{S} - (\mathbf{P}\mathbf{S}\mathbf{P} + (\mathbf{I} - \mathbf{P})\,\mathbf{S}\,(\mathbf{I} - \mathbf{P}))\|_{\mathrm{op}} \leq \frac{\Delta}{8\lambda_{\max}} \|\mathbf{S}\|_{\mathrm{op}} \leq \frac{\Delta}{8}. \tag{7.53}$$

Next, we claim that $\{u_j\}_{j \in [k]}$ are the eigenvectors of $\mathbf{P}\mathbf{S}\mathbf{P}$, so that

$$\sum_{j \in [k]} \tilde{\lambda}_j u_j u_j^\top = \mathbf{P}\mathbf{S}\mathbf{P}.$$

To see this, let $w_j \in \mathbb{R}^k$ be an eigenvector of $\mathbf{V}^\top \mathbf{M}^\top \mathbf{M} \mathbf{V}$ with eigenvalue $\tilde{\lambda}_j$, and let $u_j = \mathbf{V} w_j$, as

in Line 4 of Algorithm 27. Then indeed we have (since $\mathbf{V}^\top\mathbf{V}$ is the identity)

$$\mathbf{PSP}u_j = \mathbf{V}\mathbf{V}^\top\mathbf{S}\mathbf{V}\mathbf{V}^\top\mathbf{V}w_j = \mathbf{V}\left(\mathbf{V}^\top\mathbf{M}^\top\mathbf{M}\mathbf{V}w_j\right) = \tilde{\lambda}_j\mathbf{V}w_j = \tilde{\lambda}_j u_j.$$

We then compute, using the definition of $\widehat{\mathbf{S}}$ in Line 6,

$$\left\|(\mathbf{PSP} + (\mathbf{I}-\mathbf{P})\,\mathbf{S}\,(\mathbf{I}-\mathbf{P})) - \widehat{\mathbf{S}}\right\|_{\mathrm{op}} = \frac{\Delta}{4\lambda_{\max}}\left\|(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})\right\|_{\mathrm{op}} \leq \frac{3\Delta}{8\lambda_{\max}}\left\|\mathbf{S}\right\|_{\mathrm{op}} \leq \frac{3\Delta}{8}. \quad (7.54)$$

Here, we used Proposition 24 once more to (loosely) upper bound $(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})$ by $1.5\mathbf{S}$. Combining (7.52), (7.53), and (7.54) gives the first conclusion in (7.51).

We next claim the top eigenvalue of $(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})$ is at most $(1+\frac{\Delta}{4\lambda_{\max}})\tilde{\lambda}_k$; this follows from the second and third parts of Theorem 1 of [404]. Thus, by scaling down $(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})$ by a factor $1-\frac{\Delta}{4\lambda_{\max}}$, we have that its largest eigenvalue is smaller than the smallest of $\mathbf{PSP}$. Let $\{\tilde{\lambda}_j\}_{j\in[d]\setminus[k]}$, $\{u_j\}_{j\in[d]\setminus[k]}$ complete an eigendecomposition of $\widehat{\mathbf{S}}$. We conclude that none of the eigenvalues of $\widehat{\mathbf{S}} - \mathbf{PSP}$ will be truncated in the projection since they are not in the top k, so Fact 10 yields that

$$\nabla r^*(\widehat{\mathbf{S}}) = \frac{k}{\sum_{j\in[k]}\min(\sigma,\alpha_j)+T}\left(\sum_{j\in[k]}\min(\sigma,\alpha_j)u_ju_j^\top + \sum_{j\notin[k]}\alpha_j u_j u_j^\top\right),$$
$$\text{where } T := \operatorname{Tr}\exp\left(\left(1-\frac{\Delta}{4\lambda_{\max}}\right)(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})\right),$$
$$\text{and } \sigma := \exp(\tau(\tilde{\lambda})),\ \alpha_j := \exp(\tilde{\lambda}_j) \text{ for all } j\in[d].$$

Specifically, this form is clear for the first k eigenvectors, and for the remainder the ∇r^* operation applies an exponentiation and scaling (since they will not be truncated), which does not affect the relevant basis. Finally, by applying the following Lemma 94 with $\gamma = \frac{\Delta}{6}$, and using that the eigenvectors of our returned $\widehat{\mathbf{Y}}$ align exactly with those of $\widehat{\mathbf{S}}$, we have the desired second bound in (7.51). We remark that the only place we used the fact that Power was used in Line 3 thus far in this proof was in citing Theorem 1 of [404]; however, if Algorithm 5 of [136] is used, a similar statement on the top eigenvalue of $\widehat{\mathbf{S}} - \mathbf{PSP}$ follows by their Remark 6.9.

Complexity guarantee. When $\mathbf{S}$ is given in the form $\mathbf{M}^\top\mathbf{M}$, the cost of a matrix-vector product in $\mathbf{S}$ is $O(nd)$. So, from Corollary 26 the cost of Line 3 is bounded by

$$O\left(ndk\cdot\frac{\lambda_{\max}}{\Delta}\log^2\left(\frac{d}{\Delta\delta}\frac{\lambda_{\max}}{\lambda_{\min}}\right)\right).$$

In Line 4, the cost of forming the matrix $\mathbf{MV}$ is $O(ndk)$, and forming its Gram matrix and performing an eigendecomposition takes time $O(k^2d+k^3)$; left-multiplying all resulting vectors by $\mathbf{V}$ also takes time $O(k^2d)$. The cost of Line 5 for each $j\in[k]$ is $O(nd+kd)$, so the overall cost is also $O(ndk)$. Line 8 is a scalar optimization problem and will not dominate the complexity (tolerance to error in

a binary search is guaranteed via Lemma 94). The only remaining cost is in Line 6.

To estimate $\operatorname{Tr}\exp(\mathbf{A})$ to $1\pm\gamma$ accuracy for a positive semidefinite matrix $\mathbf{0} \preceq \mathbf{A} \preceq \lambda_{\max}\mathbf{I}$ (here, we note Proposition 24 guarantees $\mathbf{A} = (1 - \frac{\Delta}{4\lambda_{\max}})(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P}) \preceq \mathbf{S} \preceq \lambda_{\max}\mathbf{I}$), we will use two facts well-known in the approximate semidefinite programming literature. First, Theorem 4.1 of [466] shows that a degree-$O(\lambda_{\max}\log\frac{1}{\gamma})$ polynomial p has the property that

$$\left(1-\frac{\gamma}{3}\right)\exp(\mathbf{A}) \preceq p(\mathbf{A}) \preceq \left(1+\frac{\gamma}{3}\right)\exp(\mathbf{A}).$$

Moreover, the Johnson-Lindenstrauss lemma (e.g. the implementation given in [6]) shows that to estimate $\operatorname{Tr}\exp(\mathbf{A})$ it suffices to sample a random $\pm\frac{1}{\sqrt{r}}$ matrix $\mathbf{G}\in\mathbb{R}^{d\times r}$ for some $r = O(\log(\frac{d}{\delta})\gamma^{-2})$ and then compute

$$\sum_{j\in[r]}\left\|p\left(\frac{1}{2}\mathbf{A}\right)\mathbf{G}_{:j}\right\|_2^2 \approx_{1\pm\frac{\gamma}{3}} \sum_{j\in[r]}\left\|\exp\left(\frac{1}{2}\mathbf{A}\right)\mathbf{G}_{:j}\right\|_2^2 = \operatorname{Tr}\left(\exp\left(\frac{1}{2}\mathbf{A}\right)\mathbf{G}\mathbf{G}^\top\exp\left(\frac{1}{2}\mathbf{A}\right)\right).$$

This last quantity is a $1\pm\frac{\gamma}{3}$ approximation of $\operatorname{Tr}\exp(\mathbf{A})$ with probability $1-\frac{\delta}{2}$. The cost of this whole procedure is dominated by $O(r\lambda_{\max}\log\frac{1}{\gamma})$ matrix-vector multiplies to $\mathbf{A}$; for our choice of $\mathbf{A}$, each multiplication costs $O(nd)$ time, leading to an overall complexity of (as $\gamma = \Theta(\Delta)$)

$$O\left(nd\cdot\frac{\lambda_{\max}}{\Delta^2}\log\left(\frac{1}{\Delta}\right)\log\left(\frac{d}{\delta}\right)\right).$$

For any $v\in\mathbb{R}^d$, essentially the same strategy of sampling a random $\mathbf{G}\in\mathbb{R}^{d\times r}$ and computing

$$v^\top p\left(\frac{1}{2}\mathbf{A}\right)\mathbf{G}\mathbf{G}^\top p\left(\frac{1}{2}\mathbf{A}\right)v = \sum_{j\in[r]}\left(\left\langle\mathbf{G}_{:j}, p\left(\frac{1}{2}\mathbf{A}\right)v\right\rangle\right)^2.$$

suffices for estimating the quadratic form in $\exp(\mathbf{A})$ to $1\pm\epsilon$ accuracy, where now r and the polynomial degree depend on ϵ rather than γ. We can first apply the polynomial to each column of $\mathbf{G}$ and then compute inner products with v. To compute approximate quadratic forms in $\widehat{\mathbf{Y}}$, every part of $\widehat{\mathbf{Y}}$ is explicitly given except for the component $\exp(\mathbf{A})$ for $\mathbf{A} = (1-\frac{\Delta}{4\lambda_{\max}})(\mathbf{I}-\mathbf{P})\mathbf{S}(\mathbf{I}-\mathbf{P})$, which we can approximate with the above strategy in the desired time.

Finally, for a batch of n vectors $\{v_i\}_{i\in[n]}$, note that we can first compute all the vectors $p(\frac{1}{2}\mathbf{A})\mathbf{G}_{:j}$ in the desired time, at which point the cost of computing each quadratic form is reduced to $O(dr)$. Thus, the overall complexity is $O(ndr)$, where we adjust the logarithm in the definition of r by a factor of n to union bound the failure probability. □

We now provide the helper Lemma 94, which we remark crucially improves the error analysis in Section 7.3 of [136] by a factor of k, allowing us to avoid an additional $\operatorname{poly}(k)$ dependence.

Lemma 94. *Let $\gamma \in (0,1)$, $k \in [d]$. Given nonnegative $\{\alpha_j\}_{j\in[d]}$ sorted with $\alpha_1 \geq \ldots \geq \alpha_d$, let $T := \sum_{j\notin[k]} \alpha_j$, and let $\widehat{T} \in [(1-\gamma)T, (1+\gamma)T]$. Define σ and $\hat{\sigma}$ to be fixed points of*

$$k\sigma = \sum_{j\in[k]} \min(\sigma, \alpha_j) + T, \; k\hat{\sigma} = \sum_{j\in[k]} \min(\hat{\sigma}, \alpha_j) + \widehat{T}.$$

Then, we have

$$\sum_{j\in[d]} \left| \frac{k\min(\sigma,\alpha_j)}{\sum_{i\in[k]} \min(\sigma,\alpha_i) + T} - \frac{k\min(\hat{\sigma},\alpha_j)}{\sum_{i\in[k]} \min(\hat{\sigma},\alpha_i) + \widehat{T}} \right| \leq 3k\gamma.$$

Proof. We first comment briefly on the existence of σ, $\hat{\sigma}$. Note that in the setting of the lemma,

$$f(\hat{\sigma}) := \frac{\hat{\sigma}}{\sum_{i\in[k]} \min(\hat{\sigma},\alpha_i) + \widehat{T}}$$

is an increasing, continuous function of $\hat{\sigma}$ in the range $[0,\infty)$ which satisfies $f(0) = 0$ and $f(\infty) = \infty$, so there must be a unique $\hat{\sigma}$ satisfying $f(\hat{\sigma}) = \frac{1}{k}$. Existence of σ is proven similarly. Next, we claim

$$\hat{\sigma} \in [(1-\gamma)\sigma, (1+\gamma)\sigma]. \tag{7.55}$$

By our earlier argument, it suffices to show that $f((1+\gamma)\sigma) \geq \frac{1}{k}$ and $f((1-\gamma)\sigma) \leq \frac{1}{k}$, so that an appeal to continuity and monotonicity of f yields (7.55). To see the former bound, note that

$$\begin{aligned} f((1+\gamma)\sigma) &= \frac{(1+\gamma)\sigma}{\sum_{i\in[k]} \min((1+\gamma)\sigma,\alpha_i) + \widehat{T}} \\ &\geq \frac{(1+\gamma)\sigma}{(1+\gamma)\sum_{i\in[k]} \min(\sigma,\alpha_i) + (1+\gamma)T} \\ &= \frac{\sigma}{\sum_{i\in[k]} \min(\sigma,\alpha_i) + T} = \frac{1}{k}. \end{aligned}$$

The last equality used the definition of σ; the only inequality used $\widehat{T} \leq (1+\gamma)T$ by assumption, and $\min((1+\gamma)\sigma,\alpha_i) \leq \min((1+\gamma)\sigma,(1+\gamma)\alpha_i) = (1+\gamma)\min(\sigma,\alpha_i)$. Similarly,

$$\begin{aligned} f((1-\gamma)\sigma) &= \frac{(1-\gamma)\sigma}{\sum_{i\in[k]} \min((1-\gamma)\sigma,\alpha_i) + \widehat{T}} \\ &\leq \frac{(1-\gamma)\sigma}{(1-\gamma)\sum_{i\in[k]} \min(\sigma,\alpha_i) + (1-\gamma)T} = \frac{1}{k}. \end{aligned}$$

Here we used $(1-\gamma)\min(\sigma, \alpha_i) \le \min((1-\gamma)\sigma, \alpha_i)$. Now, we claim (7.55) implies that for all $j \in [d]$,

$$\left| \frac{\min(\sigma, \alpha_j)}{\sum_{i \in [k]} \min(\sigma, \alpha_i) + T} - \frac{\min(\hat{\sigma}, \alpha_j)}{\sum_{i \in [k]} \min(\hat{\sigma}, \alpha_i) + \widehat{T}} \right| \le \frac{3\gamma \min(\sigma, \alpha_j)}{\sum_{i \in [k]} \min(\sigma, \alpha_i) + T}. \tag{7.56}$$

To see this, we may upper and lower bound for each $j \in [d]$,

$$\begin{aligned} &(1-\gamma)\min(\sigma, \alpha_j) \le \min(\hat{\sigma}, \alpha_j) \le (1+\gamma)\min(\sigma, \alpha_j) \\ \implies &(1-\gamma)\left(\sum_{i \in [k]} \min(\sigma, \alpha_i) + T\right) \le \sum_{i \in [k]} \min(\hat{\sigma}, \alpha_i) + \widehat{T} \le (1+\gamma)\left(\sum_{i \in [k]} \min(\sigma, \alpha_i) + T\right) \\ \implies &\frac{(1-3\gamma)\min(\sigma, \alpha_j)}{\sum_{i \in [k]} \min(\sigma, \alpha_i) + T} \le \frac{\min(\hat{\sigma}, \alpha_j)}{\sum_{i \in [k]} \min(\hat{\sigma}, \alpha_i) + \widehat{T}} \le \frac{(1+3\gamma)\min(\sigma, \alpha_j)}{\sum_{i \in [k]} \min(\sigma, \alpha_i) + T}. \end{aligned} \tag{7.57}$$

This yields (7.56), which upon summing and using that $\min(\sigma, \alpha_j) = \alpha_j$ for all $j \notin [k]$, and the definition of T, yields the final conclusion after multiplying by k. □

We additionally state one helper guarantee on the properties of $\widehat{\mathbf{Y}}$.

Lemma 95. *With probability at least $1-\delta$, the output $\widehat{\mathbf{Y}}$ of Algorithm 27 satisfies*

$$\left\|\widehat{\mathbf{Y}}\right\|_{\mathrm{tr}} \le \left(1 + \frac{\Delta}{2}\right) k, \ \left\|\widehat{\mathbf{Y}}\right\|_{\mathrm{op}} \le 1 + \frac{\Delta}{2}.$$

Proof. By the definition of ∇r^*, we have that $\nabla r^*(\widehat{\mathbf{S}}) \in \mathcal{Y}$ so has operator norm at most 1 and trace norm at most k. For the first conclusion, (7.51) implies

$$\left\|\widehat{\mathbf{Y}}\right\|_{\mathrm{tr}} \le \left\|\nabla r^*(\widehat{\mathbf{S}})\right\|_{\mathrm{tr}} + \frac{k\Delta}{2}.$$

For the second conclusion, (7.57) and the operator norm bound on $\mathcal{Y}$ imply

$$\lambda_{\max}\left(\widehat{\mathbf{Y}}\right) \le (1+3\gamma)\,\lambda_{\max}\left(\nabla r^*(\widehat{\mathbf{S}})\right) \le 1 + \frac{\Delta}{2}.$$

□

Finally, we state our complete algorithm, an approximate version of the KFMMW method.

Corollary 26. *Suppose the input gain matrices to Algorithm 28 satisfy the bound, for all $t \ge 0$,*

$$\mathbf{0} \preceq \eta \mathbf{G}_t \preceq \frac{1}{2}\mathbf{I}.$$

Algorithm 28: ApproxKFMMW$(k, \{\mathbf{G}_t\}_{0\le t\le T}, \eta, \Delta, \delta)$

Input: Gain matrices $\{\mathbf{G}_t\}_{0\le t\le T}$, step size $\eta > 0$, accuracy $\Delta \in (0,1)$, $\delta \in (0,1)$;
$\mathbf{Y}_0 \leftarrow \frac{k}{d}\mathbf{I}$, $\mathbf{S}_0 \leftarrow \nabla r(\mathbf{Y}_0) = \log(\frac{k}{d})\mathbf{I}$;
for $0 \le t < T$ **do**
  $\mathbf{S}_{t+1} \leftarrow \mathbf{S}_t + \eta \mathbf{G}_t$;
  $\widehat{\mathbf{Y}}_{t+1} \leftarrow$ ApproxProject$(\mathbf{S}_{t+1} + (1 + \log(\frac{d}{k}))\mathbf{I}, t+2, 1, k, \Delta, \frac{\delta}{T})$;

Further, suppose that the $\{\mathbf{G}_t\}_{t\ge 0}$ are weakly decreasing in Loewner order. With probability $1-\delta$,

$$\|\mathbf{G}_T\|_k \le \frac{2}{T}\sum_{t=0}^{T-1}\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + \frac{k\log d}{\eta T} + \frac{k\Delta}{\eta}.$$

The complexity of Algorithm 28 is

$$O\left(ndk \cdot \frac{T^2}{\Delta^2}\log^2\left(\frac{dT}{\Delta\delta}\right)\right),$$

and the cost of providing $(1\pm\epsilon)$-approximate access to quadratic forms for any fixed n vectors $\{v_i\}_{i\in[n]} \subset \mathbb{R}^d$ through any $\widehat{\mathbf{Y}}_t$ for any $\epsilon \in (0,1)$, with failure probability at most δ, is

$$O\left(nd \cdot \frac{T}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{nd}{\delta}\right)\right).$$

Proof. We claim first that it suffices to show that the conclusion of Proposition 25 holds in each iteration t. To see why this is enough, matrix Hölder on the conclusion of Proposition 23 yields

$$\begin{aligned}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle \le \left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + \|\mathbf{G}_t\|_{\mathrm{op}}\left\|\widehat{\mathbf{Y}}_t - \mathbf{Y}_t\right\|_{\mathrm{tr}} &\le \left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + k\Delta\|\mathbf{G}_t\|_{\mathrm{op}} \le \left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + \frac{k\Delta}{2\eta} \\ \implies \frac{1}{T}\sum_{t=0}^{T-1}\langle \mathbf{G}_t, \mathbf{U}\rangle &\le \frac{2}{T}\sum_{t=0}^{T-1}\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + \frac{k\log d}{\eta T} + \frac{k\Delta}{\eta}.\end{aligned}$$

In the last inequality in the first line, we used the assumption that $\eta\mathbf{G} \preceq \frac{1}{2}\mathbf{I}$. Supremizing over $\mathbf{U}$, and using monotonicity of the gain matrices, yields the conclusion. Next, we prove that calling ApproxProject is valid. Note ∇r^* is invariant under shifts by the identity, so it suffices to first shift $\mathbf{S}_0$ to be $\mathbf{I}$ and hence $\lambda_{\min} = 1$ is a valid bound. Since for all $t \ge 0$, the change in $\mathbf{S}_t$ (i.e. $\eta\mathbf{G}_t$) is positive semidefinite and bounded by $\mathbf{I}$, we can set $\lambda_{\max} = t+2$ in the call. Finally, the failure probability comes from a union bound over all iterations, and the overall complexity is T times the cost of a single ApproxProject operation, given by Proposition 25. □

7.4 Matrix dictionary recovery SDP solvers

In this section, we develop general solvers for the types of structured "mixed packing-covering" problems defined in Section 7.1.3, which we collectively call matrix dictionary recovery SDPs.

In Section 7.4.1, we solve a basic version of this problem where $\mathbf{B} = \mathbf{I}$, i.e. the constraints are multiples of the identity. In Section 7.4.2, we give a more general solver able to handle arbitrary constraints, whose runtime depends polylogarithmically on the conditioning of said constraints. Our main results Theorems 42 and 43 are stated in Section 7.1.3, and we provide full proofs here.

7.4.1 Identity constraints

In this section, we consider the special case of the problem (7.7), (7.8) in which $\mathbf{B} = \mathbf{I}$. To solve this problem, we first develop a framework for solving the decision variant of the problem (7.7), (7.8). Given a set of matrices $\{\mathbf{M}_i\}_{i\in[n]} \in \mathbb{S}^d_{\geq 0}$ and a parameter $\kappa \geq 1$, we wish to determine

$$\text{does there exist } w \in \mathbb{R}^n_{\geq 0} \text{ such that } \lambda_{\max}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right) \leq \kappa \lambda_{\min}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right)? \tag{7.58}$$

We note that the problem (7.58) is a special case of the more general mixed packing-covering semidefinite programming problem defined in [285], with packing matrices $\{\mathbf{M}_i\}_{i\in[n]}$ and covering matrices $\{\kappa \mathbf{M}_i\}_{i\in[n]}$. We define an *$\epsilon$-tolerant tester* for the decision problem (7.58) to be an algorithm which returns "yes" whenever (7.58) is feasible for the value $(1-\epsilon)\kappa$ (along with weights $w \in \mathbb{R}^n_{\geq 0}$ certifying this feasibility), and "no" whenever it is infeasible for the value $(1+\epsilon)\kappa$ (and can return either answer in the middle range). After developing such a tester, we apply it to solve the (approximate) optimization variant (7.7), (7.8) by incrementally searching for the optimal κ.

To develop a tolerant tester for (7.58), we require access to an algorithm for solving the optimization variant of a pure packing SDP,

$$\mathrm{OPT}(v) := \max_{w\in\mathbb{R}^n_{\geq 0} \,:\, \sum_{i\in[n]} w_i \mathbf{M}_i \preceq \mathbf{I}} v^\top w. \tag{7.59}$$

The algorithm is based on combining a solver for the testing variant of (7.59), developed in Section 7.1.4 with a binary search. We state its guarantees as Proposition 26, and defer a proof to Appendix F.2.

Proposition 26. *Let* OPT_+ *and* OPT_- *be known upper and lower bounds on* $\mathrm{OPT}(v)$ *as in* (7.59). *There is an algorithm,* $\mathcal{A}_{\mathrm{pack}}$, *which succeeds with probability* $\geq 1-\delta$, *whose runtime is*

$$O\left(\mathcal{T}_{\mathrm{mv}}\left(\{\mathbf{M}_i\}_{i\in[n]}\right) \cdot \frac{\log^2(ndT(\delta\epsilon)^{-1})\log^2 d}{\epsilon^5}\right) \cdot T \text{ for } T = O\left(\log\log\frac{\mathrm{OPT}_+}{\mathrm{OPT}_-} + \log\frac{1}{\epsilon}\right),$$

and returns an ϵ-multiplicative approximation to $\mathrm{OPT}(v)$, and w attaining this approximation.

We require one additional tool, a regret analysis of matrix multiplicative weights from [21].

Proposition 27 (Theorem 3.1, [21])**.** *Consider a sequence of gain matrices $\{\mathbf{G}_t\}_{0\le t<T} \subset \mathbb{S}^d_{\ge 0}$, which all satisfy for step size $\eta > 0$, $\|\eta \mathbf{G}_t\|_2 \le 1$. Then iteratively defining (from $\mathbf{S}_0 := \mathbf{0}$)*

$$\mathbf{Y}_t := \frac{\exp(\mathbf{S}_t)}{\mathrm{Tr}\exp(\mathbf{S}_t)},\ \mathbf{S}_{t+1} := \mathbf{S}_t - \eta \mathbf{G}_t,$$

we have the bound for any $\mathbf{U} \in \mathbb{S}^d_{\ge 0}$ with $\mathrm{Tr}(\mathbf{U}) = 1$,

$$\frac{1}{T}\sum_{0\le t<T} \langle \mathbf{G}_t, \mathbf{Y}_t - \mathbf{U}\rangle \le \frac{\log d}{\eta T} + \frac{1}{T}\sum_{t\in[T]} \eta \left\|\mathbf{G}_t\right\|_2 \langle \mathbf{G}_t, \mathbf{Y}_t\rangle .$$

Finally, we are ready to state our ϵ-tolerant tester for the decision problem (7.58) as Algorithm 29.

Lemma 96. *Algorithm 29 meets its output guarantees (as specified on Line 2).*

Proof. Throughout, assume all calls to $\mathcal{A}_{\mathrm{pack}}$ and the computation of approximations as given by Lines 6 and 13 succeed. By union bounding over T iterations, this gives the failure probability.

We first show that if the algorithm terminates on Line 15, it is always correct. By the definition of $\mathcal{A}_{\mathrm{pack}}$, all $\mathbf{G}_t \preceq \kappa \mathbf{I}$, so throughout the algorithm, $-\mathbf{S}_{t+1} \preceq \eta\kappa T \mathbf{I} \preceq \frac{11\kappa \log d}{\epsilon}$. If the check on Line 14 passes, we must have $-\mathbf{S}_{t+1} \succeq \frac{11\log d}{\epsilon}\mathbf{I}$, and hence the matrix $-\frac{1}{t+1}\mathbf{S}_{t+1}$ has condition number at most κ. The conclusion follows since the reweighting $\bar{x}$ induces $\mathbf{S}_{t+1}$.

We next prove correctness in the "no" case. Suppose the problem (7.60) is feasible; we show that the check in Line 9 will never pass (so the algorithm never returns "no"). Let $v_t^\star$ be the vector which is entrywise exactly $\{\langle \mathbf{M}_i, \mathbf{Y}_t\rangle\}_{i\in[n]}$, and let v_t' be a $\frac{\epsilon}{10}$-multiplicative approximation to $v_t^\star$ such that v_t is an entrywise $\frac{\epsilon}{10n}$-additive approximation to v_t'. By definition, it is clear $\mathrm{OPT}(\kappa v_t') \ge (1 - \frac{\epsilon}{10})\mathrm{OPT}(\kappa v_t^\star)$. Moreover, by the assumption that all $\lambda_{\max}(\mathbf{M}_i) \ge 1$, all $w_i \le 1$ in the feasible region of the problem (7.59). Hence, the combined additive error incurred by the approximation $\langle \kappa v_t, w\rangle$ to $\langle \kappa v_t', w\rangle$ for any feasible w is $\frac{\epsilon}{10}$. All told, by the guarantee of $\mathcal{A}_{\mathrm{pack}}$,

$$\kappa \langle v_t, x_t\rangle \ge \left(1 - \frac{\epsilon}{10}\right)^2 \mathrm{OPT}(\kappa v_t^\star) - \frac{\epsilon}{10},\ \text{where } \mathrm{OPT}(\kappa v_t^\star) = \max_{\substack{\sum_{i\in[n]} w_i \mathbf{M}_i \preceq \mathbf{I} \\ w\in\mathbb{R}^n_{\ge 0}}} \kappa \left\langle \mathbf{Y}_t, \sum_{i\in[n]} w_i \mathbf{M}_i \right\rangle . \tag{7.62}$$

However, by feasibility of (7.60) and scale invariance, there exists a $w \in \mathbb{R}^n_{\ge 0}$ with $\sum_{i\in[n]} w_i \mathbf{M}_i \preceq \mathbf{I}$ and $(1-\epsilon)\kappa \sum_{i\in[n]} w_i \mathbf{M}_i \succeq \mathbf{I}$. Since $\mathbf{Y}_t$ has trace 1, this certifies $\mathrm{OPT}(\kappa v_t^\star) \ge \frac{1}{1-\epsilon}$, and thus

$$\kappa \langle v_t, x_t\rangle \ge \left(1 - \frac{\epsilon}{10}\right)^2 \cdot \frac{1}{1-\epsilon} - \frac{\epsilon}{10} > 1 - \frac{\epsilon}{5}.$$

Algorithm 29: DecideStructuredMPC($\{\mathbf{M}_i\}_{i\in[n]}, \kappa, \mathcal{A}_{\text{pack}}, \delta, \epsilon$)

Input: $\{\mathbf{M}_i\}_{i\in[n]} \in \mathbb{S}^{d\times d}_{\geq 0}$ such that $1 \leq \lambda_{\max}(\mathbf{M}_i) \leq 2$ for all $i \in [n]$, $\kappa > 1$, $\mathcal{A}_{\text{pack}}$ which on input $v \in \mathbb{R}^n_{\geq 0}$ returns $w \in \mathbb{R}^n_{\geq 0}$ satisfying (recalling definition (7.59))

$$\sum_{i\in[n]} w_i \mathbf{M}_i \preceq \mathbf{I},\ v^\top w \geq \left(1 - \frac{\epsilon}{10}\right) \text{OPT}(v),\ \text{with probability} \geq 1 - \frac{\delta}{2T} \text{ for some } T = O\left(\frac{\kappa \log d}{\epsilon^2}\right),$$

failure probability $\delta \in (0,1)$, tolerance $\epsilon \in (0,1)$;

Output: With probability $\geq 1 - \delta$: "yes" or "no" is returned. The algorithm must return "yes" if there exists $w \in \mathbb{R}^n_{\geq 0}$ with

$$\lambda_{\max}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right) \leq (1-\epsilon)\kappa \lambda_{\min}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right), \tag{7.60}$$

and if "yes" is returned, a vector w is given with

$$\lambda_{\max}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right) \leq (1+\epsilon)\kappa \lambda_{\min}\left(\sum_{i\in[n]} w_i \mathbf{M}_i\right). \tag{7.61}$$

;

$\eta \leftarrow \frac{\epsilon}{10\kappa}$, $T \leftarrow \left\lceil \frac{10 \log d}{\eta \epsilon} \right\rceil$, $\mathbf{Y}_0 \leftarrow \frac{1}{d}\mathbf{I}$, $\mathbf{S}_0 \leftarrow \mathbf{0}$;

for $0 \leq t < T$ **do**

  $\mathbf{Y}_t \leftarrow \frac{\exp(\mathbf{S}_t)}{\text{Tr}\exp(\mathbf{S}_t)}$;

  $v_t \leftarrow$ entrywise nonnegative $(\frac{\epsilon}{10}, \frac{\epsilon}{10\kappa n})$-approximations to $\{\langle \mathbf{M}_i, \mathbf{Y}_t\rangle\}_{i\in[n]}$, with probability $\geq 1 - \frac{\delta}{4T}$;

  $x_t \leftarrow \mathcal{A}_{\text{pack}}(\kappa v_t)$;

  $\mathbf{G}_t \leftarrow \kappa \sum_{i\in[n]} [x_t]_i \mathbf{M}_i$;

  if $\kappa \langle x_t, v_t\rangle < 1 - \frac{\epsilon}{5}$ **then**

    Return: "no";

  $\mathbf{S}_{t+1} \leftarrow \mathbf{S}_t - \eta \mathbf{G}_t$;

  $\tau \leftarrow \frac{\log d}{\epsilon}$-additive approximation to $\lambda_{\min}(-\mathbf{S}_{t+1})$, with probability $\geq 1 - \frac{\delta}{4T}$;

  if $\tau \geq \frac{12 \log d}{\epsilon}$ **then**

    Return: ("yes", $\bar{x}$) for $\bar{x} := \frac{1}{t+1} \sum_{0\leq s\leq t} x_s$;

Return: ("yes", $\bar{x}$) for $\bar{x} := \frac{1}{T} \sum_{0\leq t<T} x_t$;

Hence, whenever the algorithm returns "no" it is correct. Assume for the remainder of the proof that "yes" is returned on Line 18. Next, we observe that whenever $\mathcal{A}$ succeeds on iteration t, $\sum_{i\in[n]}[x_t]_i\mathbf{M}_i \preceq \mathbf{I}$, and hence in every iteration we have $\|\mathbf{G}_t\|_2 \le \kappa$. Proposition 27 then gives

$$\frac{1}{T}\sum_{0\le t<T}\langle \mathbf{G}_t, \mathbf{Y}_t - \mathbf{U}\rangle \le \frac{\log d}{\eta T} + \frac{1}{T}\sum_{t\in[T]}\eta\|\mathbf{G}_t\|_2\langle \mathbf{G}_t, \mathbf{Y}_t\rangle, \text{ for all } \mathbf{U}\in\mathbb{S}^d_{\ge 0} \text{ with } \mathrm{Tr}(\mathbf{U}) = 1.$$

Rearranging the above display, using $\eta\|\mathbf{G}_t\|_2 \le \frac{\epsilon}{10}$, and minimizing over $\mathbf{U}$ yields

$$\lambda_{\min}\left(\frac{1}{T}\sum_{0\le t<T}\mathbf{G}_t\right) \ge \frac{1-\frac{\epsilon}{10}}{T}\sum_{0\le t<T}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle - \frac{\log d}{\eta T} \ge \frac{1-\frac{\epsilon}{10}}{T}\sum_{0\le t<T}\langle \mathbf{G}_t, \mathbf{Y}_t\rangle - \frac{\epsilon}{10}.$$

The last inequality used the definition of T. However, by definition of v_t, we have for all $0 \le t < T$,

$$\langle \mathbf{Y}_t, \mathbf{G}_t\rangle = \kappa\sum_{i\in[n]}[x_t]_i\langle \mathbf{M}_i, \mathbf{Y}_t\rangle \ge \left(1-\frac{\epsilon}{10}\right)\kappa\langle x_t, v_t\rangle \ge \left(1-\frac{\epsilon}{10}\right)\left(1-\frac{\epsilon}{5}\right) \ge 1 - \frac{3\epsilon}{10}. \tag{7.63}$$

The second-to-last inequality used that Line 9 did not pass. Combining the previous two displays,

$$\kappa\lambda_{\min}\left(\sum_{i\in[n]}\bar{x}_i\mathbf{M}_i\right) = \lambda_{\min}\left(\frac{1}{T}\sum_{0\le t<T}\mathbf{G}_t\right) \ge \left(1-\frac{\epsilon}{10}\right)\left(1-\frac{3\epsilon}{10}\right) - \frac{\epsilon}{10} \ge 1 - \frac{\epsilon}{2}.$$

On the other hand, since all $0 \le t < T$ have $\sum_{i\in[n]}[x_t]_i\mathbf{M}_i \preceq \mathbf{I}$, by convexity $\sum_{i\in[n]}\bar{x}_i\mathbf{M}_i \preceq \mathbf{I}$. Combining these two guarantees and $(1+\epsilon)(1-\frac{\epsilon}{2}) \ge 1$ shows $\bar{x}$ is correct for the "yes" case. □

We bound the runtime complexities of Lines 6, 7, and 13 of Algorithm 29 in the following sections.

Approximating inner products

In this section, we bound the complexity of Line 6 of Algorithm 29. We will use the following two standard helper results on random projections and approximation theory.

Fact 11 (Johnson-Lindenstrauss [160])**.** *For $0 \le \epsilon \le 1$, let $k = \Theta\left(\frac{1}{\epsilon^2}\log\frac{d}{\delta}\right)$ for an appropriate constant. For $\mathbf{Q}\in\mathbb{R}^{k\times d}$ with independent uniformly random unit vector rows in $\mathbb{R}^d$ scaled down by $\frac{1}{\sqrt{k}}$, with probability $\ge 1-\delta$ for any fixed $v\in\mathbb{R}^d$,*

$$(1-\epsilon)\|\mathbf{Q}v\|_2^2 \le \|v\|_2^2 \le (1+\epsilon)\|\mathbf{Q}v\|_2^2.$$

Fact 12 (Polynomial approximation of exp [466], Theorem 4.1)**.** *Let $\mathbf{M}\in\mathbb{S}^d_{\ge 0}$ have $\mathbf{M}\preceq R\mathbf{I}$. Then for any $\delta > 0$, there is an explicit polynomial p of degree $O(\sqrt{R\log\frac{1}{\delta} + \log^2\frac{1}{\delta}})$ with*

$$\exp(-\mathbf{M}) - \delta\mathbf{I} \preceq p(\mathbf{M}) \preceq \exp(-\mathbf{M}) + \delta\mathbf{I}.$$

We also state a simple corollary of Fact 12.

Corollary 27. *Given $R > 1$, $\mathbf{M} \in \mathbb{S}^d_{\geq 0}$, and κ with $\mathbf{M} \preceq \kappa \mathbf{I}$, we can compute a degree-$O(\sqrt{\kappa R} + R)$ polynomial p such that for $\mathbf{P} = p(\mathbf{M})$,*

$$\exp(-\mathbf{M}) - \exp(-R)\,\mathbf{I} \preceq \mathbf{P} \preceq \exp(-\mathbf{M}) + \exp(-R)\,\mathbf{I}.$$

Using these tools, we next demonstrate that we can efficiently approximate the trace of a negative exponential of a bounded matrix, and quadratic forms through it.

Lemma 97. *Given $\mathbf{M} \in \mathbb{S}^d_{\geq 0}$, $R, \kappa, \epsilon > 0$ such that $\lambda_{\min}(\mathbf{M}) \leq R$ and $\lambda_{\max}(\mathbf{M}) \leq \kappa R$, $\delta \in (0,1)$, we can compute an ϵ-multiplicative approximation to $\operatorname{Tr}\exp(-\mathbf{M})$ with probability $\geq 1 - \delta$ in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{M}) \cdot \sqrt{\kappa} R \cdot \frac{\log \frac{d}{\delta}}{\epsilon^2}\right).$$

Proof. First, with probability at least $1 - \delta$, choosing $k = O(\frac{1}{\epsilon^2} \log \frac{d}{\delta})$ in Fact 11 and taking a union bound guarantees that for all rows $j \in [d]$, we have

$$\left\|\mathbf{Q}\left[\exp\left(-\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2 \text{ is a } \frac{\epsilon}{3}\text{-multiplicative approximation of } \left\|\left[\exp\left(-\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2.$$

Condition on this event in the remainder of the proof. The definition

$$\operatorname{Tr}\exp(-\mathbf{M}) = \sum_{j \in [d]} \left\|\left[\exp\left(-\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2,$$

and the sequence of equalities

$$\begin{aligned}\sum_{j \in [d]} \left\|\mathbf{Q}\left[\exp\left(-\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2 &= \operatorname{Tr}\left(\exp\left(-\frac{1}{2}\mathbf{M}\right)\mathbf{Q}^\top \mathbf{Q} \exp\left(-\frac{1}{2}\mathbf{M}\right)\right) \\ &= \operatorname{Tr}\left(\mathbf{Q}\exp(-\mathbf{M})\,\mathbf{Q}^\top\right) = \sum_{\ell \in [k]} \left\|\exp\left(-\frac{1}{2}\mathbf{M}\right)\mathbf{Q}_{\ell:}\right\|_2^2,\end{aligned}$$

implies that it suffices to obtain a $\frac{\epsilon}{3}$-multiplicative approximation to the last sum in the above display. Since $\operatorname{Tr}\exp(-\mathbf{M}) \geq \exp(-R)$ by the assumption on $\lambda_{\min}(\mathbf{M})$, it then suffices to approximate each term $\left\|\exp(-\frac{1}{2}\mathbf{M})\mathbf{Q}_{\ell:}\right\|_2^2$ to an additive $\frac{\epsilon}{3k}\exp(-R)$. For simplicity, fix some $\ell \in [k]$ and denote $q := \mathbf{Q}_{\ell:}$; recall $\|q\|_2^2 = \frac{1}{k}$ from the definition of $\mathbf{Q}$ in Fact 11.

By rescaling, it suffices to demonstrate that on any unit vector $q \in \mathbb{R}^d$, we can approximate $\left\|\exp(-\frac{1}{2}\mathbf{M})q\right\|_2^2$ to an additive $\frac{\epsilon}{3}\exp(-R)$. To this end, we note that (after shifting the definition of

R by a constant) Corollary 27 provides a matrix $\mathbf{P}$ with $\mathcal{T}_{\text{mv}}(\mathbf{P}) = O(\mathcal{T}_{\text{mv}}(\mathbf{M}) \cdot \sqrt{\kappa} R)$ and

$$\exp(-\mathbf{M}) - \frac{\epsilon}{3}\exp(-R)\,\mathbf{I} \preceq \mathbf{P} \preceq \exp(-\mathbf{M}) + \frac{\epsilon}{3}\exp(-R)\,\mathbf{I},$$

which exactly meets our requirements by taking quadratic forms. The runtime follows from the cost of applying $\mathbf{P}$ to each of the $k = O(\frac{1}{\epsilon^2}\log\frac{d}{\delta})$ rows of $\mathbf{Q}$. □

Lemma 98. *Given $\mathbf{M} \in \mathbb{S}^d_{\geq 0}$ and κ with $\mathbf{M} \preceq \kappa\mathbf{I}$, $1 > c > 0$, $\delta \in (0,1)$, $\epsilon > 0$, and a set of matrices $\{\mathbf{M}_i\}_{i\in[n]}$ with decompositions of the form (7.9) and $1 \leq \lambda_{\max}(\mathbf{M}_i) \leq 2$ for all $i \in [n]$, we can compute (ϵ, c)-approximations to all*

$$\{\langle \mathbf{M}_i, \exp(-\mathbf{M})\rangle\}_{i\in[n]},$$

with probability $\geq 1 - \delta$ in time

$$O\left(\mathcal{T}_{\text{mv}}\left(\mathbf{M}, \{\mathbf{V}_i\}_{i\in[n]}\right) \cdot \sqrt{\kappa}\log\frac{c}{m} \cdot \frac{\log\frac{mn}{\delta}}{\epsilon^2}\right).$$

Proof. First, observe that for all $i \in [n]$, letting $\{v_j^{(i)}\}_{j\in[m]}$ be columns of $\mathbf{V}_i \in \mathbb{R}^{d\times m}$, we have

$$\langle \mathbf{M}_i, \exp(-\mathbf{M})\rangle = \sum_{j\in[m]} \left(v_j^{(i)}\right)^\top \exp(-\mathbf{M}) \left(v_j^{(i)}\right).$$

Hence, to provide an (ϵ, c) approximation to $\langle \mathbf{M}_i, \exp(-\mathbf{M})\rangle$ it suffices to provide, for all $i \in [n]$, $j \in [m]$, an $(\epsilon, \frac{c}{m})$-approximation to $\left(v_j^{(i)}\right)^\top \exp(-\mathbf{M}) \left(v_j^{(i)}\right)$. As in the proof of Lemma 97, by taking a union bound it suffices to sample a $\mathbf{Q} \in \mathbb{R}^{k\times d}$ for $k = O(\frac{1}{\epsilon^2}\log\frac{mn}{\delta})$ and instead compute all $\|\mathbf{Q}\exp(-\frac{1}{2}\mathbf{M})v_j^{(i)}\|_2^2$ to additive error $\frac{c}{m}$. We will instead show how to approximate, for arbitrary vectors q, v with norm at most 1,

$$\left\langle q, \exp\left(-\frac{1}{2}\mathbf{M}\right) v\right\rangle^2 \text{ to additive error } \frac{c}{2m}.$$

By letting q range over rows of $\mathbf{Q}$ renormalized by $\sqrt{k}$, and scaling all $v_j^{(i)}$ by a factor of $\sqrt{2}$, this yields the desired result. To this end, consider using $\langle q, \mathbf{P}v\rangle^2$ for some $\mathbf{P}$ with $-\frac{c}{6m}\mathbf{I} \preceq \mathbf{P} - \exp(-\frac{1}{2}\mathbf{M}) \preceq \frac{c}{6m}\mathbf{I}$. Letting the difference matrix be $\mathbf{D} := \mathbf{P} - \exp(-\frac{1}{2}\mathbf{M})$, we compute

$$\begin{aligned}\left(q^\top \exp\left(-\frac{1}{2}\mathbf{M}\right) v\right)^2 - \left(q^\top \mathbf{P} v\right)^2 &= 2\left(q^\top \exp\left(-\frac{1}{2}\mathbf{M}\right) v\right)\left(q^\top \mathbf{D} v\right) + \left(q^\top \mathbf{D} v\right)^2 \\ &\leq 2\|\mathbf{D}\|_2 + \|\mathbf{D}\|_2^2 \leq \frac{c}{2m}.\end{aligned}$$

We used q and v have ℓ_2 norm at most 1, $\exp(-\frac{1}{2}\mathbf{M}) \preceq \mathbf{I}$, and $\|\mathbf{D}\|_2 \leq \frac{c}{6m} \leq 1$. Hence, $\langle q, \mathbf{P}v\rangle^2$ is a

valid approximation. The requisite $\mathbf{P}$ is given by Corollary 27 with $\mathcal{T}_{\text{mv}}(\mathbf{P}) = O(\mathcal{T}_{\text{mv}}(\mathbf{M}) \cdot \sqrt{\kappa} \log \frac{c}{m})$.

Finally, the runtime follows from first applying $\mathbf{P}$ to rows of $\mathbf{Q}$ to explicitly form $\widetilde{\mathbf{Q}}$ with k rows, and then computing all $\|\widetilde{\mathbf{Q}} v_j^{(i)}\|_2^2$ for all $i \in [n]$, $j \in [m]$. □

Combining Lemmas 97 and 98, we bound the cost of Line 6 in Algorithm 29.

Lemma 99. *We can implement Line 6 of Algorithm 29 in time*

$$O\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]}\right) \cdot \sqrt{\kappa} \cdot \frac{\log^3(\frac{mnd\kappa}{\delta\epsilon})}{\epsilon^3}\right).$$

Proof. Since $-\mathbf{S}_t$ is an explicit linear combination of $\{\mathbf{M}_i\}_{i\in[n]}$, we have $\mathcal{T}_{\text{mv}}(\mathbf{S}_t) = O(\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]}))$. We first obtain a $\frac{\epsilon}{30}$ approximation to the denominator in Line 6 within the required time by applying Lemma 97 with $\kappa \leftarrow O(\kappa)$, $R \leftarrow O(\frac{\log d}{\epsilon})$, $\epsilon \leftarrow \frac{\epsilon}{30}$, and adjusting the failure probability by $O(T)$. The bound on $\lambda_{\min}(-\mathbf{S}_t)$ comes from the check on Line 13 and the algorithm yields the bound on $\lambda_{\max}(-\mathbf{S}_t)$. Next, we obtain a $(\frac{\epsilon}{30}, \frac{\epsilon}{30\kappa n})$ approximation to each numerator in Line 6 within the required time by using Lemma 98 with $\kappa \leftarrow O(\frac{\kappa \log d}{\epsilon})$ and adjusting constants appropriately. Combining these approximations to the numerators and denominator yields the result. □

Implementing a packing oracle

In this section, we bound the complexity of Line 7 of Algorithm 29 by using Proposition 26.

Lemma 100. *We can implement Line 7 of Algorithm 29 in time*

$$O\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]}\right) \cdot \frac{\log^5(\frac{nd\kappa}{\delta\epsilon})}{\epsilon^5}\right).$$

Proof. First, we observe that the proof of the "no" case in Lemma 96 demonstrates that in all calls to $\mathcal{A}$, we can set our lower bound $\text{OPT}_- = 1 - O(\epsilon)$, since the binary search of Proposition 26 will never need to check smaller values to determine whether the test on Line 9 passes. On the other hand, the definition of $\text{OPT}(\kappa v_t^\star)$ in (7.62), as well as $\text{OPT}(\kappa v_t) \le (1 + \frac{\epsilon}{10})\text{OPT}(\kappa v_t^\star)$ by the multiplicative approximation guarantee, shows that it suffices to set $\text{OPT}_+ \le (1 + O(\epsilon))\kappa$.

We will use the algorithm of Proposition 26 as $\mathcal{A}_{\text{pack}}$ in Algorithm 29. In our setting, we argued $\frac{\text{OPT}_+}{\text{OPT}_-} = O(\kappa)$, giving the desired runtime bound via Proposition 26. □

Approximating the smallest eigenvalue

In this section, we bound the complexity of Line 13 of Algorithm 29, which asks to approximate the smallest eigenvalue of a matrix $\mathbf{M}$ to additive error. At a high level, our strategy is to use the power method on the negative exponential $\exp(-\mathbf{M})$, which we approximate to additive error via Corollary 27. We first state a guarantee on the power method from [404].

Fact 13 (Theorem 1, [404]). *For any $\delta \in (0,1)$ and $\mathbf{M} \in \mathbb{S}^d_{\succeq 0}$, there is an algorithm, $\mathsf{Power}(\mathbf{M}, \delta)$, which returns with probability at least $1-\delta$ a value V such that $\lambda_{\max}(\mathbf{M}) \geq V \geq 0.9\lambda_{\max}(\mathbf{M})$. The algorithm runs in time $O(\mathcal{T}_{\mathrm{mv}}(\mathbf{M}) \log \frac{d}{\delta})$, and is performed as follows:*

1. *Let $u \in \mathbb{R}^d$ be a random unit vector.*
2. *For some $\Delta = O(\log \frac{d}{\delta})$, let $v \leftarrow \frac{\mathbf{M}^\Delta u}{\|\mathbf{M}^\Delta u\|_2}$.*
3. *Return $\|\mathbf{M}v\|_2$.*

Lemma 101. *We can implement Line 13 of Algorithm 29 in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]}\right) \cdot \sqrt{\kappa} \cdot \frac{\log^2(\frac{d\kappa}{\delta\epsilon})}{\epsilon}\right).$$

Proof. Throughout, denote $\mathbf{M} := -\mathbf{S}_{t+1}$, $L := \lambda_{\min}(\mathbf{M})$ and $R := \frac{\log d}{\epsilon}$, and note that (assuming all previous calls succeeded), we must have $L \leq 14R$ since the previous iteration had $L \leq 13R$ and $\mathbf{S}_t$ is changing by an $O(\epsilon)$-spectrally bounded matrix each iteration. It suffices to obtain V with

$$0.9(\exp(-L) - \exp(-20R)) \leq V \leq \exp(-L), \tag{7.64}$$

and then return $-\log(V)$, to obtain an R-additive approximation to L. To see this, it is immediate that $-\log(V) \geq L$ from the above display. Moreover, for the given ranges of L and R, it is clear

$$\begin{aligned}
&\exp(-20R + L) \leq \exp(-6R) \leq 1 - \frac{3}{2R} \\
\implies &\exp(-L) - \exp(-20R) \geq \exp(-L) \cdot \frac{3}{2R} \\
\implies &\exp(-L) - \exp(-20R) \geq \exp\left(-L - \frac{2R}{3}\right) \\
\implies &\log\left(\frac{1}{\exp(-L) - \exp(-20R)}\right) \leq L + \frac{2R}{3}.
\end{aligned}$$

Combining with

$$-\log(V) \leq \log\left(\frac{1}{\exp(-L) - \exp(-20R)}\right) + \log\frac{10}{9} \leq \log\left(\frac{1}{\exp(-L) - \exp(-20R)}\right) + \frac{R}{3}$$

yields the claim. It hence suffices to provide V satisfying (7.64) in the requisite time. To do so, we first use Corollary 27 with $\kappa \leftarrow O(\frac{\kappa \log d}{\epsilon})$ to produce $\mathbf{P}$ with $\mathcal{T}_{\mathrm{mv}}(\mathbf{P}) = O(\mathcal{T}_{\mathrm{mv}}(\mathbf{M}) \cdot \sqrt{\kappa} \cdot \frac{\log d}{\epsilon})$ and

$$\exp(-\mathbf{M}) - \frac{1}{2}\exp(-20R)\mathbf{I} \preceq \mathbf{P} \preceq \exp(-\mathbf{M}) + \frac{1}{2}\exp(-20R)\mathbf{I}.$$

The conclusion follows by applying Fact 40 to $\mathbf{P} - \frac{1}{2}\exp(-20R)\mathbf{I}$ and adjusting δ. □

Runtime of the optimization variant

Finally, we put these pieces together to solve the optimization variant of (7.7), (7.8). We begin by stating the runtime of Algorithm 29, which follows from combining Lemmas 99, 100, and 101.

Corollary 28. *Algorithm 29 can be implemented in time*

$$O\left(\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}) \cdot \kappa^{1.5} \cdot \frac{\log^6(\frac{mnd\kappa}{\delta\epsilon})}{\epsilon^7}\right).$$

Theorem 42. *Given matrices $\{\mathbf{M}_i\}_{i\in[n]}$ with explicit factorizations* (7.9)*, such that* (7.7) *is feasible for $\mathbf{B} = \mathbf{I}$ and some $\kappa \geq 1$, we can return weights $w \in \mathbb{R}^n_{\geq 0}$ satisfying* (7.8) *with probability $\geq 1 - \delta$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]}) \cdot \kappa^{1.5} \cdot \frac{\log^6(\frac{mnd\kappa}{\delta\epsilon})}{\epsilon^8}\right).$$

Proof. First, to guarantee all $\mathbf{M}_i$ satisfy $1 \leq \lambda_{\max}(\mathbf{M}_i) \leq 2$, we exploit the scale-invariance of the problem (7.7), (7.8) and rescale each matrix by a 2-approximation to its largest eigenvalue. This can be done with Fact 40 and does not bottleneck the runtime.

Next, we perform an incremental search on κ initialized at 1, and increasing in multiples of $1 + \frac{\epsilon}{3}$, returning the first guess of κ such that Algorithm 29 returns "yes." This will yield an ϵ-approximation to the optimal κ, and the runtime follows from combining Corollary 28 with the overhead of this incremental search, which runs $O(\frac{\log \kappa}{\epsilon})$ times; we do not lose an extra logarithmic factor from the search due to the geometric sequence $\kappa^{1.5} + \kappa^{1.5}(1 - O(\epsilon)) + \ldots$ summing to $O(\frac{\kappa}{\epsilon})$. □

7.4.2 General constraints

In this section, we consider a more general setting in which there is a constraint matrix $\mathbf{B}$ in the problem (7.7), (7.8) which we have matrix-vector product access to, but we cannot efficiently invert $\mathbf{B}$. We show that we can obtain a runtime similar to that in Theorem 42 (up to logarithmic factors and one factor of $\sqrt{\kappa}$), with an overhead depending polylogarithmically on α and β defined in (10.32) and restated here for convenience:

$$\mathbf{B} \preceq \mathcal{M}(\mathbf{1}) \preceq \alpha\mathbf{B},\ \mathbf{I} \preceq \mathbf{B} \preceq \beta\mathbf{I}.$$

Homotopy method preliminaries

Our algorithm will be a "homotopy method" which iteratively finds reweightings of the $\{\mathbf{M}_i\}_{i\in[n]}$ which $(1+\epsilon)\kappa$-approximate $\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})$, for a sequence of λ values. More specifically, we first bound the required range of λ via two simple observations.

Lemma 102. *Let $\lambda \geq \frac{1}{\epsilon}$. Then,*

$$\mathbf{B} + \lambda\mathcal{M}(\mathbf{1}) \preceq \sum_{i\in[n]} (1+\lambda)\mathbf{M}_i \preceq (1+\epsilon)\left(\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})\right).$$

Proof. The first inequality is immediate from (10.32). The second follows from $\mathbf{B} \succeq \mathbf{0}$. □

Lemma 103. *Let $\lambda \leq \frac{\epsilon\kappa}{2\alpha}$, and let $w \in \mathbb{R}^n_{\geq 0}$ satisfy*

$$\mathbf{B} + \lambda\mathcal{M}(\mathbf{1}) \preceq \sum_{i\in[n]} w_i\mathbf{M}_i \preceq (1+\epsilon)\kappa\left(\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})\right).$$

Then, the same w satisfies

$$\mathbf{B} \preceq \sum_{i\in[n]} w_i\mathbf{M}_i \preceq (1+2\epsilon)\kappa\mathbf{B}.$$

Proof. The first inequality is immediate. The second is equivalent to

$$\epsilon\kappa\mathbf{B} \succeq (1+\epsilon)\lambda\mathcal{M}(\mathbf{1})$$

which follows from $\epsilon\kappa \geq (1+\epsilon)\lambda\alpha$ and (10.32). □

Our homotopy method is driven by the observation that if (7.7) is feasible for a value of κ, then it is also feasible for the same κ when $\mathbf{B}$ is replaced with $\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})$ for any $\lambda \geq 0$.

Lemma 104. *For any $\lambda \geq 0$, if (7.7) is feasible for $\kappa \geq 1$, there also exists $w^\star_\lambda \in \mathbb{R}^n_{\geq 0}$ with*

$$\mathbf{B} + \lambda\mathcal{M}(\mathbf{1}) \preceq \sum_{i\in[n]} [w^\star_\lambda]_i\mathbf{M}_i \preceq \kappa\left(\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})\right).$$

Proof. It suffices to choose $w^\star_\lambda = w^\star + \lambda\mathbf{1}$ where $w^\star$ is feasible for (7.7). □

To this end, our algorithm will proceed in K phases, where $K = \lceil \log_2 \frac{2\alpha}{\epsilon^2\kappa} \rceil$, setting

$$\lambda^{(0)} = \frac{1}{\epsilon},\ \lambda^{(k)} = \frac{\lambda^{(0)}}{2^k} \text{ for all } 0 \leq k \leq K.$$

In each phase k for $0 \leq k \leq K$, we will solve the problem (7.7), (7.8) for $\mathbf{B} \leftarrow \mathbf{B} + \lambda^{(k)}\mathcal{M}(\mathbf{1})$. To do so, we will use the fact that from the previous phase we have access to a matrix which is a 3κ-spectral approximation to $\mathbf{B}$ which we can efficiently invert.

Lemma 105. *Suppose for some $\lambda \geq 0$, $\kappa \geq 1$, and $0 < \epsilon \leq \frac{1}{2}$, we have $w \in \mathbb{R}^n_{\geq 0}$ such that*

$$\mathbf{B} + \lambda\mathcal{M}(\mathbf{1}) \preceq \sum_{i\in[n]} w_i\mathbf{M}_i \preceq (1+\epsilon)\kappa(\mathbf{B} + \lambda\mathcal{M}(\mathbf{1})).$$

Then defining $\mathbf{D} := \mathcal{M}(w)$*, we have*

$$\mathbf{B} + \frac{\lambda}{2}\mathcal{M}(\mathbf{1}) \preceq \mathbf{D} \preceq 3\kappa \left(\mathbf{B} + \frac{\lambda}{2}\mathcal{M}(\mathbf{1})\right).$$

Proof. This is immediate from the assumption and

$$\mathbf{B} + \lambda\mathcal{M}(\mathbf{1}) \preceq 2\left(\mathbf{B} + \frac{\lambda}{2}\mathcal{M}(\mathbf{1})\right).$$

□

Now, Lemma 102 shows we can access weights satisfying (7.8) for $\mathbf{B} \leftarrow \mathbf{B} + \lambda^{(0)}\mathcal{M}(\mathbf{1})$, and Lemma 103 shows if we can iteratively compute weights satisfying (7.8) for each $\mathbf{B} + \lambda^{(k)}\mathcal{M}(\mathbf{1})$, $0 \leq k \leq K$, then we solve the original problem up to a constant factor in ϵ. Moreover, Lemma 104 implies that the problem (7.7) is feasible for all phases assuming it is feasible with some value of κ for the original problem. Finally, Lemma 105 shows that we start each phase with a matrix $\mathcal{M}(w)$ which we can efficiently invert using the linear operator in (7.10), which is a 3κ-spectral approximation to the constraint matrix. We solve this self-contained subproblem in the following section using rational approximations to the square root, and an appropriate call to Theorem 42.

Inverse square root approximations with a preconditioner

In this section, we show how to efficiently approximate the inverse square root of some $\mathbf{B} \in \mathbb{S}^d_{\succ \mathbf{0}}$ given access to a matrix $\mathcal{M}(w) \in \mathbb{S}^d_{\succ \mathbf{0}}$ satisfying

$$\mathbf{B} \preceq \mathcal{M}(w) \preceq \kappa\mathbf{B}$$

and supporting efficient inverse access in the form of (7.10). For brevity in this section we will use $\widetilde{\mathcal{M}}_{\lambda,\epsilon}$ to denote the linear operator guaranteeing (7.10) for $\mathcal{M}(w) + \lambda\mathbf{I}$. Given this access, we will develop a subroutine for efficiently applying a linear operator $\mathbf{R} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that

$$\left\|(\mathbf{R} - \mathbf{B}^{-\frac{1}{2}})v\right\|_2 \leq \epsilon \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2 \text{ for all } v \in \mathbb{R}^d. \tag{7.65}$$

We will also use $\mathcal{T}_{\text{mv}}$ to denote $\mathcal{T}_{\text{mv}}(\mathbf{B}) + \mathcal{T}_{\text{mv}}(\mathcal{M}(w)) + \mathcal{T}^{\text{sol}}_{\mathcal{M}}$ for brevity in this section, where $\mathcal{T}^{\text{sol}}_{\mathcal{M}}$ is the cost of solving a system in $\mathcal{M}(w)$ to constant accuracy (see (7.10)). Our starting point is the following result in the literature on preconditioned accelerated gradient descent, which shows we can efficiently solve linear systems in combinations of $\mathbf{B}$ and $\mathbf{I}$.

Lemma 106. *Given any* $\lambda \geq 0$ *and* $\epsilon > 0$*, we can compute a linear operator* $\widehat{\mathcal{M}}_{\lambda,\epsilon}$ *such that*

$$\left\|(\widehat{\mathcal{M}}_{\lambda,\epsilon} - (\mathbf{B} + \lambda\mathbf{I})^{-1})v\right\|_2 \leq \epsilon \left\|(\mathbf{B} + \lambda\mathbf{I})^{-1}v\right\|_2 \text{ for all } v \in \mathbb{R}^d,$$

and

$$\mathcal{T}_{\text{mv}}(\widehat{\mathcal{M}}_{\lambda,\epsilon}) = O\left(\mathcal{T}_{\text{mv}} \cdot \sqrt{\kappa} \log \kappa \log \frac{1}{\epsilon}\right).$$

Proof. This follows from Theorem 4.4 in [292], the fact that $\mathcal{M}(w)+\lambda\mathbf{I}$ is a κ-spectral approximation to $\mathbf{B}+\lambda\mathbf{I}$, and our assumed linear operator $\widetilde{\mathcal{M}}_{\lambda,(10\kappa)^{-1}}$ which solves linear systems in $\mathcal{M}(w)+\lambda\mathbf{I}$ to relative error $\frac{1}{10\kappa}$ which can be applied in time $O(\mathcal{T}_{\text{mv}} \cdot \log \kappa)$ (the assumption (7.10)). □

We hence can apply and (approximately) invert matrices of the form $\mathbf{B}+\lambda\mathbf{I}$. We leverage this fact to approximate inverse square roots using the following approximation result.

Proposition 28 (Lemma 13, [298])**.** *For any matrix $\mathbf{B}$ and $\epsilon \in (0,1)$, there is a rational function $r(\mathbf{B})$ of degree $L = O(\log \frac{1}{\epsilon} \log \kappa(\mathbf{B}))$ such that for all vectors $v \in \mathbb{R}^d$,*

$$\left\|(r(\mathbf{B}) - \mathbf{B}^{-\frac{1}{2}})v\right\|_2 \leq \epsilon \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2.$$

The rational function $r(\mathbf{B})$ has the form, for $\{\lambda_\ell, \nu_\ell\}_{\ell\in[L]} \cup \{\nu_{L+1}\} \subset \mathbb{R}_{\geq 0}$ computable in $O(L)$ time,

$$r(\mathbf{B}) = \left(\prod_{\ell\in[L]} (\mathbf{B}+\lambda_\ell\mathbf{I})\right)\left(\prod_{\ell\in[L+1]} (\mathbf{B}+\nu_\ell\mathbf{I})^{-1}\right). \tag{7.66}$$

We note that Proposition 28 as it is stated in [298] is for the square root (not the inverse square root), but these are equivalent since all terms in the rational approximation commute. We show how to use Lemma 106 to efficiently approximate the inverse denominator in (7.66).

Lemma 107. *For any vector $v \in \mathbb{R}^d$, $\{\nu_\ell\}_{\ell\in[L+1]} \subset \mathbb{R}_{\geq 0}$ and $\epsilon > 0$, we can compute a linear operator $\widehat{\mathbf{D}}_\epsilon$ such that for the denominator of (7.66), denoted*

$$\mathbf{D} := \prod_{\ell\in[L+1]} (\mathbf{B}+\nu_\ell\mathbf{I}),$$

we have

$$\left\|(\widehat{\mathbf{D}}_\epsilon - \mathbf{D}^{-1})v\right\|_2 \leq \epsilon \left\|\mathbf{D}^{-1}v\right\|_2,$$

and

$$\mathcal{T}_{\text{mv}}(\widehat{\mathbf{D}}_\epsilon) = O\left(\mathcal{T}_{\text{mv}} \cdot \sqrt{\kappa} \cdot L^2 \log(\kappa) \log\left(\frac{\kappa L \cdot \kappa(\mathbf{B})}{\epsilon}\right)\right).$$

Proof. We give our linear operator $\widehat{\mathbf{D}}_\epsilon$ in the form of an algorithm, which applies a sequence of linear operators to v in the alloted time. Denote $\mathbf{B}_\ell := \mathbf{B}+\nu_\ell\mathbf{I}$ for all $\ell \in [L+1]$. We also define

$$\mathbf{\Pi}_\ell := \prod_{i\in[\ell]} \mathbf{B}_i^{-1} \text{ for all } \ell \in [L+1], \text{ and } \mathbf{\Pi}_0 := \mathbf{I}.$$

We define a sequence of vectors $\{v_\ell\}_{0\le\ell\le L+1}$ as follows: let $v_0 := v$, and for all $\ell \in [L+1]$ define

$$v_\ell \leftarrow \widehat{\mathcal{M}}_{\nu_{L+2-\ell},\Delta} v_{\ell-1} \text{ for } \Delta := \frac{\epsilon}{3(L+1)\kappa(\mathbf{B})^{2(L+1)}}.$$

Here we use notation from Lemma 106. In particular, for the given Δ, we have for all $\ell \in [L+1]$,

$$\left\|v_\ell - \mathbf{B}_{L+2-\ell}^{-1} v_{\ell-1}\right\|_2 \le \Delta \left\|v_{\ell-1}\right\|_2. \tag{7.67}$$

Our algorithm will simply return v_{L+1}, which is the application of a linear operator to v within the claimed runtime via Lemma 106. We now prove correctness. We first prove a bound on the sizes of this iterate sequence. By the triangle inequality and (7.67), for each $\ell \in [L+1]$ we have

$$\left\|v_\ell\right\|_2 \le \left\|v_\ell - \mathbf{B}_{L+2-\ell}^{-1} v_{\ell-1}\right\|_2 + \left\|\mathbf{B}_{L+2-\ell}^{-1} v_{\ell-1}\right\|_2 \le (1+\Delta)\left\|\mathbf{B}_{L+2-\ell}^{-1}\right\|_2 \left\|v_{\ell-1}\right\|_2.$$

Applying this bound inductively, we have that

$$\left\|v_\ell\right\|_2 \le (1+\Delta)^\ell \left(\prod_{i\in[\ell]} \left\|\mathbf{B}_{L+2-\ell}^{-1}\right\|_2\right) \left\|v_0\right\|_2 \le 3\left(\prod_{i\in[\ell]} \left\|\mathbf{B}_{L+2-\ell}^{-1}\right\|_2\right) \left\|v_0\right\|_2. \tag{7.68}$$

Next, we expand by the triangle inequality that

$$\begin{aligned}
\left\|v_{L+1} - \mathbf{\Pi}_{L+1} v_0\right\|_2 &\le \sum_{\ell\in[L+1]} \left\|\mathbf{\Pi}_{L+1-\ell} v_\ell - \mathbf{\Pi}_{L+2-\ell} v_{\ell-1}\right\|_2 \\
&\le \sum_{\ell\in[L+1]} \left\|\mathbf{\Pi}_{L+1-\ell}\right\|_2 \left\|v_\ell - \mathbf{B}_{L+2-\ell}^{-1} v_{\ell-1}\right\|_2 \\
&\le \sum_{\ell\in[L+1]} \Delta \left\|\mathbf{\Pi}_{L+2-\ell}\right\|_2 \left\|v_{\ell-1}\right\|_2 \\
&\le \sum_{\ell\in[L+1]} 3\Delta \left\|\mathbf{\Pi}_{L+1}\right\|_2 \left\|v_0\right\|_2 \le 3\Delta(L+1)\left\|\mathbf{\Pi}_{L+1}\right\|_2 \left\|v_0\right\|_2.
\end{aligned}$$

The second inequality used commutativity of all $\{\mathbf{B}_\ell\}_{\ell\in[L+1]}$ and the definition of $\mathbf{\Pi}_{L+2-\ell}$, the third used the guarantee (7.67), the fourth used (7.68) and that all $\{\mathbf{B}_\ell\}_{\ell\in[L+1]}$ have similarly ordered eigenvalues, and the last is straightforward. Finally, correctness of returning v_{L+1} follows from $\mathbf{D} = \mathbf{\Pi}_{L+1}$ and the bound

$$\kappa(\mathbf{\Pi}_{L+1}) \le \kappa(\mathbf{B})^{L+1} \implies 3\Delta(L+1)\left\|\mathbf{\Pi}_{L+1}\right\|_2 \left\|v_0\right\|_2 \le \epsilon \left\|\mathbf{\Pi}_{L+1} v_0\right\|_2.$$

To see the first claim, every $\mathbf{B}_\ell$ clearly has $\kappa(\mathbf{B}_\ell) \le \kappa(\mathbf{B})$, and we used the standard facts that for commuting $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d_{\succ \mathbf{0}}$, $\kappa(\mathbf{A}) = \kappa(\mathbf{A}^{-1})$ and $\kappa(\mathbf{AB}) \le \kappa(\mathbf{A})\kappa(\mathbf{B})$. □

At this point, obtaining the desired (7.65) follows from a direct application of Lemma 107 and the fact that the numerator and denominator of (7.66) commute with each other and $\mathbf{B}$.

Lemma 108. *For any vector $v \in \mathbb{R}^d$ and $\epsilon > 0$, we can compute a linear operator $\mathbf{R}_\epsilon$ such that*

$$\left\|(\mathbf{R}_\epsilon - \mathbf{B}^{-\frac{1}{2}})v\right\|_2 \leq \epsilon \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2,$$

and

$$\mathcal{T}_{\text{mv}}(\mathbf{R}_\epsilon) = O\left(\mathcal{T}_{\text{mv}} \cdot \sqrt{\kappa} \cdot \log^6\left(\frac{\kappa \cdot \kappa(\mathbf{B})}{\epsilon}\right)\right).$$

Proof. Let $\epsilon' \leftarrow \frac{\epsilon}{3}$, and let $\mathbf{D}$ and $\mathbf{N}$ be the (commuting) numerator and denominator of (7.66) for the rational function in Proposition 28 for the approximation factor ϵ'. Let $u \leftarrow \mathbf{N}v$, which we can compute explicitly within the alloted runtime. We then have from Proposition 28 that

$$\left\|\mathbf{D}^{-1}u - \mathbf{B}^{-\frac{1}{2}}v\right\|_2 \leq \epsilon' \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2.$$

Moreover by Lemma 107 we can compute a vector w within the allotted runtime such that

$$\left\|\mathbf{D}^{-1}u - w\right\|_2 \leq \epsilon' \left\|\mathbf{D}^{-1}u\right\|_2 \leq \epsilon' \left(\left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2 + \left\|\mathbf{D}^{-1}u - \mathbf{B}^{-\frac{1}{2}}v\right\|_2\right) \leq 2\epsilon' \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2.$$

The vector w follows from applying an explicit linear operator to v as desired. Finally, by combining the above two displays we have the desired approximation quality as well:

$$\left\|\mathbf{B}^{-\frac{1}{2}}v - w\right\|_2 \leq 3\epsilon' \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2 = \epsilon \left\|\mathbf{B}^{-\frac{1}{2}}v\right\|_2.$$

□

Implementing the homotopy method

In this section, we implement the homotopy method outlined in Section 7.4.2 by using the inverse square root access given by Lemma 108. We require the following helper result.

Lemma 109. *Suppose for some $\epsilon \in (0, 1)$, and $\mathbf{M}, \mathbf{N} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\kappa(\mathbf{N}) \leq \beta$, that*

$$-\epsilon\mathbf{N}^{-1} \preceq \mathbf{M}^{-1} - \mathbf{N}^{-1} \preceq \epsilon\mathbf{N}^{-1}.$$

Then,

$$-9\epsilon\beta\mathbf{N}^2 \preceq \mathbf{M}^2 - \mathbf{N}^2 \preceq 9\epsilon\beta\mathbf{N}^2.$$

Proof. The statement is scale-invariant, so suppose for simplicity that $\mathbf{I} \preceq \mathbf{N} \preceq \beta\mathbf{I}$. Also, by rearranging the assumed bound, we have

$$-3\epsilon\mathbf{N} \preceq \mathbf{M} - \mathbf{N} \preceq 3\epsilon\mathbf{N}.$$

Write $\mathbf{D} = \mathbf{M} - \mathbf{N}$ such that $\|\mathbf{D}\|_2 \le 3\epsilon$. We bound the quadratic form with an arbitrary unit vector $v \in \mathbb{R}^d$:

$$\begin{aligned}\left|v^\top(\mathbf{M}^2 - \mathbf{N}^2)v\right| &= \left|v^\top(\mathbf{N}^2 - (\mathbf{N}+\mathbf{D})^2)v\right| \\ &= \left|v^\top(\mathbf{ND} + \mathbf{DN} + \mathbf{D}^2)v\right| \\ &\le 3\|\mathbf{D}\|_2 \|\mathbf{N}\|_2 \le 9\epsilon\beta.\end{aligned}$$

This completes the proof upon using $\mathbf{N}^2 \succeq \mathbf{I}$. □

We now state our main claim regarding solving (7.7), (7.8) with general $\mathbf{B}$.

Theorem 43. *Given matrices $\{\mathbf{M}_i\}_{i\in[n]}$ with explicit factorizations (7.9), such that (7.7) is feasible for some $\kappa \ge 1$ and we can solve linear systems in linear combinations of $\{\mathbf{M}_i\}_{i\in[n]}$ to ϵ relative accuracy in the sense of (7.10) in $\mathcal{T}^{sol}_{\mathcal{M}} \cdot \log \frac{1}{\epsilon}$ time, and $\mathbf{B}$ satisfying*

$$\mathbf{B} \preceq \mathcal{M}(\mathbf{1}) \preceq \alpha\mathbf{B},\ \mathbf{I} \preceq \mathbf{B} \preceq \beta\mathbf{I}, \tag{7.11}$$

we can return weights $w \in \mathbb{R}^n_{\ge 0}$ satisfying (7.8) with probability $\ge 1-\delta$ in time

$$O\left(\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]} \cup \{\mathbf{B}\}\right) + \mathcal{T}^{sol}_{\mathcal{M}}\right) \cdot \kappa^2 \cdot \frac{\log^{11}(\frac{mnd\kappa\beta}{\delta\epsilon})}{\epsilon^8} \cdot \log\frac{\alpha}{\epsilon}\right).$$

Proof. We follow Section 7.4.2, which shows that it suffices to solve the following problem $O(\log \frac{\alpha}{\epsilon})$ times. We have an instance of (7.7), (7.8) for the original value of κ, and some matrix $\mathbf{B}$ with $\kappa(\mathbf{B}) \le \beta$ such that we know $w \in \mathbb{R}^n_{\ge 0}$ with $\mathbf{B} \preceq \mathcal{M}(w) \preceq 3\kappa\mathbf{B}$. We wish to compute w' with

$$\mathbf{B} \preceq \mathcal{M}(w') \preceq (1+\epsilon)\kappa\mathbf{B}. \tag{7.69}$$

To do so, we use Lemma 108, which yields a matrix $\mathbf{R}$ such that

$$\frac{\epsilon}{45\beta}\mathbf{B}^{-\frac{1}{2}} \preceq \mathbf{R} - \mathbf{B}^{-\frac{1}{2}} \preceq \frac{\epsilon}{45\beta}\mathbf{B}^{-\frac{1}{2}} \text{ and } \mathcal{T}_{\text{mv}}(\mathbf{R}) = O\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{V}_i\}_{i\in[n]} \cup \{\mathbf{B}\}\right) \cdot \sqrt{\kappa} \cdot \log^6\left(\frac{\kappa\beta}{\epsilon}\right)\right).$$

By Lemma 109, this implies that

$$-\frac{\epsilon}{5}\mathbf{B} \preceq \mathbf{R}^{-2} - \mathbf{B} \preceq \frac{\epsilon}{5}\mathbf{B}. \tag{7.70}$$

Now, if we solve (7.7), (7.8) with $\mathbf{R}^{-2}$ in place of $\mathbf{B}$ to accuracy $1+\frac{\epsilon}{5}$, since the optimal value of κ has changed by at most a factor of $1+\frac{\epsilon}{5}$, it suffices to compute w' such that

$$\mathbf{R}^{-2} \preceq \mathcal{M}(w') \preceq \left(1 + \frac{3\epsilon}{5}\right)\kappa\mathbf{R}^{-2},$$

since combined with (7.70) this implies (7.69) up to rescaling w'. Finally, to compute the above reweighting it suffices to apply Theorem 42 with $\mathbf{M}_i \leftarrow \mathbf{R}\mathbf{M}_i\mathbf{R}$ for all $i \in [n]$. It is clear these matrices satisfy (7.9) with the decompositions given by $\mathbf{V}_i \leftarrow \mathbf{R}\mathbf{V}_i$, and we can apply these matrices in time $\mathcal{T}_{\text{mv}}(\mathbf{R}) + \mathcal{T}_{\text{mv}}(\mathbf{V}_i)$. It is straightforward to check that throughout the proof of Theorem 42, this replaces each cost of $\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]})$ with $\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]}) + \mathcal{T}_{\text{mv}}(\mathbf{R})$, giving the desired runtime. □

7.5 Schatten packing

7.5.1 Mirror descent interpretation of [379]

We begin by reinterpreting the [379] solver, which achieves the state-of-the-art parallel runtime for packing LPs[10]. An (ℓ_∞) packing LP algorithm solves the following decision problem.[11].

Problem 2 (ℓ_∞ packing linear program)**.** *Given entrywise nonnegative $\mathbf{A} \in \mathbb{R}^{d\times n}_{\geq 0}$, either find primal solution $x \in \Delta^n$ with $\|\mathbf{A}x\|_\infty \leq 1+\epsilon$ or dual solution $y \in \Delta^d$ with $\mathbf{A}^\top y \geq (1-\epsilon)\mathbf{1}$.*

Algorithm 30: PackingLP$(\mathbf{A}, \epsilon)$

Input: $\mathbf{A} \in \mathbb{R}^{d\times n}_{\geq 0}$, $\epsilon \in [0, \frac{1}{2}]$;
$K \leftarrow \frac{3\log(d)}{\epsilon}$, $\eta \leftarrow K^{-1}$, $T \leftarrow \frac{4\log(d)\log(nd/\epsilon)}{\epsilon^2}$;
$[w_0]_i \leftarrow \frac{\epsilon}{n^2 d}$ for all $i \in [n]$, $z \leftarrow \mathbf{0}$, $t \leftarrow 0$;
while $\mathbf{A}w_t \leq K\mathbf{1}$, $\|w_t\|_1 \leq K$ **do**
  $v_t \leftarrow \frac{\exp(\mathbf{A}w_t)}{\|\exp(\mathbf{A}w_t)\|_1}$;
  $g_t \leftarrow \max(0, \mathbf{1} - \mathbf{A}^\top v_t)$ entrywise;
  $w_{t+1} \leftarrow w_t \circ (1+\eta g_t)$, $z \leftarrow z + v_t$, $t \leftarrow t+1$;
  if $t \geq T$ **then**
    Return: $y \leftarrow \frac{1}{T}z$;
Return: $x \leftarrow \frac{w_t}{\|w_t\|_1}$;

The following result is shown in [379].

Proposition 29. PackingLP *(Algorithm 30) solves Problem 2 in $O(\text{nnz}(\mathbf{A}) \cdot \frac{\log(d)\log(nd/\epsilon)}{\epsilon^2})$ time.*

Oour interpretation of the analysis of [379], combines two ingredients: a potential argument and mirror descent, which yields a dual feasible point if $\|w_t\|_1$ did not grow sufficiently.

Potential argument. The potential used by [379] is $\log(\sum_{j\in[d]} \exp([\mathbf{A}w_t]_j)) - \|w_t\|_1$, well-known to be a $O(\log d)$-additive approximation of $\|\mathbf{A}w_t\|_\infty - \|w_t\|_1$. As soon as $\|\mathbf{A}w_t\|_\infty$ or $\|w_t\|_1$ reaches

[10]The [379] solver also generalizes to covering and mixed objectives; we focus on packing in this work.
[11]Packing linear programs are sometimes expressed as the optimization problem $\max_{x\geq 0, \mathbf{A}x\leq \mathbf{1}} \|x\|_1$, similarly to (7.12); these problems are equivalent up to a standard binary search, see e.g. discussion in [285].

the scale $O(\frac{\log d}{\epsilon})$, by nonnegativity this becomes a multiplicative guarantee, motivating the setting of threshold K. To prove the potential is monotone, [379] uses step size K^{-1} and a Taylor approximation; combining with the termination condition yields the desired claim.

Mirror descent. To certify that w_t grows sufficiently (e.g. the method terminates in few iterations, else dual feasibility holds), we interpret the step $w_{t+1} \leftarrow w_t \circ (1+\eta g_t)$ as approximate entropic mirror descent. Specifically, we track the quantity $\sum_{0 \le t < T} \langle \eta g_t, u \rangle$, and show that if $\|w_t\|_1$ has not grown sufficiently, then it must be bounded for every $u \in \Delta^n$, certifying dual feasibility. Formally, for any g_t sequence and $u \in \Delta^n$, we show

$$O(\log(nd/\epsilon)) + \log\left(\frac{\|w_T\|_1}{\|w_0\|_1}\right) \ge \sum_{0 \le t < T} \langle \eta g_t, u \rangle \ge \eta \sum_{0 \le t < T} \left\langle \mathbf{1} - \mathbf{A}^\top v_t, u \right\rangle.$$

The last inequality followed by g_t being an upwards truncation. If $\|w_T\|_1$ is bounded (else, we have primal feasibility), we show the entire above expression is bounded $O(\log \frac{nd}{\epsilon})$ for any u. Thus, by setting $T = O(\frac{\log(nd/\epsilon)}{\eta\epsilon})$ and choosing u to be each coordinate indicator, it follows that the average of all v_t is coordinatewise at least $1 - \epsilon$, and solves Problem 2 as a dual solution.

Our g_t is chosen as the (truncated) gradient of the function used in the potential analysis, so its form allows us to interpret dual feasibility (e.g. v_t has ℓ_1 norm 1 and is a valid dual point). Our analysis patterns standard mirror descent, complemented by side information which says that lack of a primal solution can transform a regret guarantee into a feasibility bound. We apply this framework to analyze ℓ_p variants of Problem 2, via different potentials; nonetheless, our proofs are quite straightforward upon adopting this perspective, and we believe it may yield new insights for instances with positivity structure.

7.5.2 ℓ_p-norm packing linear programs

In this section, we give a complete self-contained example of the framework proposed in Section 7.5.1, for approximately solving ℓ_p norm packing linear programs. Specifically, we now consider the generalization of Problem 2 to ℓ_p norms; throughout, $q = \frac{p}{p-1}$ is the dual norm.

Problem 3 (ℓ_p packing linear program)**.** *Given entrywise nonnegative $\mathbf{A} \in \mathbb{R}^{d \times n}_{\ge 0}$, either find primal solution $x \in \Delta^n$ with $\|\mathbf{A}x\|_p \le 1 + \epsilon$ or dual solution $y \in \mathbb{R}^d_{\ge 0}, \|y\|_q = 1$ with $\mathbf{A}^\top y \ge (1-\epsilon)\mathbf{1}$.*

For $p = \frac{\log d}{\epsilon}$, Problem 3 recovers Problem 2 up to constants as ℓ_p multiplicatively approximates ℓ_∞ by $1 + \epsilon$. We now state our method for solving Problem 3 as Algorithm 31.

Other than changing parameters, the only difference from Algorithm 30 is that v is a point with unit ℓ_q norm induced by the gradient of our potential Φ_t. We state our main potential fact, whose proof is based straightforwardly on Taylor expanding $\|\cdot\|_p$, and deferred to Appendix F.3 for brevity.

Lemma 110. *In all iterations t of Algorithm 31, defining $\Phi_t := \|\mathbf{A}w_t\|_p - \|w_t\|_1$, $\Phi_{t+1} \le \Phi_t$.*

Algorithm 31: PNormPacking($\mathbf{A}, \epsilon, p$)

1 **Input:** $\mathbf{A} \in \mathbb{R}^{d \times n}_{\geq 0}, \epsilon \in [0, \frac{1}{2}], p \geq 2$;
2 $\eta \leftarrow p^{-1}, T \leftarrow \frac{4p \log(\frac{nd}{\epsilon})}{\epsilon}$;
3 $[w_0]_i \leftarrow \frac{\epsilon}{n^2 d}$ for all $i \in [n]$, $z \leftarrow \mathbf{0}$, $t \leftarrow 0$;
4 **while** $\|w_t\|_1 \leq \epsilon^{-1}$ **do**
5 $\quad g_t \leftarrow \max(0, \mathbf{1} - \mathbf{A}^\top (v_t)^{p-1})$ entrywise, for $v_t \leftarrow \frac{\mathbf{A} w_t}{\|\mathbf{A} w_t\|_p}$;
6 $\quad w_{t+1} \leftarrow w_t \circ (1 + \eta g_t)$, $z \leftarrow z + (v_t)^{p-1}$, $t \leftarrow t + 1$;
7 $\quad$ **if** $t \geq T$ **then**
8 $\quad\quad$ **Return:** $y = \frac{z}{\|z\|_q}$;
9 **Return:** $x = \frac{w_t}{\|w_t\|_1}$;

We now prove our main result, which leverages the potential bound following the framework of Section 7.5.1. In the proof, we assume that entries of $\mathbf{A}$ are bounded by $n\epsilon^{-1}$; this does not incur more loss than a constant multiple of ϵ in the guarantees, and a proof can be found as Lemma 291.

Theorem 47. *Algorithm 31 runs in time $O(\text{nnz}(\mathbf{A}) \cdot \frac{p \log(nd/\epsilon)}{\epsilon})$. Further, its output solves Problem 3.*

Proof. The runtime follows from Line 7 (each iteration cost is dominated by multiplication through $\mathbf{A}$), so we prove correctness. Define potential Φ_t as in Lemma 110, and note that as $w_0 = \frac{\epsilon}{n^2 d}\mathbf{1}$,

$$\Phi_0 \leq \|\mathbf{A} w_0\|_p \leq \frac{1}{n} \|\mathbf{1}\|_p \leq 1.$$

The second inequality followed from our assumption on $\mathbf{A}$ entry sizes (Lemma 291). If Algorithm 31 breaks out of the while loop of Line 4, we have by Lemma 110 that for x returned on Line 11,

$$\|\mathbf{A} w_t\|_p - \|w_t\|_1 \leq 1 \implies \|\mathbf{A} x\|_p \leq \frac{1 + \|w_t\|_1}{\|w_t\|_1} \leq 1 + \epsilon.$$

Thus, primal feasibility is always correct. We now prove correctness of dual feasibility. First, let $V_x(u) = \sum_{i \in [n]} u_i \log(\frac{u_i}{x_i})$ be the Kullback-Leibler divergence from x to u, for $x, u \in \Delta^d$. Define the normalized points $x_t = \frac{w_t}{\|w_t\|_1}$ in each iteration. Expanding definitions,

$$\begin{aligned} V_{x_{t+1}}(u) - V_{x_t}(u) &= \sum_{i \in [n]} u_i \log \frac{[x_t]_i}{[x_{t+1}]_i} \\ &= \sum_{i \in [n]} u_i \left(\log \left(\frac{\|w_{t+1}\|_1}{\|w_t\|_1} \right) + \log \left(\frac{1}{1 + \eta [g_t]_i} \right) \right) \\ &\leq -\eta(1 - \eta) \langle g_t, u \rangle + \log \left(\frac{\|w_{t+1}\|_1}{\|w_t\|_1} \right). \end{aligned} \tag{7.71}$$

The only inequality used the bounds, for $g, \eta \in [0, 1]$,

$$\log\left(\frac{1}{1+\eta g}\right) \leq g \log\left(\frac{1}{1+\eta}\right) \leq -\eta(1-\eta)g.$$

Telescoping (7.71) over all T iterations, and using $V_{x_0}(u) \leq \log n$ for all $u \in \Delta^n$ since x_0 is uniform, we have that whenever Line 4 is not satisfied before the check on Line 7 (i.e. $t \geq T$),

$$\eta(1-\eta) \sum_{0 \leq t < T} \langle g_t, u \rangle \leq \log\left(\frac{\|w_T\|_1}{\|w_0\|_1}\right) + V_{x_0}(u) \leq \log\left(\frac{nd}{\epsilon^2}\right) + \log n \leq 2\log\left(\frac{nd}{\epsilon}\right). \tag{7.72}$$

The last inequality used $\|w_T\|_1 \leq \epsilon^{-1}$ by assumption, and $\|w_0\|_1 = \frac{\epsilon}{nd}$. Next, since each $g_t \geq \mathbf{1} - \mathbf{A}^\top (v_t)^{p-1}$ entrywise, defining $\bar{z} = \frac{z}{T}$,

$$\sum_{0 \leq t < T} \langle g_t, u \rangle \geq \sum_{0 \leq t < T} \left\langle \mathbf{1} - \mathbf{A}^\top (v_t)^{p-1}, u \right\rangle = T \left\langle \mathbf{1} - \mathbf{A}^\top \bar{z}, u \right\rangle, \text{ for all } u \in \Delta^n. \tag{7.73}$$

Combining (7.72) and (7.73), and rearranging, yields by definition of T,

$$\left\langle \mathbf{1} - \mathbf{A}^\top \bar{z}, u \right\rangle \leq \frac{2\log(\frac{nd}{\epsilon})}{T\eta(1-\eta)} \leq \frac{4p\log(\frac{nd}{\epsilon})}{T} \leq \epsilon \implies \mathbf{A}^\top \bar{z} \geq 1 - \epsilon \text{ entrywise.}$$

The last claim follows by setting u to be each coordinate-sparse simplex vector. Finally, since $\frac{\bar{z}}{\|\bar{z}\|_q} = \frac{z}{\|z\|_q}$, to show that y is a correct dual solution to Problem 3 it suffices to show $\|\bar{z}\|_q \leq 1$. This follows as $\bar{z}$ is an average of the $(v_t)^{p-1}$, convexity of ℓ_q norms, and that for all t,

$$\left\|(v_t)^{p-1}\right\|_q^q = \sum_{j \in [d]} v_t^p = \sum_{j \in [d]} \frac{[\mathbf{A}w_t]_j^p}{\|\mathbf{A}w_t\|_p^p} = 1.$$

□

7.5.3 Schatten-norm packing semidefinite programs

We generalize Algorithm 31 to solve Schatten packing semidefinite programs, which we now define.

Problem 4. *Given $\{\mathbf{A}_i\}_{i \in [n]} \in \mathbb{S}^d_{\geq 0}$, either find primal solution $x \in \Delta^n$ with $\left\|\sum_{i \in [n]} x_i \mathbf{A}_i\right\|_p \leq 1+\epsilon$ or dual solution $\mathbf{Y} \in \mathbb{S}^d_{\geq 0}$, $\|\mathbf{Y}\|_q = 1$ with $\langle \mathbf{A}_i, \mathbf{Y} \rangle \geq 1 - \epsilon$ for all $i \in [n]$.*

We assume that p is an odd integer for simplicity (sufficient for our applications), and leave for interesting future work the cases when p is even or noninteger. The potential used in the analysis and an overall guarantee are stated here, and deferred to Appendix F.3. The proofs are simple modifications of Lemma 110 and Theorem 47 using trace inequalities (similar to those in [285]) in place of scalar inequalities, as well as efficient approximation of quantities in Line 5 via the standard

Algorithm 32: SchattenPacking$(\{\mathbf{A}_i\}_{i\in[n]}, \epsilon, p)$

Input: $\{\mathbf{A}_i\}_{i\in[n]} \in \mathbb{S}^d_{\succeq 0}, \epsilon \in [0, \frac{1}{2}], p \geq 2$;
$\eta \leftarrow p^{-1}, T \leftarrow \frac{4p\log(\frac{nd}{\epsilon})}{\epsilon}$;
$[w_0]_i \leftarrow \frac{\epsilon}{n^2 d}$ for all $i \in [n]$, $z \leftarrow 0$;
while $\|w_t\|_1 \leq \epsilon^{-1}$ **do**
  $g_t \leftarrow \max\left(0, \mathbf{1} - \left\langle \mathbf{A}_i, \mathbf{V}_t^{p-1} \right\rangle\right)$ entrywise, for $\mathbf{V}_t \leftarrow \frac{\sum_{i\in[n]}[w_t]_i \mathbf{A}_i}{\left\|\sum_{i\in[n]}[w_t]_i \mathbf{A}_i\right\|_p}$;
  $w_{t+1} \leftarrow w_t \circ (1 + \eta g_t)$, $\mathbf{Z} \leftarrow \mathbf{Z} + (\mathbf{V}_t)^{p-1}$, $t \leftarrow t + 1$;
  if $t \geq T$ **then**
    Return: $\mathbf{Y} = \frac{\mathbf{Z}}{\|\mathbf{Z}\|_q}$;
Return: $x = \frac{w_t}{\|w_t\|_1}$;

technique of Johnson-Lindestrauss projections.

Lemma 111. *In all iterations t of Algorithm 32, defining $\Phi_t := \left\|\sum_{i\in[n]}[w_t]_i \mathbf{A}_i\right\|_p - \|w_t\|_1$, $\Phi_{t+1} \leq \Phi_t$.*

Theorem 48. *Let p be odd. Algorithm 32 runs in $O(\frac{p\log(nd/\epsilon)}{\epsilon})$ iterations, and its output solves Problem 4. Each iteration is implementable in $O(\text{nnz} \cdot \frac{p\log(nd/\epsilon)}{\epsilon^2})$, where* nnz *is the number of nonzero entries amongst all $\{\mathbf{A}_i\}_{i\in[n]}$, losing $O(\epsilon)$ in the quality of Problem 4 with probability $1 - \text{poly}((nd/\epsilon)^{-1})$.*

7.5.4 Schatten packing with a ℓ_∞ constraint

We remark that the framework outlined in Section 7.5.1 is flexible enough to handle mixed-norm packing problems. Specifically, we give the following guarantee which will used in an application.

Proposition 30. *Following Theorem 48's notation, let p be odd, $\{\mathbf{A}_i\}_{i\in[n]} \in \mathbb{S}^d_{\succeq 0}$, $0 < \epsilon = O(\alpha)$, and*

$$\min_{\substack{x\in\Delta^n \\ \|x\|_\infty \leq \frac{1+\alpha}{n}}} \|\mathcal{A}(x)\|_p = \text{OPT}. \tag{7.74}$$

for $\mathcal{A}(x) := \sum_{i\in[n]} x_i \mathbf{A}_i$. Given estimate of OPT *exponentially bounded in $\frac{nd}{\epsilon}$, there is a procedure calling Algorithm 76 $O(\log \frac{nd}{\epsilon})$ times giving $x \in \Delta^n$ with $\|x\|_\infty \leq \frac{(1+\alpha)(1+\epsilon)}{n}$, $\|\mathcal{A}(x)\|_p \leq (1+\epsilon)\text{OPT}$. Algorithm 76 runs in $O(\frac{\log(nd/\epsilon)\log n}{\epsilon^2})$ iterations, each implementable in time $O(\text{nnz} \cdot \frac{p\log(nd/\epsilon)}{\epsilon^2})$.*

Our method, found in Appendix F.3, approximately solves (7.74) by first applying a standard binary search to place $\mathcal{A}(x)$ on the right scale, for which it suffices to solve an approximate decision problem. Then, we apply a truncated mirror descent procedure on the potential $\Phi(w) = \log(\exp(\|\mathcal{A}(w)\|_p) + \exp(\frac{n}{1+\alpha}\|w\|_\infty)) - \|w\|_1$, and prove correctness for solving the decision problem following the framework we outlined in Section 7.5.1.

Chapter 8

High-Dimensional Robust Statistics in Almost-Linear Time

This chapter is based on [289, 180, 288, 181] with Ilias Diakonikolas, Arun Jambulapati, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, and Tselil Schramm.

8.1 Organization

In this chapter, we present several related results on efficiently performing statistical estimation or learning tasks in high dimensions, under the presence of strong contamination (an adversary who is allowed to replace a bounded fraction of the dataset with arbitrary points from the support of the distribution one wishes to learn about). Performing these tasks in high dimensions with an explicit (polynomial-time) algorithm is a challenging endeavor, and it was not until recently that the polynomial-time tractability of the tasks considered in this chapter was settled [175, 337, 176]. The main contributions of this chapter are to go even further, demonstrating that several basic variants of these tasks are in fact tractable in *almost-linear time*. To attain these results, we will build heavily upon SDP technology introduced in Chapter 7.

Because many of the results in this chapter directly build upon similar techniques (utilizing variants of matrix multiplicative weights and other tools inspired by SDP solvers for certifying and filtering corrupted datasets), we present them in one cohesive section. We now give an overview of the organization of this chapter, as well as the problems it studies.

1. **Clustering mixture models.** We consider the list-decodable mean estimation problem [118], and its application to robust clustering tasks, in Sections 8.2, 8.3, 8.4, and 8.5.

2. **Parameter estimation in generalized linear models.** We consider learning a latent

parameter in GLMs from corrupted observations in Sections 8.6, 8.7, 8.8, 8.9, and 8.10.

3. **Principal component analysis.** We consider the problem of robustly recovering an approximate top eigenvector of a sub-Gaussian distribution in Section 8.11.

8.2 Clustering mixture models in almost-linear time

We develop novel algorithms achieving almost-optimal runtimes for two closely related fundamental problems in high-dimensional statistical estimation: clustering well-separated mixture models and mean estimation in the list-decodable learning ("majority-outlier") regime. Before we formally state our contributions, we provide the necessary background and motivation for this work.

Clustering well-separated mixture models. Mixture models are a well-studied class of generative models used widely in practice. Given a family of distributions $\mathcal{F}$, a mixture model $\mathcal{M}$ with k components is specified by k distributions $\mathrm{dist}_1, \ldots, \mathrm{dist}_k \in \mathcal{F}$ and nonnegative mixing weights $\alpha_1, \ldots, \alpha_k$ summing to one, and its law is given by $\sum_{i \in [k]} \alpha_i \, \mathrm{dist}_i$. That is, to draw a sample from $\mathcal{M}$, we first choose $i \in [k]$ with probability α_i, and then draw a sample from dist_i. When the weights are all equal to $\frac{1}{k}$, we call the mixture *uniform*. Mixture models, especially Gaussian mixture models, have been widely studied in statistics since pioneering work of Pearson in 1894 [440], and more recently, in theoretical computer science [159, 35, 520, 7, 307, 97, 45, 454].

A canonical learning task for mixture models is the *clustering problem*. Namely, given independent samples drawn from $\mathcal{M}$, the goal is to approximately recover which samples came from which component. To ensure that this inference task is information-theoretically possible, a common assumption is that $\mathcal{M}$ is "well-separated" and "well-behaved": for example, we may assume each component dist_i is sufficiently concentrated (with sub-Gaussian tails or bounded moments), and that component means have pairwise distance at least Δ, for sufficiently large Δ. The goal is then to efficiently and accurately cluster samples from $\mathcal{M}$ with as small a separation as possible.

The prototypical example is the case of uniform mixtures of bounded-covariance Gaussians, i.e. mixtures of the form $\mathcal{M} = \sum_{i \in [k]} \frac{1}{k} \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$, where each $\mathbf{\Sigma}_i$ is unknown and satisfies $\|\mathbf{\Sigma}_i\|_{\mathrm{op}} \le \sigma^2$. Prior to the current work, the fastest known algorithm for this learning problem was due to [7], building on [520]. Notably, [7] gave a polynomial-time clustering algorithm when $\Delta = \Omega(\sigma\sqrt{k})$. Interestingly, the algorithmic approach of [520, 7] is surprisingly simple and elegant: first, run k-PCA on the set of n samples in $\mathbb{R}^d$ to find a k-dimensional subspace (which can be shown to approximately capture the span of the component means), and then perform a distance-based clustering algorithm in this subspace. The runtime of this algorithm is dominated by $\widetilde{\Omega}(ndk)$ – the cost of (approximate) k-PCA.[1] The idea of using k-PCA as a subroutine to solve the clustering problem is very natural and has also been useful in practice. Indeed, using PCA as a preprocessing step before applying

[1] Throughout, when convenient, we hide polylogarithmic factors in the sample size and algorithm failure probabilities with the $\widetilde{O}$ notation. We reserve the terminology "almost-linear" to mean linear up to subpolynomial factors, and the terminology "nearly-linear" to mean linear up to polylogarithmic factors.

further learning algorithms (such as clustering) is so ubiquitous that it is commonly suggested by introductory textbooks on machine learning, see e.g. [403].

However, in our setting, since the size of the description this problem is $O(nd)$, the runtime of k-PCA is off from linear time by a factor of roughly k. In many real-world settings, this factor of k is quite significant. For instance, modern image datasets such as ImageNet [170] often have hundreds or thousands of different classes and subclasses [470]. As a result, many clustering tasks on these datasets often have k of the same order. The resulting overhead would cause many tasks to be infeasible at scale on these datasets. Yet, despite considerable attention over the last two decades,[2] no faster algorithm has been developed for the clustering task. In particular, the runtime of k-PCA has remained a bottleneck in this setting.

The preceding discussion motivates the following natural question.

Question 1. *Can we cluster mixtures of k "well-separated" structured distributions without the use of k-PCA? More ambitiously, is there a clustering algorithm that runs in (almost)-*linear time*?*

Prior to the current work, this question remained open even for uniform k-mixtures of identity covariance Gaussians with pairwise mean separation as large as $\mathrm{poly}(k)$. In addition to its fundamental interest, a runtime improvement of this sort may have significant practical implications for clustering at scale in real-world applications, see e.g. [438, 538], where spectral methods are commonly used. As our main contribution, *we resolve Question 1 for the general class of mixtures of bounded-covariance distributions under information-theoretically near-optimal separation.*

List-decodable mean estimation. In many statistical settings, including machine learning security [62, 81, 494, 176] and exploratory data analysis e.g. in biology [461, 361, 436], datasets contain arbitrary — and even adversarially chosen — outliers. The central question of the field of robust statistics is to design estimators tolerant to a small amount of unconstrained contamination. Classical work in this field [31, 512, 278, 513, 263, 279] developed robust estimators for many basic tasks, although with computational costs scaling exponentially in the problem dimension. More recently, a line of work in computer science, starting with [175, 337], developed the first computationally-efficient learning algorithms (attaining near-optimal error) for various estimation problems. Subsequently, there has been significant progress in algorithmic robust statistics in a variety of settings (see [182] for a survey).

In many of these works, it is typically assumed that the fraction of corrupted points is less than $\frac{1}{2}$. Indeed, when more than half the points are corrupted, the problem is ill-posed: there is not necessarily a uniquely-defined notion of "uncorrupted samples." While outputting a *single* accurate hypothesis in this regime is information-theoretically impossible, one may be able to compute a *small list* of hypotheses with the guarantee that *at least one of them* is accurate. This relaxed notion of estimation is known as *list-decodable learning* [56, 118].

[2] We note that a recent line of work has developed sophisticated polynomial-time clustering algorithms under smaller separation assumptions, see e.g. [186, 275, 330]. These algorithms leverage higher moments of the distribution and consequently require significantly higher sample and computational complexity.

Definition 27 (List-decodable learning)**.** *Given a parameter $0 < \alpha < \frac{1}{2}$ and a distribution family $\mathcal{F}$ on $\mathbb{R}^d$, a* list-decodable learning algorithm *takes as input α and a multiset T of n points such that an unknown α fraction of T are independent samples from an unknown distribution* dist $\in \mathcal{F}$*, and no assumptions are made on the remaining samples. Given T and α, the goal is to output a "small" list of hypotheses at least one of which is close to the target parameter of* dist*.*

Arguably the most fundamental problem in the list-decodable learning setting is mean estimation, wherein the goal is to output a small list of hypotheses, one of which is close to the true mean. A natural problem in its own right, list-decodable mean estimation generalizes the problem of learning well-separated mixture models (as explained below) and can model important applications such as crowdsourcing [495, 390] or semi-random community detection in stochastic block models [118]. Moreover, it is particularly useful in the context of semi-verified learning [118, 390], where a learner can audit a small amount of trusted data. *An important remark is that the parameter $\alpha \in (0, \frac{1}{2})$ can be quite small in some of these applications and should not necessarily be thought of as a constant.* In addition to applications in clustering mixture models, a concrete example is the crowdsourcing setting with many unreliable responders studied in [390], where the parameter α is tiny, depending inversely-polynomially on other problem parameters such as the dimension.

The parameter α in the list-decodable mean estimation setting plays a very similar role to the parameter $\frac{1}{k}$ in learning (uniform) mixture models. This is no coincidence: list-decodable mean estimation can be thought of as a natural robust generalization of clustering well-separated mixtures. Indeed, if we run a list-decodable mean estimation algorithm on a dataset drawn from a uniform mixture of k sufficiently nice and well-separated distributions with α set to $\frac{1}{k}$, the output list *must* contain a candidate mean which is close to the mean of each component. This is because from the perspective of the list-learning algorithm, each component could be the "true" unknown distribution dist, and thus the list must contain a hypothesis close to the mean of this "true" distribution. This small list of hypotheses can then typically be used to cluster the original dataset. One conceptually important implication of this observation is that list-decodable mean estimation algorithms also naturally lead to algorithms for clustering well-separated mixture models (even in the presence of a small fraction of corrupted samples) — a reduction we formalize in this work.

The first polynomial-time algorithm for list-decodable mean estimation, when $\mathcal{F}$ is the family of bounded-covariance distributions, was by [118]. The [118] algorithm was based on black-box calls to semidefinite program solvers and had a large polynomial runtime. Since then, a sequence of works [179, 136] (as well as developments contained in this thesis) have obtained substantially improved runtimes for this problem, while retaining the (near-optimal) statistical guarantees of [118]. In particular, in Sections G.2, G.3, and G.4 we develop an algorithm which runs in time $\widetilde{O}(\frac{nd}{\alpha})$ and achieves near-optimal error (within a polylogarithmic factor).

Interestingly, as in the case of clustering mixture models, the $\widetilde{\Omega}(\frac{nd}{\alpha})$ runtime dependence of the algorithm of Sections G.2, G.3, and G.4 is *also* due to running a k-PCA subroutine — for

$k = \Omega(\frac{1}{\alpha})$ — to reduce the problem to a k-dimensional subspace. In more detail, the algorithm of Sections G.2, G.3, and G.4 can be viewed as a reduction from list-decodable mean estimation to polylogarithmically many calls to k-PCA (for carefully chosen matrices). Thus, the cost of k-PCA appears as a runtime barrier in state-of-the-art algorithms for list-decodable mean estimation as well. In regimes where α is small, the $\widetilde{\Omega}(\frac{nd}{\alpha})$ runtime is significantly sub-optimal in the input size. This leaves open whether the extraneous linear dependence on α^{-1} is improvable, and brings us to our second main question.

Question 2. *Can we perform list-decodable mean estimation with near-optimal statistical guarantees in (almost)-linear time?*

In this chapter, *we similarly resolve Question 2 for the class of bounded-covariance distributions.*

8.2.1 Our results

We answer both Question 1 and Question 2 in the affirmative, up to subpolynomial factors. Perhaps surprisingly, to resolve the longstanding open problem of clustering mixture models in almost-linear time, we develop an almost-linear time algorithm for the (much more general) problem of list-decodable mean estimation. To then solve the clustering problem, we develop a fast post-processing technique that efficiently reduces the clustering task to list-decodable mean estimation. In light of this development, we begin by presenting our list-decodable estimation result.

Theorem 49 (informal, see Theorem 51)**.** *For any fixed constant $\epsilon_0 > 0$, there is an algorithm* FastMultifilter *with the following guarantee. Let* dist *be a distribution over $\mathbb{R}^d$ with unknown mean μ^* and unknown covariance $\mathbf{\Sigma}$ with $\|\mathbf{\Sigma}\|_{\mathrm{op}} \le \sigma^2$, and let $\alpha \in (0, 1)$. Given α and a multiset of $n = \Omega(\frac{d}{\alpha})$ points on $\mathbb{R}^d$ such that an α-fraction are i.i.d. draws from* dist*,* FastMultifilter *runs in time $O(n^{1+\epsilon_0} d)$ and outputs a list L of $O(\alpha^{-1})$ hypotheses so that with high probability we have*

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 = O\left(\frac{\sigma \log \alpha^{-1}}{\sqrt{\alpha}}\right).$$

Notably, in the setting of Theorem 49, a sample complexity of $\Omega(\frac{d}{\alpha})$, error of $\Omega(\sigma\alpha^{-\frac{1}{2}})$, and list size $\Omega(\alpha^{-1})$ are all information-theoretically necessary [186]. Hence, up to a $\log(\alpha^{-1})$ factor in the error, Theorem 49 achieves optimal statistical guarantees for this problem in almost-linear time.

Leveraging Theorem 49, and combining it with a new almost-linear time post-processing procedure of the resulting list, we achieve our almost-linear runtime for clustering well-separated mixtures under only a second moment bound assumption — even in the presence of a small fraction of outliers. In more detail, our algorithm can tolerate a fraction of outliers proportional to the relative size of the smallest true cluster. For brevity, in this introduction, we will state the natural special case of our clustering result for uniform bounded-covariance mixtures without outliers. We also achieve

similar (indeed, slightly stronger) guarantees when the mixture components are sub-Gaussian or have bounded fourth moments.

Theorem 50 (informal, see Corollaries 33, 35, 36)**.** *For any fixed constant $\epsilon_0 > 0$, there is an algorithm with the following guarantee. Given a multiset of $n = \Omega(dk)$ i.i.d. samples from a uniform mixture model $\mathcal{M} = \sum_{i\in[k]} \frac{1}{k} \mathrm{dist}_i$, where each component dist_i has unknown mean μ_i, unknown covariance matrix $\mathbf{\Sigma}_i$ with $\|\mathbf{\Sigma}_i\|_{\mathrm{op}} \le \sigma^2$, and $\min_{i,i'\in[k],i\neq i'} \|\mu_i - \mu_{i'}\|_2 = \widetilde{\Omega}(\sqrt{k})\,\sigma$, the algorithm runs in time $O(n^{1+\epsilon_0} \max(k,d))$, and with high probability correctly clusters 99% of the points.*

Some remarks are in order. First, we note that pairwise mean separation of $\Omega(\sqrt{k}\,\sigma)$ is information-theoretically necessary for accurate clustering to be possible for bounded covariance components. The algorithm establishing Theorem 50 nearly achieves the optimal separation. Secondly, and crucially, our clustering algorithm runs in almost-linear time. Finally, as previously alluded to, our clustering method is robust to outliers, and can handle mixtures with arbitrary weights, with guarantees depending on the smallest weight (see Corollary 36 for a precise statement).

It is worth commenting on the $\max(k,d)$ term appearing in the running time of Theorem 50. Our algorithm runs in almost-linear time as long as $k \le d$. For the extreme regime where $k \gg d$, our algorithm has running time $O(n^{1+\epsilon_0}k)$. In this parameter regime, it is plausible that $\Omega(nk)$ is a runtime bottleneck for the following reason: even if we are given (exactly) the centers μ_i, $i \in [k]$ for free, $\Omega(nk)$ time seems to be required to simply assign each of the n points to its closest center.

Remark 8 (Prior work)**.** The prior works [7, 45] obtained polynomial-time clustering algorithms with similar statistical guarantees as Theorem 50, under the (much stronger) assumption that each component distribution dist_i has sub-Gaussian tails. For bounded covariance distributions, these algorithms require the stronger mean separation of $\Omega(k\sigma)$ [44]. On the other hand, the clustering methods obtained in [118] (as an application of their list-decodable mean estimator) (i) require sub-Gaussian components, and (ii) partition the dataset into $C \cdot k$ for some constant $C > 2$ — as opposed to k — clusters. In summary, prior work has not explicitly obtained *even a polynomial-time* clustering algorithm in the bounded covariance setting with separation $o(k)\sigma$.

8.2.2 Technical overview

Here, we describe the techniques developed for our problem at a high level, and how they circumvent several conceptual runtime barriers encountered by prior approaches to list-decodable mean estimation and clustering mixture models. Our full proofs are quite technically challenging, and involve several additional steps which we omit here for clarity of exposition. Throughout this section, we assume that the "scale" of the problem is $\sigma = 1$ for simplicity (e.g. distribution covariances are bounded by $\mathbf{I}$).

Prior approaches and their limitations

In this section, we briefly describe two recent fast algorithms for list-decodable mean estimation, developed by [179] and Sections G.2, G.3, and G.4,[3] focusing on tools used in their analyses and bottlenecks in extending their techniques to obtain (almost)-linear runtimes. For brevity, we will cite the algorithm of Sections G.2, G.3, and G.4 as [180] in this discussion, where it was first presented.

Multifiltering. Filtering is one of the most popular techniques for robust estimation [175, 177, 360, 492, 182]. In the minority-outlier setting, filtering is based on the idea of designing certificates of corruption, which either ensure that a current estimate suffices, or can be used to identify a set of points to filter on containing more outliers (corrupted points) than inliers (clean points). Iterating this process terminates in polynomial time, because (roughly speaking) it eventually removes all outliers.

In the context of list-decodable mean estimation, standard filtering guarantees are insufficient, because we cannot afford to remove as many inliers as outliers. To overcome this difficulty, [186] introduced the "multifilter" in the context of Gaussian mean estimation, which was extended to bounded covariance distributions in [179]. At a high level, a multifilter iterates through a tree of candidate subsets, and looks for ways to either "cluster" a subset or "split" it into multiple (overlapping) subsets.[4] To ensure an efficient runtime, a multifilter maintains a potential guaranteeing that the tree size does not blow up (i.e. there are never too many candidate subsets), and carefully chooses to split or cluster based on subset sample statistics, thus ensuring that some tree node always retains a large fraction of inliers. Previous multifilters chose to split or cluster subsets based on *one-dimensional* projections along top eigenvectors of sample covariances, which can be dominated by a single outlier. In the worst case, this leads to an iteration count scaling polynomially with the dimension.

Filtering via matrix multiplicative weights. The approach taken by the fastest algorithms for mean estimation in both majority-inlier [194] and majority-outlier [180] settings is heavily motivated by filtering. In the majority-inlier case, every iteration of the filter is nearly-linear time, so the only bottleneck to an overall fast runtime is the number of iterations. However, simple hard instances show that only projecting onto the worst directions of empirical covariances may lead to an $\Omega(d)$ runtime overhead. The main idea of [194] was to choose scores capturing *multiple* bad directions at a time, preventing this worst-case behavior. These scores were based on quadratic forms with certain trace-one matrices derived from the *matrix multiplicative weights* (MMW) regret minimization framework from semidefinite programming [534, 34]. By using MMW regret bounds, [194] designs a filter that efficiently decreases the empirical covariance operator norm, which is used as a potential to yield convergence in *polylogarithmically* many iterations.

In the majority-outlier setting, the story is somewhat murkier. To overcome complications of

[3] We focus on [180] instead of [136] in this technical exposition, as they both apply Ky Fan semidefinite programming machinery to obtain fast runtimes, but the [180] approach is more relevant to us.

[4] In [179], these subsets were replaced by weight functions, but the intuition is very similar in both cases.

prior list-decodable mean estimation algorithms (e.g. the multifilter), which interleaved "filtering" and "clustering" steps, [180] designed a "k-dimensional filter", for $k = \Theta(\frac{1}{\alpha})$, that they called SIFT, decoupling the two goals. Specifically, SIFT uses scores based on k-dimensional projections to hone in on a subspace outside of which the empirical mean is accurate. It then efficiently clusters in just this subspace; combined with appropriate Ky Fan norm generalizations of MMW, the number of iterations is then improved to polylogarithmic. However, this approach of decoupling filtering and clustering appears to inherently use k-dimensional PCA as a subroutine, for $k = \Theta(\frac{1}{\alpha})$, even just to learn an "important" subspace a single time. Hence, this approach encounters a similar runtime bottleneck as prior algorithms for clustering mixture models [520, 7].

Challenges in combining techniques. As mentioned, the approach of [180] seems to inherently run into a runtime barrier at $\Omega(\frac{nd}{\alpha})$ due to its reliance on k-PCA. This suggests that to overcome this barrier, we need to develop a new algorithm which both (1) does not disentangle filtering and clustering steps, and (2) relies on univariate projections. It is natural to then try to merge the multifilter with a MMW-based potential to ensure rapid convergence.

Unfortunately, there are several obstacles towards combining these frameworks. A primary complication is that the regret minimization approach of [194] requires multiple consecutive rounds before it can ensure an appropriate potential decreases. This is because of its reliance on MMW, a "mirror descent" algorithm which typically does not provide monotone guarantees on iterates (and hence requires multiple iterations to bound regret) [161]. It is unclear how to make these arguments work within the multifilter framework, which interleaves two types of steps (splitting and clustering) that may have incompatible guarantees across iterations.

Finally, even if it were possible to combine the multifilter with a MMW-based potential analysis, there are still various difficulties towards obtaining an almost-linear runtime coming from the size of our hypothesis tree. For example, making the decision to split or cluster at a node typically requires $\Omega(nd)$ time (e.g. to compute scores), which we cannot afford to perform more than subpolynomially often. This is problematic because our multifilter tree certainly contains $\Omega(\frac{1}{\alpha})$ nodes: in the uniform mixture model case, our tree must contain hypotheses corresponding to every true cluster.

Our techniques

One-shot potential framework. In order to deal with the first of the two obstacles discussed (the non-monotonicity of MMW regret guarantees), our starting point is a framework for fast robust mean estimation (cf. Section 8.3.3), essentially matching the guarantees of [194] with a more transparent analysis. Crucially, our new framework comes with a "one-shot" potential function that shows monotone progress *at every iteration*, making it more amenable to combination with a multifilter (which needs to argue how potentials evolve between different types of steps).

In more detail, our new fast algorithm in the majority-inlier setting guarantees monotone progress on the "Schatten-norm" potential $\mathrm{Tr}(\mathbf{Y}_t^2)$, where $\mathbf{Y}_t := \mathbf{M}_t^{\log(d)}$ and $\mathbf{M}_t = \sum_{i \in T} [w_t]_i (X_i - \mu_t)(X_i -$

$\mu_t)^\top$ is the weighted empirical covariance with respect to the current weight vector w_t. We then use $\mathbf{Y}_t$ to sample carefully chosen Gaussian random vectors to locate outliers in multiple univariate directions. By using the guarantees of Johnson-Lindenstrauss projections, we can use these univariate filters to ensure the next (weighted) empirical covariance matrix satisfies

$$\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \leq O(1)\mathrm{Tr}(\mathbf{Y}_t^2) \; . \tag{8.1}$$

Combining (8.1) with a fact from [289] shows that our potential decays geometrically, resulting in rapid convergence. Fortunately, we can use the same potential in the multifilter context, as long as we guarantee that (8.1) holds for *every* child of a node (whether a split or cluster step is used). In particular, applying (8.1) repeatedly for any path in the multifilter tree implies that the depth is polylog(d). It remains to bound the *width* of the tree (the computational cost per layer), while maintaining the invariant that at least one node on every level preserves enough inliers.

Warmup: fast Gaussian multifilter via indicator weights. Recall that our other obstacle towards an almost-linear runtime is that each of the $\Omega(\alpha^{-1})$ nodes of our multifilter tree requires $\Omega(nd)$ time to decide on a multifiltering step. Our strategy is to reduce the *total number of nodes* across each layer of the tree, so that the total cost of multifiltering on all of them is roughly nd. We achieve this goal by ensuring that our multifilter always maintains nodes which specify subsets of our original data (i.e. 0-1 weights rather than soft weights $\in [0, 1]$). Hence, each layer of our new multifilter trades off the *number of subsets* with the *cost of multifiltering* on each subset. Considering the two extreme layers is illustrative of this tradeoff: at the root, our algorithm performs a single one-dimensional projection on the entire dataset; at the leaves, it performs $O(\alpha^{-1})$ one-dimensional projections, each on a subset consisting of roughly an α-fraction of the original dataset.

As a warmup, we first show how to achieve this in the case where the ground-truth, dist, is a bounded-covariance Gaussian (see Section G.5), so we can exploit strong concentration bounds. In particular, we know that in any linear projection almost all of the inliers will lie in an interval of logarithmic length. If almost all of our sample points in a subset are clustered within such an interval, we can explicitly remove all samples outside of it. On the other hand, if our samples are spread out, we can split them into two (unweighted) subsets with sufficient overlap to ensure that at least one of the children subsets will contain almost all the inliers, as long as the parent did. We can in fact apply such a partitioning strategy iteratively along each univariate projection, until each remaining subset is contained in a short interval; this suffices to imply (8.1).

From Gaussians to bounded-covariance distributions. Substantially more technical care is required in the bounded-covariance setting to achieve an almost-linear runtime without sacrificing the error rate. Notably, we will no longer be able to guarantee that the subsets lie in short intervals, due to weaker concentration properties. This also means that we cannot deterministically remove points, making it more challenging to ensure the weight functions we keep are indicators.

We overcome these challenges in Section 8.4 through several new technical developments. We

first weaken the outcome guarantee of our recursive partitioning strategy, from ensuring each cluster lies in a short interval, to requiring bounded variance, which we show suffices to advance on the potential (8.1). Furthermore, we use a randomized dropout strategy in place of the "clustering" step of the multifilter, and design fast quantile checks to ensure the "split" step can be conducted in nearly-constant time. By carefully combining these subroutines, we can indeed ensure every child of nodes in a layer satisfies (8.1), and that the total computational cost of splitting or clustering on the entire layer is almost-linear. With our earlier depth bound, this yields our full runtime guarantee.

Reducing clustering to list-decodable learning. In Section 8.5, we demonstrate that several mixture model clustering tasks enjoy benefits from the speedups afforded by our list-decodable learning methods. In the following, assume we have a list L of size $O(\alpha^{-1})$ and $L \supseteq \{\hat{\mu}_i\}_{i\in[k]}$ with $\|\hat{\mu}_i - \mu_i\|_2 = \widetilde{O}(\sqrt{\alpha^{-1}})$ for all $i \in [k]$, where μ_i is the mean of the mixture component dist_i.

For sub-Gaussian components, we build on a clustering algorithm of [186] and improve it to run in nearly-linear time via randomized distance comparisons. The main idea of the [186] algorithm is to exploit concentration, which implies that with high probability, all points drawn from dist_i have a closest hypothesis in L at distance $\widetilde{O}(\sqrt{\alpha^{-1}})$ from μ_i. By rounding every sample to its nearest hypothesis, and assuming separation $\widetilde{\Omega}(\sqrt{\alpha^{-1}})$ between component means, we can perform an efficient equivalence class partitioning which clusters the data. We observe that this framework is tolerant to a small amount of poorly-behaved points or outliers and generalizes to cluster components with bounded fourth moments.

For our most general application of clustering mixtures under only bounded component covariances, as stated in Theorem 50, the same framework does not apply as a constant fraction of all points may be misbehaved due to weak concentration. To address this, we develop a new post-processing technique, relying on the following observation: letting $\mathbf{P}$ be the projection onto the $O(\alpha^{-1})$-dimensional subspace spanned by L, any sample hit by $\mathbf{P}$ will lie within distance $O(\sqrt{\alpha^{-1}})$ of its corresponding cluster mean in the low-dimensional subspace with constant probability. We use this observation to drop hypotheses which are too far away from the true means, and then an appropriate equivalence relation suffices for clustering. The runtime bottleneck of this strategy is the computation and application of $\mathbf{P}$ to our dataset, which can be quite expensive. We show that by instead measuring distances in a $O(\log d)$-dimensional subspace formed by random projections within $\mathbf{P}$, and clustering based on these estimates, we obtain similar clustering performance by exploiting guarantees of Johnson-Lindenstrauss transforms.

8.3 Preliminaries: clustering mixture models

In Section 8.3.1, we define the notation used throughout the sections relevant to clustering mixture models. Next, we recall some technical tools (primarily from prior work) which we draw upon in Section 8.3.2. We conclude with a sketch of our potential function approach to filtering in

Section 8.3.3 by demonstrating how it works for robust mean estimation in the minority-outlier regime, giving an alternative approach to obtaining the runtimes of [194]. This new approach bypasses an explicit matrix multiplicative weights argument in favor of a one-step potential. We ultimately generalize this technique to the list-decodable setting by interlacing it with clustering steps, inspired by the multifilter algorithm of [186, 179].

8.3.1 Notation

General notation. For mean $\mu \in \mathbb{R}^d$ and positive semidefinite covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{d\times d}$, we let $\mathcal{N}(\mu, \mathbf{\Sigma})$ be the standard multivariate Gaussian. For $d \in \mathbb{N}$ we let $[d] := \{j \mid j \in \mathbb{N}, 1 \le j \le d\}$. We refer to the ℓ_p norm of a vector argument by $\|\cdot\|_p$, and overload this to mean the Schatten-p norm of a symmetric matrix argument. The all-ones vector (when the dimension is clear from context) is denoted $\mathbf{1}$. The (solid) probability simplex is denoted $\Delta^n := \{x \in \mathbb{R}^n_{\ge 0}, \|x\|_1 \le 1\}$. We refer to the i^{th} coordinate of a vector v by $[v]_i$.

Matrices. Matrices will be in boldface throughout, and when the dimension is clear from context we let $\mathbf{0}$ and $\mathbf{I}$ be the zero and identity matrices. The set of $d \times d$ symmetric matrices is $\mathbb{S}^d$ and the $d \times d$ positive semidefinite cone is $\mathbb{S}^d_{\ge 0}$. We use the Loewner partial ordering $\preceq$ on $\mathbb{S}^d_{\ge 0}$. The largest eigenvalue, smallest eigenvalue, and trace of a matrix are given by $\lambda_{\max}(\cdot), \lambda_{\min}(\cdot), \mathrm{Tr}(\cdot)$ respectively. We use $\|\cdot\|_{\text{op}}$ to mean the (ℓ_2-ℓ_2) operator norm, which is the largest eigenvalue for arguments in $\mathbb{S}^d_{\ge 0}$. The inner product on $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d$ is given by $\langle \mathbf{A}, \mathbf{B}\rangle := \mathrm{Tr}(\mathbf{AB})$.

Distributions. We often associate a weight vector $w \in \Delta^n$ with a set of points $T \subset \mathbb{R}^d$ with $|T| = n$. Typically we denote this set by $\{X_i\}_{i\in T}$, where we overload T to mean the indices as well as the points. For any $T' \subseteq T$ we let $w_{T'} \in \Delta^n$ be the vector which agrees with w on T' and is 0 elsewhere. The empirical mean and covariance matrix on any subset are denoted

$$\mu_w(T') := \sum_{i\in T'} \frac{w_i}{\|w'_T\|_1} X_i,\ \mathrm{Cov}_w(T') := \sum_{i\in T'} \frac{w_i}{\|w'_T\|_1} (X_i - \mu_w(T'))(X_i - \mu_w(T'))^\top.$$

For convenience, we also define the unnormalized convariance matrix by

$$\widetilde{\mathrm{Cov}}_w(T') := \sum_{i\in T'} w_i (X_i - \mu_w(T'))(X_i - \mu_w(T'))^\top.$$

We say distribution dist with $\mathbb{E}_{X\sim\text{dist}}[X] = \mu^*$ has sub-Gaussian parameter σ in all directions if $\mathbb{E}_{X\sim\text{dist}}[\exp(s\langle X - \mu^*, v\rangle)] \le \exp(\frac{\sigma^2 s^2}{2})$ for all unit vectors v. In Section 8.5 we use concentration properties of sub-Gaussian random variables, which are well-known and can be found e.g. in the reference [456].

List-decodable mean estimation. We state the model of list-decodable mean estimation we use throughout; the setting we consider is standard from the literature. Fix a parameter $0 < \alpha < \frac{1}{2}$. Then a set $T := \{X_i\}_{i\in T} \subset \mathbb{R}^d$ of size $|T| = n = \Theta(d\alpha^{-1})$ is given, containing a subset $S \subset T$ such

that the following assumption holds.

Assumption 6. *There is a subset $S \subseteq T \subset \mathbb{R}^d$ of size $\alpha n = \Theta(d)$, and a vector $\mu^* \in \mathbb{R}^d$, such that*

$$\frac{1}{|S|}\sum_{i\in S}(X_i - \mu^*)(X_i - \mu^*)^\top \preceq \mathbf{I}.$$

We remark that this assumption is motivated by the statistical model where there is an underlying distribution dist supported on $\mathbb{R}^d$ with mean μ^* and covariance bounded by $\mathbf{I}$, and the dataset T is formed by αn independent draws from dist combined with $(1-\alpha)n$ arbitrary points. Up to constants in the "good" fraction α and the covariance bound, Proposition B.1 of [118] guarantees Assumption 6 holds with inverse-exponential failure probability. We also note that Proposition 5.4(ii) of [186] shows that the information-theoretic optimal guarantee for list-decodable estimation in the setting of Assumption 13 is to return a list L of candidate means, where $|L| = \Theta(\alpha^{-1})$, and

$$\min_{\mu\in L}\|\mu - \mu^*\|_2 = \Theta\left(\frac{1}{\sqrt{\alpha}}\right) \tag{8.2}$$

This setup handles the more general case where the upper bound matrix in Assumption 6 is $\sigma^2\mathbf{I}$ for some positive parameter σ by rescaling the space appropriately, and the error guarantee (8.2) becomes worse by a factor of σ. Because of this, we will set $\sigma = 1$ throughout for simplicity.

Finally, throughout Sections G.5 and 8.4 we will make the explicit assumption that $d \geq \alpha^{-1}$; for the regime where this is not the case, the algorithm in Section G.1 obtains optimal error rates in time $\widetilde{O}(\alpha^{-2})$ (and in fact, if we tolerate a list size of $O(\alpha^{-1}\log\frac{1}{\delta})$ where $\delta \in (0,1)$ is the failure probability of the algorithm, obtains optimal error in time $\widetilde{O}(\alpha^{-1})$).

8.3.2 Technical tools

We will use a number of technical tools throughout this work which we list here for convenience. The first few are standard facts about covariance matrices which follow directly from computation.

Fact 14 (Convexity of covariance)**.** *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$,*

$$\mu_w(T)\mu_w(T)^\top \preceq \sum_{i\in T}\frac{w_i}{\|w\|_1}X_iX_i^\top.$$

This implies that for any $v \in \mathbb{R}^d$,

$$(\mu_w(T) - v)(\mu_w(T) - v)^\top \preceq \sum_{i\in T}\frac{w_i}{\|w\|_1}(X_i - v)(X_i - v)^\top.$$

Fact 15 (Effect of mean shift)**.** *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$, and any $v \in \mathbb{R}^d$,*

$$\sum_{i \in T} \frac{w_i}{\|w\|_1} (X_i - v)(X_i - v)^\top = \mathrm{Cov}_w(T) + (\mu_w(T) - v)(\mu_w(T) - v)^\top \succeq \mathrm{Cov}_w(T).$$

Fact 16 (Alternate covariance characterization)**.** *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$,*

$$\frac{1}{2\|w\|_1^2} \left(\sum_{i,j \in T} w_i w_j (X_i - X_j)(X_i - X_j)^\top \right) = \mathrm{Cov}_w(T).$$

Next, we need the notion of safe weight removal in the list-decodable setting, adapted from Section G.2. The idea behind safe weight removal is that repeatedly performing a downweighting operation with respect to scores satisfying a certain condition results in weights which preserve some invariant, which we call saturation. We define our notions of safety and saturation, and state a key technical lemma which lets us reason about when saturation is preserved. In the following discussion assume we are in the list-decodable mean estimation setting we defined in Section 8.3.1.

Definition 28 (γ-saturated weights)**.** *We call weights $w \in \Delta^n$ γ-saturated, for some $\gamma > 1$, if $w \le \frac{1}{n}\mathbf{1}$ entrywise, and*

$$\|w_S\|_1 \ge \alpha \|w\|_1^{\frac{1}{\gamma}}.$$

Definition 29 (γ-safe scores)**.** *We call scores $\{s_i\}_{i \in T} \in \mathbb{R}^n_{\ge 0}$ γ-safe with respect to w if $w \in \Delta^n$ and*

$$\sum_{i \in S} \frac{w_i}{\|w_S\|_1} s_i \le \frac{1}{\gamma} \sum_{i \in T} \frac{w_i}{\|w_T\|_1} s_i.$$

Roughly speaking, we require this alternative notion of safe scores in the majority-outlier regime because there are less good points we can afford to throw away. The key property connecting these two definitions is the following.

Lemma 112. *Let $w^{(0)} \in \Delta^n$ be γ-saturated, and consider any algorithm of the form:*

1. *For $0 \le t < N$:*
   (a) *Let $\{s_i^{(t)}\}_{i \in T}$ be γ-safe with respect to $w^{(t)}$.*
   (b) *Update for all $i \in T$:*

$$w_i^{(t+1)} \leftarrow \left(1 - \frac{s_i^{(t)}}{s_{\max}^{(t)}}\right) w_i^{(t)}, \text{ where } s_{\max}^{(t)} := \max_{i \in T | w_i^{(t)} \neq 0} s_i^{(t)}.$$

Then, $w^{(N)}$ is also γ-saturated.

Proof. It suffices to prove this in the case $N = 1$ and then use induction. Define

$$\delta_S := \sum_{i \in S} \frac{w_i^{(0)} - w_i^{(1)}}{\left\|w_S^{(0)}\right\|_1}, \ \delta_T := \sum_{i \in T} \frac{w_i^{(0)} - w_i^{(1)}}{\left\|w^{(0)}\right\|_1}.$$

By using the assumption that $s^{(0)}$ is γ-safe, we conclude

$$\delta_S = \frac{1}{s_{\max}^{(0)}} \sum_{i \in S} \frac{w_i^{(0)}}{\left\|w_S^{(0)}\right\|_1} s_i^{(0)} \le \frac{1}{\gamma} \cdot \frac{1}{s_{\max}^{(0)}} \sum_{i \in T} \frac{w_i^{(0)}}{\left\|w^{(0)}\right\|_1} s_i^{(0)} = \frac{1}{\gamma} \delta_T.$$

Now, using γ-saturation of $w^{(0)}$ and $1 - \gamma^{-1}\delta \ge (1-\delta)^{\gamma^{-1}}$ for all $\delta \in [0,1]$ and $\gamma > 1$,

$$\left\|w_S^{(1)}\right\|_1 = (1-\delta_S)\left\|w_S^{(0)}\right\|_1 \ge (1-\delta_T)^{\gamma^{-1}} \left\|w_S^{(0)}\right\|_1 \ge \alpha \left((1-\delta_T)\left\|w^{(0)}\right\|_1\right)^{\gamma^{-1}} = \alpha \left\|w^{(1)}\right\|_1^{\gamma^{-1}}.$$

□

Finally, we include a technical lemma proved in [136].

Lemma 113 (Restatement of Lemma 296)**.** *Let $w \in \Delta^n$ have $w \le \frac{1}{n}\mathbf{1}$ entrywise, and $\|w\|_1 \ge \alpha^2$. Then,*

$$\|\mu_w(T) - \mu^*\|_2 \le \sqrt{2 \left\|\mathrm{Cov}_w(T)\right\|_{\mathrm{op}} \frac{\|w\|_1}{\|w_S\|_1} + \frac{2}{\alpha}}.$$

In light of the lower bound of [186], Lemma 113 shows to learn the mean near-optimally in the list-decodable setting, it suffices to ensure $\|w_S\|_1 = \Omega(\alpha)$ (i.e. we retain a constant fraction of the "good" weight) and $\left\|\mathrm{Cov}_w(T)\right\|_{\mathrm{op}} = \widetilde{O}(1)$ (i.e. the weighted covariance of the dataset is bounded).

8.3.3 Potential function approach to fast filtering

In this section, we outline an example of a potential function approach to fast filtering, an alternative to filtering based on matrix multiplicative weights (MMW) used in recent literature [194, 359].[5] This replacement is very useful in the list-decodable setting, as it greatly simplifies the requirements of our fast multifilter which interlaces clustering and filtering steps.

The example problem we consider in this expository section is the minority-outlier regime for robust mean estimation, when the "ground truth" distribution has covariance norm bounded by $\mathbf{I}$. We briefly describe the approach of [194] for this problem, and explain how it can be replaced with our new potential function framework. Throughout this section, fix some $0 < \epsilon < \frac{1}{2}$ and assume that amongst the dataset $T \subset \mathbb{R}^d$ of n points, there is a majority subset $S \subset T$ of size $|S| = (1-\epsilon)n$ with bounded empirical covariance: $\mathrm{Cov}_{\frac{1}{n}\mathbf{1}}(S) \preceq \mathbf{I}$.

[5] MMW guarantees are also implicitly used in approaches based on packing SDPs, see e.g. [131, 132, 136]. However, [194, 359] use MMW guarantees in a non-black-box way to design filters.

The main algorithmic step in [194] is an efficient subroutine for halving the operator norm of the empirical covariance while filtering more weight from $T \setminus S$ than from S. It is well-known in the literature that whenever the operator norm is $O(1)$, the empirical mean is within $O(\sqrt{\epsilon})$ in ℓ_2 norm from the ground truth mean (for an example of this derivation, see Lemma 3.2 of [194]). Thus, the key technical challenge is to provide a nearly-linear time implementation of this subroutine. This was accomplished in [194] using MMW-based regret guarantees, with "gain matrices" given by covariances and iterative filtering based on MMW responses. The result was a procedure which either terminates with a good mean estimate, or halves the covariance operator norm after $O(\log d)$ rounds of filtering. The latter is an artifact of many regret minimization techniques, which only guarantee progress after multiple rounds. It is natural to ask instead whether an alternative one-shot potential decrease guarantee exists; we now describe such a guarantee.

One-shot potential decrease. Our algorithm will proceed in a number of iterations, where we modify a weight vector in Δ^n associated with T. We initialize $w^{(0)} \leftarrow \frac{1}{n}\mathbf{1}$. In iteration t, we will downweight $w^{(t)} \in \Delta^n$ to obtain a new vector $w^{(t+1)}$ as follows. Define the matrices

$$\mathbf{M}_t := \widetilde{\mathrm{Cov}}_{w^{(t)}}(T) = \sum_{i \in T} w_i^{(t)} (X_i - \mu_{w^{(t)}}(T))(X_i - \mu_{w^{(t)}}(T))^\top, \ \mathbf{Y}_t := \mathbf{M}_t^{\log d}.$$

The potential we will track is $\Phi_t := \mathrm{Tr}(\mathbf{Y}_t^2)$. In order to analyze Φ_t, we require two helper facts.

Fact 17 (Lemma 7, [289])**.** *Let $\mathbf{A} \succeq \mathbf{B}$ be matrices in $\mathbb{S}^d_{\succeq 0}$, and let $p \in \mathbb{N}$. Then*

$$\mathrm{Tr}(\mathbf{A}^{p-1}\mathbf{B}) \geq \mathrm{Tr}(\mathbf{B}^p).$$

Fact 18. *For any $\gamma \geq 0$ and $\mathbf{A} \in \mathbb{S}^d_{\succeq 0}$,*

$$\gamma \mathrm{Tr}(\mathbf{A}^{2\log d}) \leq \mathrm{Tr}(\mathbf{A}^{2\log d + 1}) + d\gamma^{2\log d + 1}.$$

Proof. This is immediate since each of the d eigenvalues of $\mathbf{A}^{2\log d}$ is either at least $\gamma^{2\log d}$ or not, and both of these cases are accounted for on the right hand side of the conclusion. □

We now give the potential analysis. Our main goal will be ensuring that

$$\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \leq 20 \mathrm{Tr}(\mathbf{Y}_t^2). \tag{8.3}$$

The specific constant in the above equation is not particularly important, but is used for illustration.

We now show how (8.3) implies a rapid potential decrease. Observe that

$$\begin{aligned}\Phi_{t+1} = \mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d}\right) &\le \frac{1}{40}\mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d+1}\right) + d(40)^{2\log d} \\ &\le \frac{1}{40}\mathrm{Tr}\left(\mathbf{M}_{t}^{2\log d}\mathbf{M}_{t+1}\right) + d(40)^{2\log d} \\ &\le \frac{1}{2}\mathrm{Tr}(\mathbf{Y}_t^2) + d(40)^{2\log d} = \frac{1}{2}\Phi_t + d(40)^{2\log d}.\end{aligned} \tag{8.4}$$

The first line used Fact 18 with $\gamma = 40$, the second used Fact 17 with $\mathbf{A} = \mathbf{M}_t$ and $\mathbf{B} = \mathbf{M}_{t+1}$ (noting that if $w^{(t+1)} \le w^{(t)}$ entrywise, the unnormalized covariance matrices respect the Loewner order by Fact 15), and the third line used the assumption (8.3). This implies that we decrease Φ_t by a constant factor in every iteration, until it is roughly $d(40)^{2\log d}$, at which point the definition $\Phi_t = \mathrm{Tr}(\mathbf{M}_t^{2\log d})$ implies that $\|\mathbf{M}_t\|_{\mathrm{op}}$ is bounded by a constant. By using a naïve filtering preprocessing step, we can guarantee that $\Phi_0 = d^{O(\log d)}$, and hence the process will terminate in $O(\log^2 d)$ rounds.

Meeting the filter criterion (8.3). To complete the outline of this algorithm, we need to explain how to satisfy (8.3) via downweighting, while ensuring that we remove more weight from $T \setminus S$ than S. To do so, we define scores

$$s_i^{(t)} := (X_i - \mu_{w^{(t)}}(T))^\top \mathbf{M}_t^{2\log d}(X_i - \mu_{w^{(t)}}(T)) \text{ for all } i \in T.$$

We remark that (randomized) constant-factor approximations can be computed to all $s_i^{(t)}$ via Johnson-Lindenstrauss projections in $\widetilde{O}(nd)$ time, but for this discussion we assume we exactly know all scores. Then, by linearity of trace the condition (8.3) is implied by

$$\sum_{i\in T} w_i^{(t+1)} s_i^{(t)} \le 20\mathrm{Tr}(\mathbf{Y}_t^2), \tag{8.5}$$

since Fact 15 implies that $\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1}\right\rangle \le \sum_{i\in T} w_i^{(t+1)} s_i^{(t)}$. Finally, we note that whenever (8.5) does not hold, it must be primarily due to the effect of the outliers $T \setminus S$, because the covariance of S is bounded. Hence, we can simply set

$$w_i^{(t+1)} = \left(1 - \frac{s_i^{(t)}}{s_{\max}^{(t)}}\right)^K w_i^{(t)},$$

where K is the smallest natural number which passes the criterion (8.5). For any smaller K, it can be shown that downweighting "one more time" preserves the invariant that more outlier mass is removed, precisely because (8.5) has not been met. Finally, binary searching to find the smallest value of K meeting (8.3) yields a complete algorithm running in $\widetilde{O}(nd)$ time (for further details on the implementation of this binary search on K, see Theorem 2.4 of [194]).

Generalizing to the majority-outlier regime. Our algorithms for the list-decodable setting marry

this potential function argument with a multifilter, which produces multiple candidate filtered weight vectors on an input weight vector. We will instead show that for the *tree* of weight vectors, where a node has children given by the candidates produced from the multifilter, at least one child both halves the potential and performs only γ-safe weight removal, for some γ. After polylogarithmic layers, we will return all empirical means of leaf nodes as our list of estimates. The child on the "safe branch" will then have a bounded potential and a γ-saturated weight vector, which suffices to guarantee an accurate mean estimate.

There are a number of additional complications which arise in this extension, which we briefly mention here as a preface to the following Sections G.5 and 8.4. In order to process every layer of the multifilter tree in almost-linear time, we need to ensure that the number of datapoints across all the nodes, including repetitions, has not grown by more than a constant factor. The multifilter of [179] gives a variant of this guarantee by tracking the sums of the squared ℓ_1 norms of weights associated with different nodes as a nonincreasing potential, i.e.

$$\sum_{i \in \mathcal{S}} \left\| w^{(i)} \right\|_1^2,$$

where $\mathcal{S}$ is the set of nodes and each $w^{(i)}$ is a current candidate weight function in $\mathcal{S}$. This is not sufficiently strong of a guarantee in our setting, since even points with very small weights need to be factored into calculations and thus affect runtime. We modify this approach in two ways. First, we replace the downweighting step with a randomly subsampled filter, which we show preserves various safety conditions such as those in Lemma 112 with high probability. Next, we replace the squared ℓ_1 potential with one involving $1 + \beta$ powers, for some $\beta \in (0, 1)$, which we prove is compatible with the multifilter. Overall, our filter tree contains polylogarithmically many layers, each of which accounts for sets with total cardinality $O(n^{1+O(\beta)})$, giving us our final runtime.

8.4 Fast bounded covariance multifilter

In this section, we give our algorithm for list-decodable mean estimation under only Assumption 13. As before, we can assume without loss of generality that $\alpha \in [1/d, 1/\log^C d]$, for some constant $C > 0$. We begin by giving our main subroutine, Partition, in Section 8.4.1. The goal of Partition will be to produce child subsets $\{c_\ell\}_{\ell \in [k]}$ of a given input set p, which each satisfy the potential criterion in (G.14), reproduced here:

$$\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \right\rangle \le R^2 \mathrm{Tr}\left(\mathbf{Y}_p^2\right). \tag{8.6}$$

Recall that in Section G.5, the way we produced children satisfying condition (8.6) was by ensuring that along logarithmically many random directions, each child c_ℓ lied entirely in short intervals. We will satisfy this guarantee in this section by more directly working with the definition of (8.6), which requires each child to have small variance along the random directions, a looser condition.

To bound the variance of the child subsets, in Section 8.4.2 we develop an algorithm, SplitOrCluster, which is patterned off our earlier GaussianSplitOrCluster. It either certifies that the input set is already "close" to having bounded variance in an input direction, or identifies a split point which produces two subsets which are closer to having this property, while maintaining at least one subset retains most points in S. In the first case (the "cluster" case), we develop a postprocessing procedure Fixing in Section 8.4.4 which randomly filters points according to safe outlier scores (see Definition 29) to make the remaining cluster have truly bounded variance. In the second case (the "split" case), we develop a fast threshold checking procedure SplitOrTailBound in Section 8.4.3 which identifies a valid split in polylogarithmic time, whenever one exists; here, we note the key difficulty is that we can no longer use a fixed radius for splits, because Gaussian concentration does not hold.

We discuss runtimes of all of these algorithms in Section 8.4.5, and in particular give a runtime bound on Partition. Finally, we use Partition to develop our full algorithm, FastMultifilter, which we analyze in Section 8.4.6 through a potential argument similar to our analysis of FastGaussianMultifilter. A post-processing step used in FastMultifilter is analyzed in Section 8.4.7.

8.4.1 Reducing Partition to SplitOrCluster

The goal of this section is to develop Partition, the main subroutine of FastMultifilter. Partition has very similar guarantees to the algorithm GaussianPartition developed in Section G.5.1. It takes as input a "parent set" $T_p \subseteq T$ and produces a number of "children subsets" $\{T_{c_\ell}\}_{\ell \in [k]}$ such that every child satisfies $\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \right\rangle \le R^2 \mathrm{Tr}\left(\mathbf{Y}_p^2\right)$, where we follow the definitions (G.13), reproduced here:

$$\begin{aligned} \mathbf{M}_p &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_p), \ \mathbf{Y}_p := \mathbf{M}_p^{\log d}, \\ \mathbf{M}_{c_\ell} &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{c_\ell}), \ \mathbf{Y}_{c_\ell} := \mathbf{M}_{c_\ell}^{\log d} \text{ for all } \ell \in [k], \end{aligned} \tag{8.7}$$

This will allow us to conduct a potential analysis to bound the depth of the multifilter tree in Section 8.4.6. Moreover, we require two additional guarantees of Partition.

1. The first is the same as (G.12); namely, for some parameter $\beta \in (0, 1]$, we have

$$\sum_{\ell \in [k]} |T_{c_\ell}|^{1+\beta} \le |T_p|^{1+\beta}. \tag{8.8}$$

   This will help us bound the total work done in each layer of the multifilter tree.

2. The second is ensures at least one child preserves most points in S, assuming that the parent T_p has this property. To this end, the tools of Section 8.3.2 will vastly simplify the language of this section. In particular, we will ensure that for $\gamma = O(\log(\frac{1}{\alpha}))$, every filter step in this entire section will be with respect to γ-safe weights in at least one branch. We then apply Lemma 112 to conclude that some node at every level of the multifilter tree is γ-saturated.

For the remainder of Section 8.4, we will define

$$\gamma := 8 \log\left(\frac{1}{\alpha}\right).$$

We demonstrate one important consequence of a set being γ-saturated.

Lemma 114. *Suppose for a set $T' \subset T$, the weights $w := \frac{1}{n}\mathbf{1}_{T'} \in \Delta^n$ which place $\frac{1}{n}$ on coordinates in T' and 0 otherwise are γ-saturated (cf. Definition 28). Then,*

$$|T' \cap S| \geq \frac{\alpha n}{2}.$$

Proof. Recall that Definition 28 gives

$$\|w\|_1 \geq \|w_S\|_1 \geq \alpha \|w\|_1^{\frac{1}{\gamma}} \implies \|w\|_1^{1-\frac{1}{\gamma}} \geq \alpha \implies \|w\|_1 \geq \frac{3\alpha}{4}.$$

The first implication was by rearrangement, and the second used $\alpha^{\frac{1}{1+\gamma^{-1}}} \geq \alpha^{1+\frac{2}{\gamma}} \geq \frac{3\alpha}{4}$. Next,

$$\|w_S\|_1 \geq \alpha \|w\|_1^{\frac{1}{\gamma}} \geq \alpha \left(\frac{3\alpha}{4}\right)^{\frac{1}{\gamma}} \geq \frac{\alpha}{2}.$$

The conclusion follows since $\|w_S\|_1$ counts the elements of $T' \cap S$, normalized by $\frac{1}{n}$. □

Lemmas 112 and 114 imply that as long as we can guarantee that at every filtering step, at least one child was produced with respect to γ-safe scores, that child retains half the elements of S. We are now ready to state the algorithm Partition, which heavily relies on a subroutine 1DPartition.

As in the Gaussian case, we develop an algorithm 1DPartition which in turn is based on a subroutine SplitOrCluster, which we implement in Section 8.4.2. The algorithm 1DPartition takes an input direction v and guarantees that along this direction, every child subset produced has small variance (scaled by the length of v). 1DPartition is implemented by recursively calling SplitOrCluster, which takes as input a set T'' and produces either one or two subsets, analogously to GaussianSplitOrCluster.

Lemma 115. *The output of* Partition *satisfies the guarantees given in Line 2 of Algorithm 33, assuming correctness of* 1DPartition*.*

Proof. We will follow the proof of Lemma 307. First, to demonstrate that the subsets satisfy (8.8), inducting on the guarantee (8.9) of 1DPartition suffices. Similarly, γ-saturation of some child follows from inducting on the corresponding guarantee of 1DPartition.

Finally, using the guarantee (8.10) of 1DPartition, every $T_{c_\ell} \in \mathcal{S}_{N_{\mathrm{dir}}}$ satisfies

$$\left\langle v_j v_j^\top, \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{c_\ell}) \right\rangle \leq \frac{1}{2} R^2 \|v_j\|_2^2, \text{ for all } j \in [N_{\mathrm{dir}}].$$

Algorithm 33: $\mathsf{Partition}(T_p, \alpha, \delta, \beta, R)$

Input: $T_p \subset T$, $\alpha \in (0, \frac{1}{2})$, $\delta \in (0,1)$, $\beta \in (0,1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega\left(\max\left(\frac{1}{\beta} \cdot \sqrt{\gamma \log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\gamma \log\left(\frac{\log d}{\delta}\right)}\right)\right).$$

;

Output: With failure probability $\leq \delta$: subsets $\{T_{c_\ell}\}_{\ell \in [k]}$ of T_p, satisfying (8.8). Every child satisfies (8.6) (using notation (8.7)). If $w := \frac{1}{n}\mathbf{1}_{T_p}$ is γ-saturated, then $w' := \frac{1}{n}\mathbf{1}_{T_{c_\ell}}$ is γ-saturated for at least one child T_{c_ℓ}.;

Sample $N_{\mathrm{dir}} = \Theta(\log \frac{d}{\delta})$ vectors $\{u_j\}_{j \in [N_{\mathrm{dir}}]} \in \mathbb{R}^d$ each with independent entries ± 1. Following notation (8.7), let $v_j \leftarrow \mathbf{Y}_p u_j$ for all $j \in [N_{\mathrm{dir}}]$.;

$\mathcal{S}_0 \leftarrow T_p$;

for $j \in [N_{\mathrm{dir}}]$ **do**

  $\mathcal{S}_j \leftarrow \emptyset$;

  for $T' \in \mathcal{S}_{j-1}$ **do**

    $\mathcal{T} \leftarrow \mathsf{1DPartition}(T', \alpha, v_j, \frac{\delta}{2N_{\mathrm{dir}}}, \beta, R)$;

    $\mathcal{S}_j \leftarrow \mathcal{S}_j \cup \mathcal{T}$;

Return: $\mathcal{S}_{N_{\mathrm{dir}}}$;

In other words, the variance is small along all directions $\{\mathbf{Y}_p u_j = v_j\}_{j \in [N_{\mathrm{dir}}]}$. We conclude

$$\begin{aligned}
\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_l} \right\rangle &\leq \frac{1.4}{2n\left|T_{c_\ell}\right| N_{\mathrm{dir}}} \sum_{i,i' \in T_{c_l}} \sum_{j \in [N_{\mathrm{dir}}]} \left\langle \mathbf{Y}_p u_j, X_i - X_{i'} \right\rangle^2 \\
&= \frac{1.4}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \left\langle v_j v_j^\top, \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{c_\ell}) \right\rangle \\
&\leq \frac{1.4}{2N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} R^2 \left\|v_j\right\|_2^2 \\
&= \frac{1.4}{2N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} R^2 \left\|\mathbf{Y}_p u_j\right\|_2^2 \leq R^2 \mathrm{Tr}\left(\mathbf{Y}_p^2\right),
\end{aligned}$$

with probability at least $1 - \delta$; the first two lines and the last line follow the proof of Lemma 307, and we used the variance guarantee of $\mathsf{1DPartition}$ in the third line. We remark that we make sure to take the number of directions N_{dir} to depend logarithmically on δ (as opposed to just d), so we can apply the guarantees of [6] with probability $1 - \frac{\delta}{2}$ on the first and last lines. □

Next, we state a correctness guarantee for $\mathsf{1DPartition}$, assuming correctness of its main subroutine, $\mathsf{SplitOrCluster}$. The guarantees of $\mathsf{SplitOrCluster}$ are summarized in Line 2 of Algorithm 35. The salient features are that it takes a set T_{in} and either produces one set satisfying (8.10) deterministically, or two sets which each are strict subsets of T_{in} (and hence remove at least one point)

Algorithm 34: 1DPartition$(T', \alpha, v, \delta, \beta, R)$

Input: $T' \subset T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0,1)$, $\beta \in (0,1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega\left(\max\left(\frac{1}{\beta} \cdot \sqrt{\gamma \log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\gamma \log\left(\frac{\log d}{\delta}\right)}\right)\right).$$

;

Output: Subsets $\{T''_\ell\}_{\ell \in [k]}$ of T', such that

$$\sum_{\ell \in [k]} |T''_\ell|^{1+\beta} \leq |T'|^{1+\beta}. \tag{8.9}$$

Every child T''_ℓ for $\ell \in [k]$ has

$$\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T''_\ell), vv^\top \right\rangle \leq \frac{1}{2} R^2 \|v\|_2^2 . \tag{8.10}$$

If $w = \frac{1}{n}\mathbf{1}_{T'}$ is γ-saturated, then $w' = \frac{1}{n}\mathbf{1}_{T''_\ell}$ is γ-saturated for at least one child T''_ℓ, with failure probability $\leq \delta$.;

$\mathcal{S}_{\mathrm{in}} \leftarrow T'$, $\mathcal{S}_{\mathrm{out}} \leftarrow \emptyset$;

while $\mathcal{S}_{in} \neq \emptyset$ **do**

  $T'' \leftarrow$ the first element of $\mathcal{S}_{\mathrm{in}}$;

  $\mathcal{S}_{\mathrm{in}} \leftarrow \mathcal{S}_{\mathrm{in}} \setminus \{T''\}$;

  if SplitOrCluster$(T'', \alpha, v, \delta, \beta, R)$ *returns one set* $T^{(0)}_{\mathrm{out}}$ **then**

    $\mathcal{S}_{\mathrm{out}} \leftarrow \mathcal{S}_{\mathrm{out}} \cup \left\{T^{(0)}_{\mathrm{out}}\right\}$;

  else

    $T^{(1)}_{\mathrm{out}}, T^{(2)}_{\mathrm{out}} \leftarrow$ SplitOrCluster$(T'', \alpha, v, \delta, \beta, R)$;

    $\mathcal{S}_{\mathrm{in}} \leftarrow \mathcal{S}_{\mathrm{in}} \cup \left\{T^{(1)}_{\mathrm{out}}, T^{(2)}_{\mathrm{out}}\right\}$;

Return: $\mathcal{S}_{\mathrm{out}}$;

deterministically, and (8.9) is always maintained. In the two set case, if $\frac{1}{n}\mathbf{1}_{T_{\text{in}}}$ is γ-saturated then so is at least one output deterministically; in the one set case, the output is saturated with probability at least $1-\delta$. We will use only these features to analyze 1DPartition in Lemma 116.

Lemma 116. *The output of* 1DPartition *satisfies the guarantees given in Line 2 of Algorithm 34, assuming correctness of* SplitOrCluster*.*

Proof. Each run of Lines 4-13 results in either one set (which we call a "cluster step") or two sets (which we call a "split step"). We can view this process as a tree, where a leaf node corresponds to the result of a cluster step, and every node on the leaf-to-node path corresponds to a split step. Every time a split step occurs, it increases $|\mathcal{S}_{\text{in}}| + |\mathcal{S}_{\text{out}}|$ by one, so there are at most $(n')^{1+\beta}$ calls to SplitOrCluster where $|T'| = n'$, and thus the algorithm terminates in finite time.

Next, if T' is not saturated, then there is no failure probability, since the conditions (8.9) and (8.10) deterministically succeed. Otherwise, from the root of this partition tree, consider the root-to-leaf path which at each node corresponding to a split step takes any child which corresponds to a saturated child (one always exists because split steps deterministically succeed). The only failure probability comes from the success of the leaf-parent to leaf cluster step, which fails with probability δ. We can ignore all other bad events, because we only need to ensure one child is saturated. □

8.4.2 Reducing SplitOrCluster to SplitOrTailBound and Fixing

In this section, we state and analyze SplitOrCluster, the main subroutine of 1DPartition.

SplitOrCluster uses two subroutines, Fixing and SplitOrTailBound, which are respectively used to handle the one child and two children cases. Roughly speaking, Fixing takes as input a set T_{in} which "almost" has bounded variance in the direction v, and slightly filters extreme outliers in a way so that the result has truly bounded variance. On the other hand, SplitOrTailBound is used at a candidate threshold τ to either check that it induces sets $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ satisfying (8.12) or satisfies a certain tail bound. By stitching together tail bounds at a small number of quantiles, SplitOrCluster guarantees that at least one of these quantiles was a valid threshold, else we would attain a contradiction as Line 6 of SplitOrCluster would have passed. We state guarantees of SplitOrTailBound and Fixing as Lemmas 117 and 118, and prove them in Sections 8.4.3 and 8.4.4 respectively. We then use Lemmas 117 and 118 to prove Lemma 119, which demonstrates correctness of SplitOrCluster.

Lemma 117. *There is an algorithm,* SplitOrTailBound *(Algorithm 36), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0,1]$, $R \in \mathbb{R}_{\geq 0}$, and $\tau_0 \in \mathbb{R}$, and returns in one of two cases we call the "split" case and the "tail bound" case. In the split case, it returns (τ, r) such that the induced sets* (8.11) *satisfy* (8.12)*, and if $\frac{1}{n}\mathbf{1}_{T_{\text{in}}}$ is γ-saturated then one of the induced sets T_{out} has $\frac{1}{n}\mathbf{1}_{T_{\text{out}}}$ is γ-saturated. Otherwise, for all $t \in \mathbb{R}$ define the upper and lower tail probabilities*

$$\rho^+(t) := \Pr_{i \sim_{\text{unif}} T_{\text{in}}}[Y_i \geq t]\,,\ \rho^-(t) := \Pr_{i \sim_{\text{unif}} T_{\text{in}}}[Y_i \leq t]\,. \tag{8.13}$$

Algorithm 35: SplitOrCluster$(T_{\text{in}}, \alpha, v, \delta, \beta, R)$

Input: $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0,1)$, $\beta \in (0,1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega\left(\max\left(\frac{1}{\beta} \cdot \sqrt{\gamma \log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\gamma \log\left(\frac{\log d}{\delta}\right)}\right)\right).$$

;

Output: Either one subset $T_{\text{out}}^{(0)} \subset T_{\text{in}}$, or two subsets $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)} \subset T_{\text{in}}$. In the one subset case, $T_{\text{out}}^{(0)}$ has $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{out}}^{(0)}\right), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2$. In the two subsets case, they take the form, for some threshold value $\tau \in \mathbb{R}$ and $r \in \mathbb{R}_{\geq 0}$

$$T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i \rangle \leq \tau + r\|v\|_2\},\ T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i \rangle \geq \tau - r\|v\|_2\}, \tag{8.11}$$

and satisfy

$$\begin{aligned} \left|T_{\text{out}}^{(1)}\right|^{1+\beta} + \left|T_{\text{out}}^{(2)}\right|^{1+\beta} &< |T_{\text{in}}|^{1+\beta}, \\ \min\left(1 - \frac{\text{abs}\, T_{\text{out}}^{(1)}}{\text{abs}\, T_{\text{in}}}, 1 - \frac{\text{abs}\, T_{\text{out}}^{(2)}}{\text{abs}\, T_{\text{in}}}\right) &\geq \frac{2\gamma}{r^2}. \end{aligned} \tag{8.12}$$

In either case if $\frac{1}{n}\mathbf{1}_{T_{\text{in}}}$ is γ-saturated then $\frac{1}{n}\mathbf{1}_{T_{\text{out}}}$ is γ-saturated for at least one child T_{out}, with failure probability $\leq \delta$ only in the case one set is returned (deterministically otherwise).;

$Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$;

$\tau_{\text{med}} \leftarrow \text{med}\left(\{Y_i \mid i \in T_{\text{in}}\}\right)$, where med returns the median;

$I \leftarrow [\tau_{\text{med}} - c, \tau_{\text{med}} + c]$ is the smallest interval containing the $1 - \frac{\alpha}{4}$ quantiles of $\{Y_i \mid i \in T_{\text{in}}\}$ for $c \in \mathbb{R}_{\geq 0}$ and $2I \leftarrow [\tau_{\text{med}} - 2c, \tau_{\text{med}} + 2c]$;

if $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{mid}}\right), vv^\top \right\rangle \leq \frac{1}{8}R^2\|v\|_2^2$ *where* $T_{\text{mid}} := \{X_i \in T_{\text{in}} \mid Y_i \in 2I\}$ **then**

  Return: Fixing$(T_{\text{in}}, \alpha, v, \delta, R)$;

else

  Run both SplitOrTailBound $\left(T_{\text{in}}, v, \beta, \tau_{\text{med}} \pm \frac{1}{2^k} \cdot \sqrt{\frac{2048}{\beta^2\alpha}}\|v\|_2\right)$ for integers k with

$$0 \leq k \leq \log_2\left(\frac{2048}{\beta^2\alpha}\right)$$

  until one returns $\left(T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}\right)$ satisfying (8.11), (8.12);

  Return: $T_{\text{out}}^{(1)}$, $T_{\text{out}}^{(2)}$;

Then, in the tail bound case SplitOrTailBound *certifies*

$$\rho^+(\tau_0) \le \frac{128\gamma}{\beta^2 \left|\tau_0 - \tau_{\text{med}}\right|^2} \|v\|_2^2 \text{ if } \tau_0 \ge \tau_{\text{med}}, \text{ and } \rho^-(\tau_0) \le \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \le \tau_{\text{med}}. \tag{8.14}$$

Lemma 118. *There is an algorithm,* Fixing *(Algorithm 38), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, and $R \in \mathbb{R}_{\ge 0}$ and produces T_{out} with the following guarantee with probability at least $1 - \delta$. Define T_{mid} as in Line 5 of Algorithm 35. Then if $\frac{1}{n}\mathbf{1}_{T_{\text{in}}}$ is γ-saturated, so is $\frac{1}{n}\mathbf{1}_{T_{\text{out}}}$, and if $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{mid}}), vv^\top \right\rangle \le \frac{1}{8}R^2 \|v\|_2^2$, then $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{out}}), vv^\top \right\rangle \le \frac{1}{2}R^2 \|v\|_2^2$.*

Finally, we are ready to prove Lemma 119, the main export of this section.

Lemma 119. *The output of* SplitOrCluster *satisfies the guarantees given in Line 2 of Algorithm 35.*

Proof. If the check in Line 6 passes (the one subset case), correctness of SplitOrCluster follows immediately from the guarantees of Fixing in Lemma 118. We now focus on the two subset case.

Assume throughout this proof that the $\{Y_i\}_{i \in T_{\text{mid}}}$ are ordered by distance to τ_{med}, so $|Y_1 - \tau_{\text{med}}| \le \ldots \le |Y_m - \tau_{\text{med}}|$, where $m := |T_{\text{mid}}|$; we order the remaining elements $i \in T_{\text{in}} \setminus T_{\text{mid}}$ arbitrarily. We also define τ_{med}, T_{mid}, c, I, and $2I$ as in Lines 4-6 of SplitOrCluster. If Line 9 outputs a split for any k, then by Lemma 117, (8.11), (8.12) are satisfied, and the saturation condition is met for one of the children. It remains to show that some value of k will result in the split case of SplitOrTailBound. We show this by contradiction; if all k resulted in SplitOrTailBound returning with a tail bound guarantee, we prove T_{mid} would have passed Line 6. Assume for the remainder of the proof that SplitOrTailBound failed to find a split for all k.

First, we bound the length of the interval $2I$. Because SplitOrCluster failed to return a split for $k = 1$, by combining the corresponding tail bounds (8.14),

$$\Pr_{i \sim_{\text{unif}} T_{\text{in}}}\left[|Y_i - \tau_{\text{med}}| > \sqrt{\frac{512}{\beta^2\alpha}} \|v\|_2\right] \le \frac{\alpha}{4}.$$

Thus, we conclude

$$2I \subset [\tau_{\text{med}} - C\|v\|_2, \tau_{\text{med}} + C\|v\|_2], \text{ for } C := \sqrt{\frac{2048}{\beta^2\alpha}}.$$

We proceed to bound $\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}), vv^\top \right\rangle$ to obtain our desired contradiction. Observe that

$$\begin{aligned}
\widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}) &\preceq \mathrm{Cov}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}) \\
&= \frac{1}{|T_{\mathrm{mid}}|} \sum_{i \in T_{\mathrm{mid}}} \left(X_i - \mu_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}})\right)\left(X_i - \mu_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}})\right)^\top \\
&\preceq \frac{1}{|T_{\mathrm{mid}}|} \sum_{i \in T_{\mathrm{mid}}} \left(X_i - \bar{X}\right)\left(X_i - \bar{X}\right)^\top \text{ where } \left\langle v, \bar{X} \right\rangle = \tau_{\mathrm{med}} \\
\implies \left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}), vv^\top \right\rangle &\le \frac{1}{|T_{\mathrm{mid}}|} \sum_{i \in T_{\mathrm{mid}}} \left(Y_i - \tau_{\mathrm{med}}\right)^2.
\end{aligned} \tag{8.15}$$

Here, we used the definitions of $\widetilde{\mathrm{Cov}}$, Cov in the first two lines, and Fact 15 in the third. The last follows by definition of $\{Y_i\}_{i \in T_{\mathrm{in}}}$ and τ_{med}. Define the random variable Z to be the realization of $\operatorname{abs} Y_i - \tau_{\mathrm{med}}$ for i a uniform draw from T_{mid}, let $Z_i := \operatorname{abs} Y_i - \tau_{\mathrm{med}}$, and let $G(t) = \Pr[Z \ge t]$ be the inverse cumulative density function of Z. Notice that directly expanding implies that (where we let $m := |T_{\mathrm{mid}}|$, $Z_0 := 0$, and recall we argued $Z_m \le C \left\|v\right\|_2$ earlier)

$$\mathbb{E}[Z^2] = \sum_{i \in [m]} (Z_i^2 - Z_{i-1}^2) G(Y_i) = \int_0^{Z_m} 2tG(t)dt \le \int_0^{C\|v\|_2} 2tG(t)dt.$$

Define now $K(t)$ for each $\left\|v\right\|_2 \le t \le C\left\|v\right\|_2$ to be the smallest k such that $t^{(k)} := \frac{C}{2^k}\left\|v\right\|_2 \le t$, so $K(C\left\|v\right\|_2) = 0$, $K(t) = 1$ for $t \in \left[\frac{C}{2}\left\|v\right\|_2, C\left\|v\right\|_2\right)$, and so on. By construction, for all relevant t,

$$t^{(K(t))} \le t < 2t^{(K(t))}.$$

Since G is decreasing in its argument, we can write

$$\int_0^{C\|v\|_2} 2tG(t)dt \le \int_0^{\|v\|_2} 2tdt + \int_{\|v\|_2}^{C\|v\|_2} 2tG\left(t^{(K(t))}\right)dt \le \|v\|_2^2 + \int_{\|v\|_2}^{C\|v\|_2} 2tG\left(t^{(K(t))}\right)dt.$$

Now, for all $0 \le k \le \log_2(\frac{80}{\beta^2\alpha})$, we recall that we assumed both calls to SplitOrTailBound with thresholds $\tau_{\mathrm{med}} \pm \frac{C}{2^k}\left\|v\right\|_2$ failed to produce a split, and hence certify a tail bound (8.14). Thus,

$$G\left(t^{(k)}\right) \le \frac{512\gamma}{\beta^2\left(t^{(k)}\right)^2}\left\|v\right\|_2^2 \le \frac{2048\gamma}{\beta^2 t^2}\left\|v\right\|_2^2, \text{ for any } t \text{ with } K(t) = k.$$

Here, the first inequality used both tail bounds in (8.14) and accounted for the fact that the quantizations ρ^+, ρ^- are defined over T_{in}, and G is defined over T_{mid} with $|T_{\mathrm{mid}}| \ge \frac{1}{2}|T_{\mathrm{in}}|$; the second

inequality used that for any such t, $t < 2t^{(k)}$. Putting all these pieces together,

$$\mathbb{E}[Z^2] \le \|v\|_2^2 + \int_{\|v\|_2}^{C\|v\|_2} \frac{4096\gamma}{\beta^2 t} \|v\|_2^2 \, dt \le \|v\|_2^2 + \frac{4096\gamma}{\beta^2} \log(C) \|v\|_2^2 = O\left(\frac{\gamma}{\beta^2} \cdot \log\left(\frac{1}{\alpha\beta}\right)\right) \|v\|_2^2 .$$

Finally, recall (8.15) shows that $\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}), vv^\top \right\rangle \le \mathbb{E}[Z^2]$. Thus, the set T_{mid} should have passed the check in Line 6 under the assumed lower bound on R, yielding the desired contradiction. □

8.4.3 Implementation of SplitOrTailBound

In this section, we prove Lemma 117 by providing SplitOrTailBound and giving its analysis.

Lemma 117. *There is an algorithm,* SplitOrTailBound *(Algorithm 36), which takes as input $T_{\mathrm{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\ge 0}$, and $\tau_0 \in \mathbb{R}$, and returns in one of two cases we call the "split" case and the "tail bound" case. In the split case, it returns (τ, r) such that the induced sets* (8.11) *satisfy* (8.12)*, and if $\frac{1}{n}\mathbf{1}_{T_{\mathrm{in}}}$ is γ-saturated then one of the induced sets T_{out} has $\frac{1}{n}\mathbf{1}_{T_{\mathrm{out}}}$ is γ-saturated. Otherwise, for all $t \in \mathbb{R}$ define the upper and lower tail probabilities*

$$\rho^+(t) := \Pr_{i \sim_{\mathrm{unif}} T_{\mathrm{in}}}[Y_i \ge t] \,, \; \rho^-(t) := \Pr_{i \sim_{\mathrm{unif}} T_{\mathrm{in}}}[Y_i \le t] \,. \tag{8.13}$$

Then, in the tail bound case SplitOrTailBound *certifies*

$$\rho^+(\tau_0) \le \frac{128\gamma}{\beta^2 \left|\tau_0 - \tau_{\mathrm{med}}\right|^2} \|v\|_2^2 \;\; \text{if } \tau_0 \ge \tau_{\mathrm{med}}, \;\; \text{and } \rho^-(\tau_0) \le \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\mathrm{med}}|^2} \|v\|_2^2 \;\; \text{if } \tau_0 \le \tau_{\mathrm{med}}. \tag{8.14}$$

Proof. This proof proceeds in two parts which we show separately. First, we demonstrate that if $\tau_0 \ge \tau_{\mathrm{med}}$ and none of the runs of Lines 9-17 in Algorithm 36 return with a valid split, then indeed we can certify the tail bound (8.14) holds (and a similar guarantee holds for $\tau_0 < \tau_{\mathrm{med}}$). Second, we show whenever a split is returned and T_{in} is γ-saturated, then one of the output sets will be as well.

Correctness of tail bound. We consider the case $\tau_0 \ge \tau_{\mathrm{med}}$ here, as the other case follows symmetrically. Let $K+1$ be the first index such that $\tau_{K+1} < \tau_{\mathrm{med}}$, so the algorithm checks all pairs $(\tau_j - r_j \|v\|_2, r_j)$ for $0 \le j \le K$. Suppose that all such induced splits fail to satisfy (8.12). For all $0 \le j \le K$, let $T_j^{(1)}$, $T_j^{(2)}$ be the induced sets by the pair $(\tau_j - r_j \|v\|_2, r_j)$. Then by construction

$$\frac{\operatorname{abs} T_j^{(1)}}{\operatorname{abs} T_{\mathrm{in}}} = 1 - g_j, \;\; \frac{\operatorname{abs} T_j^{(2)}}{\operatorname{abs} T_{\mathrm{in}}} = g_{j+1}.$$

For all $0 \le j \le K-1$, recall $(\tau_j - r_j \|v\|_2, r_j)$ did not pass the check (8.12), but the second expression reads $g_j \ge \frac{2\gamma}{r_j^2}$ (since $g_j \le \frac{1}{2} \le 1 - g_{j+1}$), which is true by construction. Thus the first check did not pass and we conclude

$$g_{j+1}^{1+\beta} + (1 - g_j)^{1+\beta} > 1 \implies g_{j+1}^{1+\beta} > g_j. \tag{8.16}$$

Algorithm 36: SplitOrTailBound$(T_{\text{in}}, v, \beta, \tau_0)$

Input: $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0,1]$, $\tau_0 \in \mathbb{R}$;

Output: Either outputs (τ, r) such that $T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i\rangle \le \tau + r\|v\|_2\}, T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i\rangle \ge \tau - r\|v\|_2\}$ satisfy

$$\left|T_{\text{out}}^{(1)}\right|^{1+\beta} + \left|T_{\text{out}}^{(2)}\right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}, \ \min\left(1 - \frac{\operatorname{abs} T_{\text{out}}^{(1)}}{\operatorname{abs} T_{\text{in}}}, 1 - \frac{\operatorname{abs} T_{\text{out}}^{(2)}}{\operatorname{abs} T_{\text{in}}}\right) \ge \frac{2\gamma}{r^2},$$

or returns "Tail bound" guaranteeing that for $\tau_{\text{med}} := \operatorname{med}(\{\langle v, X_i\rangle \mid i \in T_{\text{in}}\})$ (following (8.14))

$$\rho^+(t) \le \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2}\|v\|_2^2 \text{ if } \tau_0 \ge \tau_{\text{med}}, \text{ and } \rho^-(t) \le \frac{128\gamma}{\beta^2|\tau_0 - \tau_{\text{med}}|^2}\|v\|_2^2 \text{ if } \tau_0 \le \tau_{\text{med}}.$$

;

$Y_i \leftarrow \langle v, X_i\rangle$ for all $i \in T_{\text{in}}$, $\tau_{\text{med}} \leftarrow \operatorname{med}(\{Y_i \mid i \in T_{\text{in}}\})$;

$j \leftarrow 0$;

if $\tau_0 > \max_{i\in T_{\text{in}}} Y_i$ *or* $\tau_0 < \min_{i\in T_{\text{in}}} Y_i$ **then**

  "Tail bound";

if $\tau_0 \ge \tau_{\text{med}}$ **then**

  while $\tau_j \ge \tau_{\text{med}}$ **do**

    $g_j \leftarrow \rho^+(\tau_j)$ and $r_j \leftarrow \sqrt{\frac{2\gamma}{g_j}}$;

    if $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ *induced by* $(\tau_j - r_j\|v\|_2, r_j)$ *satisfy* (8.12) **then**

      Return: $(\tau_j - r_j\|v\|_2, r_j)$;

    else

      $\tau_{j+1} \leftarrow \tau_j - 2r_j$;

    $j \leftarrow j + 1$;

else

  while $\tau_j \le \tau_{\text{med}}$ **do**

    $\ell_j \leftarrow \rho^-(\tau_j)$ and $r_j \leftarrow \sqrt{\frac{2\gamma}{\ell_j}}$;

    if $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ *induced by* $(\tau_j + r_j\|v\|_2, r_j)$ *satisfy* (8.12) **then**

      Return: $(\tau_j + r_j\|v\|_2, r_j)$;

    else

      $\tau_{j+1} \leftarrow \tau_j + 2r_j$;

    $j \leftarrow j + 1$;

Return: "Tail bound";

By applying this inequality inductively with $j = K - 1$, and recalling $g_K \le \frac{1}{2}$, we have

$$\left(\frac{1}{2}\right)^{(1+\beta)^K} \ge g_K^{(1+\beta)^K} > g_0 \implies 2^{(1+\beta)^K} < \frac{1}{g_0} \implies K = O\left(\frac{1}{\beta}\log\log\left(\frac{1}{g_0}\right)\right). \tag{8.17}$$

Next, since $\tau_{K+1} = \tau_0 - 2\sum_{0\le j\le K} r_j \|v\|_2 < \tau_{\text{med}}$, we have

$$\tau_0 - \tau_{\text{med}} < 2\sum_{j=0}^{K} r_j \|v\|_2 = \sqrt{8\gamma}\|v\|_2 \sum_{j=0}^{K}\sqrt{\frac{1}{g_j}} < \sqrt{8\gamma}\|v\|_2 \sum_{j=0}^{K} A^{\frac{1}{(1+\beta)^j}}, \text{ for } A := \sqrt{\frac{1}{g_0}}.$$

where we used our earlier guarantee (8.16) inductively. By Lemma 120, we have the desired tail bound:

$$\tau_0 - \tau_{\text{med}} < \sqrt{8\gamma}\|v\|_2 \cdot \frac{4A}{\beta} \le \frac{\sqrt{128}}{\beta}\sqrt{\frac{\gamma}{g_0}} \implies \rho^+(\tau_0) = g_0 < \frac{128\gamma}{\beta^2 \operatorname{abs}\tau_0 - \tau_{\text{med}}{}^2}\|v\|_2^2.$$

Correctness of split. Suppose that we find (τ, r) such that for

$$T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i\rangle \le \tau + r\|v\|_2\},\ T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i\rangle \ge \tau - r\|v\|_2\},$$

we have

$$\min\left(1 - \frac{\left|T_{\text{out}}^{(1)}\right|}{|T_{\text{in}}|}, 1 - \frac{\left|T_{\text{out}}^{(2)}\right|}{|T_{\text{in}}|}\right) \ge \frac{2\gamma}{r^2}. \tag{8.18}$$

We show that the downweighting $\frac{1}{n}\mathbf{1}_{T_{\text{in}}} \to \frac{1}{n}\mathbf{1}_{T_{\text{out}}^{(i)}}$ is a weight removal with respect to γ-safe scores for one of $i = 1, 2$. Let $\tau^* := \langle v, \mu^*\rangle$ where μ^* is the "true mean vector" in Assumption 13. Clearly, either $\tau^* \ge \tau$ or $\tau^* \le \tau$; suppose without loss of generality $\tau^* \ge \tau$ as the other case follows symmetrically. Define the scores $\{s_i\}_{i\in T_{\text{in}}}$ to be 1 if $i \in T_{\text{in}} \setminus T_{\text{out}}^{(2)}$ and 0 otherwise; then the downweighting $\frac{1}{n}\mathbf{1}_{T_{\text{in}}} \to \frac{1}{n}\mathbf{1}_{T_{\text{out}}^{(2)}}$ is of the form in Lemma 112, with respect to these scores. Note that

$$\frac{1}{\gamma}\sum_{i\in T_{\text{in}}}\frac{1}{|T_{\text{in}}|}s_i = \frac{1}{\gamma}\left(1 - \frac{\operatorname{abs} T_{\text{out}}^{(2)}}{\operatorname{abs} T_{\text{in}}}\right) \ge \frac{2}{r^2}$$

by the assumption (8.18). To apply Lemma 112, it remains to show that

$$\sum_{i\in S\cap T_{\text{in}}}\frac{1}{\operatorname{abs} S\cap T_{\text{in}}}s_i \le \frac{2}{r^2}. \tag{8.19}$$

However, we can extend the definition of the scores $\{s_i\}_{i\in S}$ to include points in $S \setminus T_{\text{in}}$, so that s_i is the indicator function of $\langle v, X_i\rangle < \tau - r\|v\|_2$ for all $i \in S$, which is consistent with our definitions

$\{s_i\}_{i\in T_{\text{in}}}$. Then by Chebyshev's inequality and Assumption 13, using $\langle v, \mu^*\rangle \geq \tau$,

$$\sum_{i\in S} \frac{1}{|S|} s_i \leq \Pr_{i\sim_{\text{unif}} S}\left[\langle v, X_i - \mu^*\rangle^2 > r^2 \|v\|_2^2\right] \leq \frac{1}{r^2}. \tag{8.20}$$

By Lemma 114, if T_{in} is γ-saturated then $\operatorname{abs} S \cap T_{\text{in}} \geq \frac{1}{2}|S|$; combining with (8.20) yields (8.19). □

Lemma 120. *Let $A > \sqrt{2}$, $\beta \in (0, 1]$, and let K be such that $A^{\frac{1}{(1+\beta)^K}} > \sqrt{2}$. Then, we have*

$$\sum_{j=0}^{K} A^{\frac{1}{(1+\beta)^j}} \leq \frac{4A}{\beta}.$$

Proof. Define $f(x) = A^{\frac{1}{(1+\beta)^x}}$ for any $0 \leq x \leq K$, and note this is a decreasing function in x. Thus, since by direct computation the antiderivative of A^{B^x} is $\frac{1}{\log B}\mathrm{Ei}(B^x \log(A))$ where Ei is the exponential integral,

$$\begin{aligned}\sum_{j=0}^{K} A^{\frac{1}{(1+\beta)^j}} &\leq A + \int_0^K f(x)dx = A + \frac{1}{\log(1+\beta)}\left(\mathrm{Ei}\left(\log A\right) - \mathrm{Ei}\left(\frac{\log A}{(1+\beta)^K}\right)\right) \\ &\leq A + \frac{2}{\beta}\left(\mathrm{Ei}\left(\log A\right) + 1\right).\end{aligned}$$

In the last line, we used $\log(1+\beta) \geq \frac{\beta}{2}$ for $\beta \in (0, 1]$, $\frac{\log A}{(1+\beta)^K} > \log(\sqrt{2})$ by assumption, and Ei is increasing with $\mathrm{Ei}(\log(\sqrt{2})) > -1$. The conclusion follows from $\mathrm{Ei}(\log A) + 1 \leq \frac{3}{2}A$ for $A > \sqrt{2}$. □

8.4.4 Fixing a cluster via fast filtering

In this section, we prove Lemma 118 by providing Fixing and giving its analysis. Before stating the algorithm, we provide a helper result which analyzes the effect of a "randomized dropout scheme" with respect to safe scores, and shows with high probability it is still safe.

In other words, RandDrop removes points from T' with probability proportional to their score.

Lemma 121. *The output of* RandDrop *satisfies the guarantees given in Line 2 of Algorithm 37.*

Proof. For all $i \in T'$, let Z_i be the random variable defined as

$$Z_i = \begin{cases} 1 & \text{with probability } \frac{s_i}{s_{\max}} \\ 0 & \text{with probability } 1 - \frac{s_i}{s_{\max}} \end{cases}.$$

Note that the number of points removed from T' and $T' \cap S$ are respectively $\sum_{i\in T'} Z_i$ and $\sum_{i\in T'\cap S} Z_i$. We now obtain high-probability bounds on both of these totals.

Algorithm 37: $\mathsf{RandDrop}(T', \delta_{\mathrm{rd}}, s)$

Input: $T' \subseteq T$, $\delta_{\mathrm{rd}} \in (0,1)$, 4γ-safe scores $\{s_i\}_{i \in T'}$ with respect to $w := \frac{1}{n}\mathbf{1}_{T'}$ such that $s_{\max} := \max_{i \in T'} s_i \le 24|T' \cap S|$, and

$$\sum_{i \in T'} \frac{1}{|T'|} s_i \ge 288\gamma \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right). \tag{8.21}$$

;

Output: With failure probability $\le \delta_{\mathrm{rd}}$, outputs $T'' \subseteq T'$ such that if w is γ-saturated, then $\frac{1}{n}\mathbf{1}_{T''}$ is γ-saturated.;

$T'' \leftarrow \emptyset$;

for $i \in T'$ **do**

  $T'' \leftarrow T'' \cup \{X_i\}$ with probability $1 - \frac{s_i}{s_{\max}}$;

Return: T'';

First, we lower bound $\sum_{i \in T'} Z_i$. Observe that $\mathbb{E}\left[\sum_{i \in T'} Z_i\right] = \sum_{i \in T'} \frac{s_i}{s_{\max}}$, and each Z_i is Bernoulli. Thus we can apply a Chernoff bound to obtain

$$\Pr\left[\sum_{i \in T'} Z_i < \frac{1}{2}\sum_{i \in T'} \frac{s_i}{s_{\max}}\right] \le \exp\left(-\frac{1}{8}\sum_{i \in T'} \frac{s_i}{s_{\max}}\right) \le \frac{\delta_{\mathrm{rd}}}{2},$$

where we used $s_{\max} \le 24|T'|$ and the assumed lower bound (8.21) to conclude

$$\sum_{i \in T'} \frac{s_i}{s_{\max}} \ge \sum_{i \in T'} \frac{s_i}{24|T'|} \ge 8 \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right).$$

Next, we upper bound $\sum_{i \in T' \cap S} Z_i$. We claim with failure probability at most $\frac{\delta_{\mathrm{rd}}}{2}$,

$$\sum_{i \in T' \cap S} Z_i \le \frac{1}{2\gamma} \frac{\operatorname{abs} T' \cap S}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}}. \tag{8.22}$$

Define $\mu := \sum_{i \in T' \cap S} \frac{s_i}{s_{\max}}$ to be the expectation of the left hand side of (8.22), and set

$$\Delta := \frac{1}{\mu}\left(\frac{1}{2\gamma} \frac{\operatorname{abs} T' \cap S}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}}\right) - 1$$

so that $(1+\Delta)\mu$ is the right hand side of (8.22). Recall that we assumed that $\{s_i\}_{i \in T}$ were 4γ-safe; rearranging this definition (cf. Definition 29) yields

$$\mu \le \frac{1}{4\gamma} \cdot \frac{\operatorname{abs} T' \cap S}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}} \implies \Delta\mu = \frac{1}{2\gamma} \frac{\operatorname{abs} T' \cap S}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}} - \mu \ge \frac{1}{4\gamma} \cdot \frac{\operatorname{abs} T' \cap S}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}}.$$

However, since $\frac{\operatorname{abs} T' \cap S}{s_{\max}} \geq \frac{1}{24}$ by assumption, we use (8.21) and the above equation to conclude

$$\Delta\mu \geq 3 \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right).$$

Finally, a Chernoff bound shows the failure probability of (8.22) is at most $\exp\left(-\frac{\Delta\mu}{3}\right) \leq \frac{\delta_{\mathrm{rd}}}{2}$, as desired. Thus with probability at least $1 - \delta_{\mathrm{rd}}$,

$$\sum_{i \in T' \cap S} \frac{1}{|T' \cap S|} Z_i \leq \frac{1}{\gamma} \sum_{i \in T'} \frac{1}{|T'|} Z_i.$$

Now observe that the $\{Z_i\}_{i \in T'}$ meet the definition of γ-safe scores (Definition 29). Thus, Lemma 112 applies with weights $\frac{1}{n}\mathbf{1}_{T'}$ and $\frac{1}{n}\mathbf{1}_{T''}$ and we obtain the conclusion. □

We are now ready to state the algorithm Fixing and prove its guarantees in Lemma 118.

Lemma 118. *There is an algorithm,* Fixing *(Algorithm 38), which takes as input $T_{\mathrm{in}} \subseteq T$, $v \in \mathbb{R}^d$, and $R \in \mathbb{R}_{\geq 0}$ and produces T_{out} with the following guarantee with probability at least $1 - \delta$. Define T_{mid} as in Line 5 of Algorithm 35. Then if $\frac{1}{n}\mathbf{1}_{T_{\mathrm{in}}}$ is γ-saturated, so is $\frac{1}{n}\mathbf{1}_{T_{\mathrm{out}}}$, and if $\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{mid}}), vv^\top \right\rangle \leq \frac{1}{8}R^2 \|v\|_2^2$, then $\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{out}}), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2$.*

Proof. This proof proceeds in three parts. First, we show that whenever the average score is small:

$$\frac{1}{|T_{\mathrm{out}}|} \sum_{i \in T_{\mathrm{out}}} s_i \leq 288\gamma \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right) \|v\|_2^2,$$

then the check in Line 11 will fail and the algorithm will terminate. Next, we show calls to RandDrop meet its input criteria so its conclusion holds inductively (correctness of Line 10 is also handled here). Finally, we show that Fixing fails with probability at most δ. Assume throughout that $\frac{1}{n}\mathbf{1}_{T_{\mathrm{in}}}$ is γ-saturated; else there is nothing to prove. We also use the following notation throughout:

$$\mathrm{Var}_v(T') := \frac{1}{\operatorname{abs} T'} \sum_{i \in T'} \left(Y_i - \mu_v(T')\right)^2, \text{ where } \mu_v(T') := \left\langle v, \mu_{\frac{1}{n}\mathbf{1}}(T') \right\rangle, \text{ for all } T' \subseteq T. \quad (8.23)$$

Small average score implies termination. We show that whenever the average score is small: $\frac{1}{|T_{\mathrm{out}}|} \sum_{i \in T_{\mathrm{out}}} s_i \leq 288\gamma \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right) \|v\|_2^2$, we terminate since this implies

$$\left\langle \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\mathrm{out}}), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2.$$

Algorithm 38: $\mathsf{Fixing}(T_{\text{in}}, \alpha, v, \delta, R)$

Input: $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0,1)$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega\left(\sqrt{\gamma \log\left(\frac{\log d}{\delta}\right)}\right),$$

such that for T_{mid} defined in Algorithm 35, $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{mid}}), vv^\top \right\rangle \leq \frac{1}{8}R^2 \|v\|_2^2$;

Output: Outputs $T_{\text{out}} \subset T_{\text{in}}$ with

$$\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{out}}), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2.$$

If $\frac{1}{n}\mathbf{1}_{T_{\text{in}}}$ is γ-saturated, so is $\frac{1}{n}\mathbf{1}_{T_{\text{out}}}$, with failure probability $\leq \delta$.;

if $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{in}}), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2$ **then**

  Return: T_{in};

$Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$, $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$;

$I \leftarrow [\tau_{\text{med}} - c, \tau_{\text{med}} + c]$ is the smallest interval containing the $1 - \frac{\alpha}{4}$ quantiles of $\{Y_i \mid i \in T_{\text{in}}\}$ for $c \in \mathbb{R}_{\geq 0}$ and $2I \leftarrow [\tau_{\text{med}} - 2c, \tau_{\text{med}} + 2c]$;

Define scores $\{s_i\}_{i \in T_{\text{in}}}$ by

$$s_i \leftarrow \begin{cases} 0 & Y_i \in I \\ (Y_i - (\tau_{\text{med}} - c))^2 & Y_i \leq \tau_{\text{med}} - c \\ (Y_i - (\tau_{\text{med}} + c))^2 & Y_i \geq \tau_{\text{med}} + c \end{cases}$$

;

$\delta_{\text{rd}} \leftarrow \frac{\delta}{\Omega\left(\log d \cdot \log \frac{d}{\delta}\right)}$ for a sufficiently large constant;

$T_{\text{out}} \leftarrow T_{\text{in}} \setminus \{X_i \mid s_i \geq 12 \|v\|_2^2 |S|\}$;

while $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{out}}), vv^\top \right\rangle > \frac{1}{2}R^2 \|v\|_2^2$ **do**

  $T_{\text{out}} \leftarrow \mathsf{RandDrop}(T_{\text{out}}, \delta_{\text{rd}}, \frac{s}{\|v\|_2^2})$;

Return: T_{out};

To show this, we first prove

$$\begin{aligned} s_i &> \frac{1}{16}(Y_i - \mu_{2I})^2 \text{ for all } i \in T_{\text{out}}, Y_i \notin 2I, \\ \text{where } \mu_{2I} &:= \frac{1}{\text{abs}\, T_{\text{in}} \cap \{i \mid Y_i \in 2I\}} \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} Y_i. \end{aligned} \tag{8.24}$$

In other words, μ_{2I} is the mean of points in $2I$. To see this, if $Y_i = \tau_{\text{med}} - 2c - \Delta$ for $\Delta > 0$,

$$s_i = (c + \Delta)^2, \ (Y_i - \mu_{2I})^2 \leq (4c + \Delta)^2 < 16 s_i.$$

The case when $Y_i = \tau_{\text{med}} + 2c + \Delta$ is handled similarly, which covers all $Y_i \notin 2I$. Then, following notation (8.23),

$$\begin{aligned}
\text{Var}_v(T_{\text{out}}) &= \sum_{i \in T_{\text{out}}} \frac{1}{\text{abs}\, T_{\text{out}}} \left(Y_i - \mu_v(T_{\text{out}})\right)^2 \le \sum_{i \in T_{\text{out}}} \frac{1}{\text{abs}\, T_{\text{out}}} \left(Y_i - \mu_{2I}\right)^2 \\
&= \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} \left(Y_i - \mu_{2I}\right)^2 + \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \notin 2I\}} \frac{1}{|T_{\text{out}}|} \left(Y_i - \mu_{2I}\right)^2 \\
&\le \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} \left(Y_i - \mu_{2I}\right)^2 + 16 \sum_{i \in T_{\text{out}}} \frac{1}{|T_{\text{out}}|} s_i \\
&\le \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} \left(Y_i - \mu_{2I}\right)^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 .
\end{aligned}$$

Here, the first line used Fact 15, the third used (8.24) and that all scores are nonnegative, and the last used our assumption on the average score in T_{out}. Thus,

$$\begin{aligned}
\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{out}}\right), vv^\top \right\rangle &= \sum_{i \in T_{\text{out}}} \frac{1}{n} \left(Y_i - \mu_v(T_{\text{out}})\right)^2 \\
&\le \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{n} (Y_i - \mu_{2I})^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \\
&= \left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{out}} \cap \{i \mid Y_i \in 2I\}\right), vv^\top \right\rangle + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \\
&\le \frac{1}{8} R^2 \|v\|_2^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \le \frac{1}{2} R^2 \|v\|_2^2 .
\end{aligned}$$

In the second line we used $n \ge |T_{\text{out}}|$ to handle the second term, the third line used the definition of $\widetilde{\text{Cov}}$, and the fourth line used the assumed bound on $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{\text{mid}}), vv^\top \right\rangle$, and

$$\widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{out}} \cap \{i \mid Y_i \in 2I\}\right) \preceq \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}\left(T_{\text{mid}}\right)$$

since $T_{\text{out}} \cap \{i \mid Y_i \in 2I\} \subseteq T_{\text{mid}}$ and Fact 15 implies that dropping terms from the covariance formula and shifting to the mean only decreases Loewner order. The last line used the lower bound on R.

Correctness of calls to RandDrop. We first bound the average score in $T_{\text{in}} \cap S$ at the beginning of the algorithm. Let $\text{Var}_v(T_{\text{in}} \cap S)$ denote the variance of $T_{\text{in}} \cap S$ in the direction v following (8.23). We claim that the mean of $T_{\text{in}} \cap S$ lies close to I: in particular,

$$\mu_{\frac{1}{n}\mathbf{1}}(T_{\text{in}} \cap S) \in \left[\tau_{\text{med}} - c - \sqrt{2\text{Var}_v\left(T_{\text{in}} \cap S\right)}, \tau_{\text{med}} + c + \sqrt{2\text{Var}_v\left(T_{\text{in}} \cap S\right)}\right]. \tag{8.25}$$

If this were not the case, we would have a contradiction:

$$\begin{aligned}\mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right) &\geq \frac{1}{\operatorname{abs} T_{\mathrm{in}} \cap S} \sum_{i \in T_{\mathrm{in}} \cap S \mid Y_i \in I} \left(Y_i - \mu_v\left(T_{\mathrm{in}} \cap S\right)\right)^2 \\ &> \frac{\operatorname{abs} T_{\mathrm{in}} \cap S \cap \{i \mid Y_i \in I\}}{\operatorname{abs} T_{\mathrm{in}} \cap S} \left(2\mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right)\right) \geq \mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right).\end{aligned}$$

The second inequality used that every summand is at least $2\mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right)$ if (8.25) does not hold, and the last used that I contains a $1 - \frac{\alpha}{4}$ proportion of the points in T_{in}, and by Lemma 114 $T_{\mathrm{in}} \cap S$ contains at least $\frac{\alpha}{2}$ of the points in T_{in}. Now using (8.25) and the definition of the scores,

$$\begin{aligned}\sum_{i \in T_{\mathrm{in}} \cap S} \frac{1}{\operatorname{abs} T_{\mathrm{in}} \cap S} s_i &\leq \sum_{i \in T_{\mathrm{in}} \cap S} \frac{1}{\operatorname{abs} T_{\mathrm{in}} \cap S} \left(\operatorname{abs} Y_i - \mu_v\left(T_{\mathrm{in}} \cap S\right) + \sqrt{2\mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right)}\right)^2 \\ &\leq 6\mathrm{Var}_v\left(T_{\mathrm{in}} \cap S\right) \leq 12 \left\|v\right\|_2^2.\end{aligned} \tag{8.26}$$

In the last line, we used that $T_{\mathrm{in}} \cap S$ contains at least half the points in S by Lemma 114, so Assumption 13 applies with a normalizing factor at most twice as large. This shows that Line 10 of Algorithm 38 preserves saturation, since it can only remove points in $T_{\mathrm{in}} \setminus S$ (if any point in $T_{\mathrm{in}} \cup S$ had a score larger than $12|S| \left\|v\right\|_2^2$, it would violate (8.26)). This also shows that if at any point in running Algorithm 38 we have a γ-saturated subset $T_{\mathrm{out}} \subset T_{\mathrm{in}}$, then

$$\sum_{i \in T_{\mathrm{out}} \cap S} \frac{1}{\operatorname{abs} T_{\mathrm{out}} \cap S} s_i \leq 24 \left\|v\right\|_2^2. \tag{8.27}$$

This is because compared to (8.26), we can at most double the normalizing factor by Lemma 114, and all scores are nonnegative. By combining with the first part of this proof, whenever Line 11 passes,

$$\frac{1}{|T_{\mathrm{out}}|} \sum_{i \in T_{\mathrm{out}}} s_i > 288\gamma \log\left(\frac{2}{\delta_{\mathrm{rd}}}\right) \left\|v\right\|_2^2, \tag{8.28}$$

and hence the scores are 4γ-safe as required by RandDrop. The second requirement of RandDrop is that $s_{\max} \leq 24|T_{\mathrm{out}} \cap S| \left\|v\right\|_2^2$, which is taken care of by Line 10 as $|S| \leq 2|T_{\mathrm{out}} \cap S|$ by Lemma 114. Finally, Line 11 implies (8.28) by the first part of this proof, which is the third condition of RandDrop.

Bounding failure probability. We bound the failure probability in two steps. First, we show with probability at least $1 - \frac{\delta}{2}$, there are at most (for a suitable constant)

$$N := O\left(\log d \cdot \log \frac{d}{\delta}\right)$$

calls to RandDrop. Then, we union bound to show that all these calls to RandDrop pass with probability at least $1 - \frac{\delta}{2}$. Combining gives the overall failure probability to Fixing.

To see the bound on N, observe that after Line 10, the largest score is at most $O(\left\|v\right\|_2^2 d)$, and

the algorithm ends when the largest score is at most a constant (since then (8.28) clearly fails, at which point we terminate on Line 11 by the first part of this proof). Thus, the largest score can only halve at most $O(\log d)$ times. However, observing the implementation of RandDrop, any point with score at least half the largest is dropped with probability at least $\frac{1}{2}$, and hence after $O(\log \frac{d}{\delta})$ rounds, the largest score will halve with probability at least $1 - \frac{\delta}{\Omega(\log d)}$. Union bounding over all the phases of halving the max score implies after N loops the algorithm terminates with probability $1 - \frac{\delta}{2}$.

Since there are at most N calls to RandDrop, it suffices to set $\delta_{\mathrm{rd}} = \frac{\delta}{2N}$ to check that all calls to RandDrop pass with probability $1 - \frac{\delta}{2}$. If all calls pass, we have the desired conclusion. □

8.4.5 Runtime analysis

We now give a runtime bound for 1DPartition, and use it to obtain a similar bound on Partition.

Lemma 122. *Let $n' := |T'|$, where T' is the input to* 1DPartition*. Then* 1DPartition *can be implemented to run in time*

$$O\left(n'd + (n')^{1+\beta}\left(\frac{1}{\beta}\log\log d \cdot \log\left(\frac{1}{\alpha\beta}\right) + \log d \log\frac{d}{\delta}\right)\right).$$

Proof. We begin by computing all the points $Y_i := \langle v, X_i\rangle$ for $i \in T'$ and sorting them, and store all quantiles (i.e. the number of points less than any given Y_i), which takes time $O(n'd + n' \log n')$.

Next, we bound the cost of running Fixing on an input T_{in} of size n_{in}. Given access to quantile information, and since Fixing is only ever called on a set which is formed after applying some number of splits to the original dataset T', it is straightforward to implement Lines 3-10 in time $O(n_{\mathrm{in}})$. Moreover, each loop in Lines 11-13 costs $O(n_{\mathrm{in}})$ time, and by the proof of Lemma 118, there are at most $O(\log d \log \frac{d}{\delta})$ loops. Thus overall the runtime of Fixing is

$$O\left(n_{\mathrm{in}} \log d \log\frac{d}{\delta}\right).$$

We now consider the cost of running SplitOrTailBound with a given threshold τ_0. If τ_0 does not lie in the interval of $\{Y_i\}_{i\in T_{\mathrm{in}}}$, then the runtime is $O(1)$. Otherwise, consider the case $\tau_0 \geq \tau_{\mathrm{med}}$ (note τ_{med} can be computed in constant time given quantile information). Since at least one point is larger than τ_0, $g_0 \geq \frac{1}{n}$, and hence (8.17) shows the number of threshold checks is bounded by $O(\frac{1}{\beta}\log\log d)$. Each threshold check takes constant time (we just need to compute the cardinalities of the induced $T^{(1)}_{\mathrm{out}}, T^{(2)}_{\mathrm{out}}$) and computing the next g_j and r_j takes constant time given quantile information, so the cost of SplitOrTailBound is

$$O\left(\frac{1}{\beta}\log\log d\right).$$

Correspondingly, the cost of each run of Lines 8-11 of SplitOrCluster is bounded by

$$O\left(\frac{1}{\beta}\log\log d\cdot\log\left(\frac{1}{\alpha\beta}\right)\right).$$

Now, consider the structure of 1DPartition. Lemma 116 shows that there are at most $(n')^{1+\beta}$ split steps total, so the total cost of all split steps (which run Lines 8-11 of SplitOrCluster) is

$$O\left(\frac{(n')^{1+\beta}}{\beta}\log\log d\cdot\log\left(\frac{1}{\alpha\beta}\right)\right).$$

Finally, consider all nodes in the 1DPartition which are parents of leaves. The sums of cardinalities of all such nodes is bounded by $(n')^{1+\beta}$, so the cost of running Fixing on all these nodes is

$$O\left((n')^{1+\beta}\log d\log\frac{d}{\delta}\right).$$

□

As an immediate corollary, we obtain a runtime bound on Partition.

Corollary 29. *Let $n_p := |T_p|$ for some $T_p \subseteq T$.* Partition *called on input T_p with parameter C can be implemented to run in time*

$$O\left(n_p^{1+\beta}d\log d\log\frac{d}{\delta}+n_p^{1+\beta}\left(\frac{1}{\beta}\log\log d\cdot\log\left(\frac{1}{\alpha\beta}\right)\log\frac{d}{\delta}+\log d\log^2\frac{d}{\delta}\right)\right).$$

Proof. The proof is identical to Corollary 68, where we use Lemma 122 to bound the cost over all elements of each $\mathcal{S}_j$, and there are $N_{\mathrm{dir}} = \Theta(\log\frac{d}{\delta})$ calls to 1DPartition. □

8.4.6 Full bounded covariance algorithm

Finally, we give our full algorithm for list-decodable mean estimation under Assumption 13. As in Section G.5.3, we will reduce to the bounded diameter case via the algorithm NaiveCluster (cf. Lemma 311); we reproduce its guarantees for arbitrary failure probabilities as NaiveClusterPlus.

Lemma 123 (Restatement, Lemma 306)**.** *There is a randomized algorithm,* NaiveClusterPlus(T,δ)*, which takes as input $T \subset \mathbb{R}^d$ satisfying Assumption 13 and partitions it into disjoint subsets $\{T_i'\}_{i\in[k]}$ such that with probability at least $1-\delta$, all of S is contained in the same subset, and every subset has diameter bounded by $O(\frac{d^8}{\delta^2})$. The runtime of* NaiveClusterPlus *is $O(nd+n\log n)$.*

We also require a post-processing procedure to reduce the list size, which we call IteratePostProcess. We state its guarantees in Lemma 124, and defer the description and analysis to Section 8.4.7.

Lemma 124. *There is an algorithm,* IteratePostProcess *(Algorithm 41), which takes as input T satisfying Assumption 13 and a list $L \subset \mathbb{R}^d$ of length $m \le n$ such that*

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 \le \Delta,\ \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right)$$

and returns with probability at least $1-\delta$ a subset $L' \subset L$ of size $O(\frac{1}{\alpha})$ such that $\min_{\hat{\mu} \in L'} \|\hat{\mu} - \mu^*\|_2 = O(\Delta)$, within runtime

$$O\left(\left((m+n)d + \alpha m^2 n\right) \log \frac{d}{\delta}\right).$$

Algorithm 39: FastMultifilter$(T, \alpha, \delta, \beta)$

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumption 13 with parameter $\alpha \in (0, \frac{1}{2})$, $\delta \in (0,1)$, $\beta \in (0,1]$;

Output: With failure probability $\le \delta$: L with $|L| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L$ satisfies

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\sqrt{\frac{\log\left(\frac{1}{\alpha}\right)}{\alpha}} \cdot \max\left(\frac{1}{\beta}\sqrt{\log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log\log d}\right)\right). \tag{8.29}$$

;

$\delta_{\text{outer}} \leftarrow \frac{1}{2}$;

$N_{\text{runs}} \leftarrow \lceil 2 \log \frac{2}{\delta} \rceil$;

$L \leftarrow \emptyset$;

for $j \in [N_{runs}]$ **do**

  $\{T_i'\}_{i \in [k]} \leftarrow$ NaiveClusterPlus$(T, \frac{\delta_{\text{outer}}}{3})$;

  $\alpha_i \leftarrow \frac{|T|}{|T_i'|}\alpha$ for all $i \in [k]$;

  $L \leftarrow$

  $L \cup$ IteratePostProcess $\left(T, \bigcup_{i \in [k]}$ FastMultifilterBoundedDiameter$(T_i', \alpha_i, \frac{\delta_{\text{outer}}}{3}, \beta), \frac{\delta_{\text{outer}}}{3}\right)$;

Return: IteratePostProcess $\left(T, L, \frac{\delta}{2}\right)$;

Proposition 31. FastMultifilterBoundedDiameter *meets its output specifications with probability at least $1-\delta$, within runtime*

$$O\left(n^{1+\beta} d \log^2 d \log^2 \frac{d}{\delta} + n^{1+\beta}\left(\frac{1}{\beta}\log\log d \cdot \log\left(\frac{1}{\alpha\beta}\right) \log d \log^2 \frac{d}{\delta} + \log^2 d \log^3 \frac{d}{\delta}\right)\right).$$

Proof. The proof of the error rate is identical to that in Proposition 73, where the initial potential Φ_0 is bounded by $(\frac{d}{\delta})^{O(\log d)}$ via Lemma 123, which implies the operator norm of $\widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T')$ for every node T' on layer D is $O(R^2)$, and inductively at least one such node has $|T' \cap S| \ge \frac{1}{2}|S|$ by virtue of being γ-saturated and applying Lemma 114. The failure probability follows since Partition is called at most $n^{1+\beta}D$ times, as there are at most $n^{1+\beta}$ elements of each $L^{(\ell)}$. Finally, the list size

Algorithm 40: FastMultifilterBoundedDiameter$(T, \alpha, \delta, \beta)$

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumption 13 with parameter $\alpha \in (0, \frac{1}{2})$, $\delta \in (0,1)$, $\beta \in (0,1]$;

Output: With failure probability $\leq \delta$: L_{out} with $|L_{\text{out}}| = O(\frac{n^\beta}{\alpha})$ such that some $\hat{\mu} \in L_{\text{out}}$ satisfies

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\sqrt{\frac{\log\left(\frac{1}{\alpha}\right)}{\alpha}} \cdot \max\left(\frac{1}{\beta}\sqrt{\log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log\left(\frac{\log d}{\delta}\right)}\right)\right).$$

;

$L^{(0)} \leftarrow \{T\}$, $L_{\text{out}} \leftarrow \emptyset$;

For sufficiently large constants,

$$R \leftarrow \Theta\left(\max\left(\frac{1}{\beta} \cdot \sqrt{\log\left(\frac{1}{\alpha}\right)\log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log\left(\frac{1}{\alpha}\right)\log\left(\frac{d}{\delta}\right)}\right)\right),\ D \leftarrow \Theta\left(\log d \log\frac{d}{\delta}\right)$$

;

for $\ell \in [D]$ **do**

  $L^{(\ell)} \leftarrow \emptyset$;

  for $T' \in L^{(\ell-1)}$ **do**

    Append all elements of Partition$(T', \alpha, \frac{\delta}{n^{1+\beta}D}, \beta, R)$ to $L^{(\ell)}$;

Return: List of empirical means of all sets in $L^{(D)}$ with size at least $\frac{\alpha n}{2}$;

follows since Lemma 114 implies every leaf node contains $\frac{\alpha n}{2}$, but the total size across leaves is at most $n^{1+\beta}$.

Finally, to obtain the runtime bound we can sum the guarantee of Corollary 29 across each of the D layers, and use the potential to bound the sum of all $n_p^{1+\beta}$ across the layer. □

We are now ready to state our main claim on list-decodable mean estimation. For simplicity, we state the result for $\beta \geq \frac{1}{\log d}$, as otherwise there are no runtime or statistical gains asymptotically.

Theorem 51. *For $\frac{1}{\log d} \leq \beta \leq 1$, and $\log^{\Omega(1)}(d) \leq \alpha^{-1} \leq d$,* FastMultifilter *returns a list of size $O(\frac{1}{\alpha})$ such that*

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 = O\left(\frac{1}{\beta} \cdot \frac{\log\left(\frac{1}{\alpha}\right)}{\alpha}\right),$$

with probability at least $1 - \delta$, within runtime

$$O\left(n^{1+2\beta} d \log^4 d \log\frac{1}{\delta} + nd\log^2\frac{1}{\delta}\log\frac{d}{\delta}\right).$$

Proof. We first analyze Lines 6-10 of FastMultifilter. We claim that each of the N_{runs} times these

lines run, there is a $\geq \frac{1}{2}$ probability that some $\hat{\mu}$ will be added to L satisfying (G.2), within runtime

$$O\left(n^{1+2\beta}d\log^4 d + n^{1+\beta}\left(\frac{1}{\beta}\log\log d\cdot\log\left(\frac{1}{\alpha\beta}\right)\log^3 d + \log^5 d\right)\right) = O\left(n^{1+2\beta}d\log^4 d\right).$$

To see this, we apply Proposition 31 to the relevant call of FastMultifilterBoundedDiameter. The correctness follows identically to the proof of Theorem 91, except that the size of the list of candidate means $\bigcup_{i\in[k]}$ FastMultifilterBoundedDiameter$(T_i', \alpha_i, \frac{\delta_{\text{outer}}}{3}, \beta)$ is $m = O(\frac{n^\beta}{\alpha})$. By Lemma 124, after applying IteratePostProcess the error rate is not affected by more than a constant, and the list size is $O(\frac{1}{\alpha})$. The runtime of this last step is dominated by $O(\alpha m^2 n\log d) = O(n^{1+2\beta}d\log d)$.

Next, this implies that after all runs of Lines 6-10 have finished running (with independent internal randomness), there is a $\geq 1-\frac{\delta}{2}$ probability that L contains an element $\hat{\mu}$ satisfying (G.2). At this point, the size of the list is $m = O(\frac{\log\delta^{-1}}{\alpha})$, so Line 11 takes time $O(nd\log^2\frac{1}{\delta}\log\frac{d}{\delta})$ by Lemma 124. □

This theorem, combined with the previously discussed fact that we can assume that $\alpha \in [1/d, 1/\log^{\Omega(1)} d]$, gives our desired conclusion.

8.4.7 Cleaning up the list

In this section, we provide the subroutine IteratePostProcess used in FastMultifilter, and prove Lemma 124, which shows correctness of this subroutine. At a high level, IteratePostProcess first finds a greedy cover of the input list L at distance $O(\Delta)$. Then, while the greedy cover has size at least $4k$ for $k := \lceil\frac{1}{\alpha}\rceil$, it iteratively prunes away $2k$ out of $4k$ hypotheses by testing that there are enough datapoints closest to retained hypotheses; otherwise, it returns the greedy cover.

Lemma 124. *There is an algorithm,* IteratePostProcess *(Algorithm 41), which takes as input T satisfying Assumption 13 and a list $L \subset \mathbb{R}^d$ of length $m \leq n$ such that*

$$\min_{\hat{\mu}\in L}\|\hat{\mu}-\mu^*\|_2 \leq \Delta,\ \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right)$$

and returns with probability at least $1-\delta$ a subset $L' \subset L$ of size $O(\frac{1}{\alpha})$ such that $\min_{\hat{\mu}\in L'}\|\hat{\mu}-\mu^*\|_2 = O(\Delta)$, within runtime

$$O\left(\left((m+n)d + \alpha m^2 n\right)\log\frac{d}{\delta}\right).$$

Proof. We first prove correctness, and then prove the runtime bound.

Correctness. By the Johnson-Lindenstrauss lemma as analyzed in [6], with probability at least $1-\delta$ every pair of points in $L\cup T\cup\{\mu^*\}$ has their distance preserved to a 1.1 multiplicative factor under multiplication by $\mathbf{G}^\top$. Condition on this event for the remainder of the proof.

Let $\bar{\mu}$ be the element of the input L which is guaranteed to be within distance Δ of μ^*. We will first show that $\bar{\mu}$ is never removed from L by the loop in Lines 6-11. If $\bar{\mu}$ is not a part of L_{head} in a

Algorithm 41: IteratePostProcess$(T, \alpha, L, \delta, \Delta)$

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumption 13 with parameter $\alpha \in (0, \frac{1}{2})$, $\delta \in (0, 1)$, L with $|L| = m \leq n$ such that

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 \leq \Delta, \ \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right).$$

;

Output: With failure probability $\leq \delta$: $L' \subset L$ with $|L'| = O(\frac{1}{\alpha})$ such that

$$\min_{\hat{\mu} \in L'} \|\hat{\mu} - \mu^*\|_2 = O(\Delta).$$

;

$\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm\frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{d}{\delta})$ (Johnson-Lindenstrauss matrix [6]);

$k \leftarrow \lceil \frac{1}{\alpha} \rceil$;

$L' \leftarrow$ maximal subset of L such that $\forall \hat{\mu} \neq \hat{\mu}' \in L'$, $\left\|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\right\|_2 \geq 5\Delta$;

while $|L'| \geq 4k$ **do**

  $L_{\text{head}} \leftarrow$ first $4k$ elements of L';

  $L_{\text{prune}} \leftarrow$ elements of L_{head} which are nearest neighbors of $< \frac{\alpha n}{2}$ elements of T, where $\hat{\mu} \in L_{\text{head}}$ is the nearest neighbor of $X_i \in T$ if $\left\|\mathbf{G}^\top(\hat{\mu} - X_i)\right\|_2$ is minimal amongst L_{head} ;

  $L \leftarrow L \setminus L_{\text{prune}}$;

  $L' \leftarrow$ maximal subset of L such that $\forall \hat{\mu} \neq \hat{\mu}' \in L'$, $\left\|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\right\|_2 \geq 5\Delta$;

Return: L';

given loop, clearly this is true, so suppose $\bar{\mu} \in L_{\text{head}}$, and let $\hat{\mu}$ be some other element in L_{head} with $\left\|\mathbf{G}^\top(\bar{\mu} - \hat{\mu})\right\|_2 \geq 5\Delta$; by definition of L_{head} as a subset of L', all such $\hat{\mu}$ satisfy this. Our goal will be to show that at least half of the points in S have nearest neighbor $\bar{\mu}$; to do so, it suffices to show that $\left\|\mathbf{G}^\top(X_i - \hat{\mu}))\right\|_2 \geq \left\|\mathbf{G}^\top(X_i - \bar{\mu}))\right\|_2$ with probability at most $\frac{1}{8k}$ over $i \sim S$ for each $\hat{\mu} \neq \bar{\mu}$, so the $< 4k$ other hypotheses in L_{head} can only remove $\frac{\alpha n}{2}$ of the points in S from having nearest neighbor $\bar{\mu}$, and hence $\bar{\mu}$ will not be pruned.

We now show the key claim: that for all $\hat{\mu} \neq \bar{\mu} \in L_{\text{head}}$,

$$\Pr_{i \sim_{\text{unif}} S}\left[\left\|\mathbf{G}^\top\left(X_i - \hat{\mu}\right)\right\|_2 \leq \left\|\mathbf{G}^\top\left(X_i - \bar{\mu}\right)\right\|_2\right] \leq \frac{1}{8k}.$$

Observe that by the triangle inequality, for any $i \in S$ satisfying the event above,

$$\begin{aligned} 2\left\|\mathbf{G}^\top\left(X_i - \bar{\mu}\right)\right\|_2 &\geq \left\|\mathbf{G}^\top\left(X_i - \bar{\mu}\right)\right\|_2 + \left\|\mathbf{G}^\top\left(X_i - \hat{\mu}\right)\right\|_2 \\ &\geq \left\|\mathbf{G}^\top\left(\hat{\mu} - \bar{\mu}\right)\right\|_2 \geq 5\Delta \\ \implies \left\|X_i - \mu^*\right\|_2 \geq \left\|X_i - \bar{\mu}\right\|_2 - \left\|\bar{\mu} - \mu^*\right\|_2 &\geq \Delta. \end{aligned}$$

Here we used $\left\|X_i - \bar{\mu}\right\|_2 \geq \frac{1}{1.1}\left\|\mathbf{G}^\top(X_i - \bar{\mu})\right\|_2 \geq 2\Delta$, and $\left\|\bar{\mu} - \mu^*\right\|_2 \leq \Delta$ by assumption. By Chebyshev's inequality and Assumption 13, we conclude for sufficiently large $\Delta = \Omega(\frac{1}{\sqrt{\alpha}})$,

$$\Pr_{i \sim_{\text{unif}} S}\left[\left\|\mathbf{G}^\top\left(X_i - \hat{\mu}\right)\right\|_2 \leq \left\|\mathbf{G}^\top\left(X_i - \bar{\mu}\right)\right\|_2\right] \leq \Pr_{i \sim_{\text{unif}} S}[\left\|\bar{\mu} - \mu^*\right\|_2 \geq \Delta] \leq \frac{1}{8k}.$$

Finally, we have shown that when the algorithm exits on Line 12, L' is a maximal separated subset of a pruned list L containing $\bar{\mu}$. If $\bar{\mu} \in L'$, the guarantee is immediate; otherwise, there must have been some other $\hat{\mu} \in L'$ with $\left\|\mathbf{G}^\top(\hat{\mu} - \bar{\mu})\right\|_2 \leq 5\Delta$, else $\bar{\mu}$ would have been added. For this $\hat{\mu}$,

$$\left\|\hat{\mu} - \mu^*\right\|_2 \leq 1.1\left\|\mathbf{G}^\top(\hat{\mu} - \mu^*)\right\|_2 \leq 1.1\left\|\mathbf{G}^\top(\hat{\mu} - \bar{\mu})\right\|_2 + 1.1\left\|\mathbf{G}^\top(\bar{\mu} - \mu^*)\right\|_2 = O(\Delta).$$

The list size bound follows from Line 6, as the returned L' has at most $4k$ elements.

Runtime. First, the cost of computing all projections $\mathbf{G}^\top X$ for $X \in L \cup T$ is $O((m+n)d\log\frac{d}{\delta})$. Next, Lines 6-11 can only be looped over at most $O(\alpha m)$ times, since every loop removes $2k$ elements from L which originally has size m. It remains to argue about the complexity of each loop.

The cost of computing a maximal subset in Lines 5 and 10 is $O(m^2\log\frac{d}{\delta})$, since distance comparisons under multiplication by $\mathbf{G}$ take $O(\log\frac{d}{\delta})$ and it suffices to greedily loop over the list. Similarly, the cost of computing nearest neighbors of all elements in T in Line 8 is $O(mn\log\frac{d}{\delta})$, which is the dominant term. Combining these components yields the claim. □

8.4.8 (Slightly) improving the error rate

We give a brief discussion of how it is possible to shave a $\sqrt{\log \alpha^{-1}}$ factor from the error guarantees of Theorem 51, bringing it to within a $\sqrt{\log \alpha^{-1}}$ factor from optimal when β is a constant. At a high level, this extraneous factor is due to our insistence that all weight removals be γ-safe, for some $\gamma = \Theta(\log \alpha^{-1})$. This causes the thresholds required for termination of our subroutines (e.g. for SplitOrCluster to enter the Fixing stage) to be inflated by roughly a γ factor.

We can remove this factor by using 2-safe scores instead of $\Theta(\log \alpha^{-1})$-safe scores, an idea introduced by Section G.2 to obtain improved estimation rates over the multifilter of [179]. The idea is to restart the algorithm in *phases*, where each phase corresponds to the total maintained weight being stable up to a factor of 2 (in our case, this means subset sizes are stable up to factors of 2).

We now summarize the changes to our algorithm. We will run the "outer loop" subroutine FastMultifilterBoundedDiameter (which can be viewed as constructing a multifilter tree) up until a depth of $O(\log^2 d \log \frac{1}{\alpha})$ is reached, in batches of $O(\log^2 d)$ each corresponding to a stable phase. Each batch will either meet the relevant termination condition (bounded covariance, such that e.g. (8.10) is trivially satisfied), or make progress by entering the next phase via safe weight removals.

Correspondingly, the condition (8.6) required to make improvements on the potential will be scaled differently, according to the size of the relevant set T_p at some node p. In particular, suppose we are in a phase when $\frac{1}{2}n' < |T_p| \le n'$. Then we will aim to guarantee

$$\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \right\rangle \le R^2 \sqrt{\frac{n'}{n}} \mathrm{Tr}(\mathbf{Y}_p^2),$$

where R has the same value as in FastMultifilterBoundedDiameter up to removing a $\sqrt{\log \alpha^{-1}}$. This allows us to terminate when the operator norm of some (unnormalized) $\widetilde{\mathrm{Cov}}$ matrix is $O(R^2\sqrt{\frac{n'}{n}})$, at which point Lemma 113 concludes a distance of

$$O\left(\sqrt{R^2\sqrt{\frac{n}{n'}} \cdot \frac{n'}{|T_p \cap S|}}\right) = O\left(R \cdot \frac{\sqrt[4]{n'n}}{\sqrt{|T_p \cap S|}}\right) = O\left(\frac{R}{\sqrt{\alpha}}\right),$$

where we use that 2-saturation of T_p implies $\frac{|T_p \cap S|}{n} \ge \alpha\sqrt{\frac{n'}{2n}}$.

Because of the complications this type of argument introduces, e.g. every one of our subroutines needs an extra exit condition (when the maintained subset enters the next phase), we omit a formal treatment here. However, we remark that to remove the entire $\log \alpha^{-1}$ factor from our error likely requires new ideas. This is because both branches of our key subroutine SplitOrCluster, namely SplitOrTailBound and Fixing, require this overhead. The former is because integrating variance tail bounds decaying as $t \cdot \frac{1}{t^2}$ out to $O(\alpha)$ quantiles (cf. Lemma 119) introduces a gap of $\log(\frac{1}{\alpha})$. The latter is because we employ randomize dropout to maintain subsets (rather than weights); our dropout method requires a threshold of roughly $\log \log d$ (cf. Lemma 121) to obtain high-probability

guarantees after union bounding polylog(d) times. For $\alpha^{-1} = \log^{\Omega(1)} d$, this is again a $\log(\frac{1}{\alpha})$ gap.

8.5 Clustering mixture models

We define a *mixture model* to be a mixture $\sum_{i\in[k]} \alpha_i \operatorname{dist}_i$ where $\{\alpha_i\}_{i\in[k]} \in \mathbb{R}^k_{\geq 0}$, $\sum_{i\in[k]} \alpha_i = 1$, and all dist_i are supported on $\mathbb{R}^d$. In Sections 8.5.1 and 8.5.2 we handle the case where all distributions are sub-Gaussians: dist_i has mean μ_i, and sub-Gaussian parameter ≤ 1 in all directions (cf. Section 8.3.1). We begin with the uncorrupted, uniform mixture case as a warmup in Section 8.5.1, and show how our method tolerates non-uniformity and adversarial outliers in Section 8.5.2. We then give a simple extension of our algorithm to handle mixtures where each component has bounded fourth moment in Section 8.5.3, and finally tackle the case of bounded-covariance mixture models in Section 8.5.4.

Broadly, all of our clustering algorithms follow the same design framework. We first demonstrate using concentration and existence of a good hypothesis (the list-decodable learning guarantee), that the "nearest hypothesis" to every non-adversarial point is close to the true mean. We next prune our hypotheses down by only keeping those with a substantial number of nearby points; by arguing that the number of adversarial points (or points that appear adversarial due to anti-concentration) is small, no large "coalition" of bad points can be formed, and hence all kept hypotheses are near a true mean. Finally, assuming enough separation between true means, we can define a partition of the points based on their nearest hypotheses. In Section 8.5.4, we will use a more direct clustering process in the subspace spanned by candidates, combined with fast projected distance approximations, in order to obtain a tighter separation guarantee.

Throughout, we will frequently use that by Chernoff, the sum of any Bernoulli random variables whose expectation is $\Omega(d)$ will deviate from its expectation by at most any multiplicative constant with probability at least $1 - \exp(-\Omega(d))$. For example, for a dataset of size $n = \Omega(dk)$ drawn from a uniform mixture $\sum_{i\in[k]} \frac{1}{k} \operatorname{dist}_i$, each component dist_i will contribute between $0.99\frac{n}{k}$ and $1.01\frac{n}{k}$ points with probability at least $1 - k\exp(-\Omega(d))$, or more simply $1 - \exp(-\Omega(d))$ for $k = O(d)$.

8.5.1 Clustering uniform (sub-)Gaussian mixture models

We first consider the simple setting where all $\alpha_i = \frac{1}{k}$ and all dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions. We assume access to a *list-decoding* algorithm $\mathcal{A}$ which returns a list L of length $O(k)$, such that for each $i \in [k]$, L contains $\hat{\mu}_i$ such that $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$, for some $\Delta = \Omega(\sqrt{k})$ (in particular, FastMultifilter suffices for $\mathcal{A}$). Finally, we assume access to a dataset $\mathbf{X} = \{X_j\}_{j\in[n]}$ of size $n = \Theta(dk)$ drawn from the mixture model independently of $\mathcal{A}$, where we say that each X_j is "associated with" an index $i \in [k]$ (designating the component it is drawn from). We will now demonstrate how to cluster a dataset using calls to $\mathcal{A}$, assuming a sufficiently large separation between the means of any two mixture components.

Algorithm 42: ClusterUniformGMM($\mathbf{X}, L, \Delta, k, \delta$)

Input: $\mathbf{X} = \{X_j\}_{j\in[n]} \sim \sum_{i\in[k]} \frac{1}{k} \mathrm{dist}_i$ where dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and $n = \Theta(dk)$, L of size $O(k)$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \Omega(\sqrt{k})$, $\delta \in (0, 1)$;

$\mathbf{G} \in \mathbb{R}^{d\times c} \leftarrow$ entrywise $\pm\frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [6]);

Let $m : [n] \to L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\left\|\mathbf{G}^\top(X_j - \hat{\mu})\right\|_2$;

Define an equivalence relation $\sim$ on $\mathbf{X}$ by $X_i \sim X_j$ iff $\left\|\mathbf{G}^\top(m(i) - m(j))\right\|_2 \leq 18\Delta$; if this is not an equivalence relation, then return any labeling;

Return: Labeling of $\mathbf{X}$ associated with $\sim$;

We begin with the following observation.

Lemma 125. *Consider some $X_j \in \mathbf{X}$ associated with $i \in [k]$. With probability at least $1 - \exp(-\Omega(\Delta^2)))$, for every pair $\hat{\mu}, \hat{\mu}' \in L$, $\hat{\mu} \neq \hat{\mu}'$ letting $v_{\hat{\mu}\hat{\mu}'}$ be the unit vector in the direction $\hat{\mu} - \hat{\mu}'$,*

$$\langle v_{\hat{\mu}\hat{\mu}'}, X_j\rangle < \langle v_{\hat{\mu}\hat{\mu}'}, \mu_i\rangle + \Delta.$$

Proof. This is a standard application of sub-Gaussian concentration (on the one-dimensional distribution $\mathcal{N}(\langle v_{\hat{\mu}\hat{\mu}'}, \mu_i\rangle, \langle \mathbf{\Sigma}_i, v_{\hat{\mu}\hat{\mu}'} v_{\hat{\mu}\hat{\mu}'}^\top\rangle)$), where we union bound across $O(k^2)$ pairs of elements in L. We simplify by using $\Delta = \Omega(\sqrt{k})$, so the exponential term dominates the k^2 union bound overhead. □

Next, we give our key structural lemma regarding the map m.

Lemma 126. *Following notation of Algorithm 42, with probability at least $1 - \delta - n\exp(-\Omega(\Delta^2))$, every X_j associated with $i \in [k]$ satisfies $\|m(j) - \mu_i\|_2 \leq 7\Delta$.*

Proof. With probability at least $1-\delta$, all pairwise distances between $\mathbf{X}\cup L$ and itself are preserved by multiplication through $\mathbf{G}^\top$ up to a 1 ± 0.1 factor [6] (which we will call the "Johnson-Lindenstrauss guarantee" henceforth); condition on this event for the remainder of the proof. Suppose for contradiction that $\|m(j) - \mu_i\|_2 > 7\Delta$, and let $\hat{\mu}_i \in L$ denote any (fixed) hypothesis which is promised to satisfy $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$.[6] By the triangle inequality, $\|m(j) - \hat{\mu}_i\|_2 > 6\Delta$. Then letting v be the unit vector in the direction of $m(j) - \hat{\mu}_i$,

$$\langle v, \mu_i\rangle \leq \langle v, \hat{\mu}_i\rangle + \Delta,\ \langle v, m(j)\rangle > \langle v, \hat{\mu}_i\rangle + 6\Delta$$
$$\implies \langle v, \mu_i\rangle < \langle v, m(j)\rangle - 5\Delta.$$

[6] In the case multiple such hypotheses exist, any satisfactory (but fixed) $\{\hat{\mu}_i\}_{i\in[k]} \subseteq L$ will do.

Next, $\left\|\mathbf{G}^\top(X_j - m(j))\right\|_2 \le \left\|\mathbf{G}^\top(X_j - \hat{\mu}_i)\right\|_2$ implies $\|X_j - m(j)\|_2 \le 2\|X_j - \hat{\mu}_i\|_2$ by the Johnson-Lindenstrauss guarantee, or $\langle v, X_j\rangle \ge \left\langle v, \frac{2\hat{\mu}_i + m(j)}{3}\right\rangle$. Combining with the above displayed equation,

$$\langle v, X_j\rangle \ge \frac{2}{3}\langle v, \hat{\mu}_i\rangle + \frac{1}{3}\langle v, m(j)\rangle > \langle v, \mu_i\rangle + \Delta.$$

Applying Lemma 125 and union bounding over all $j \in [n]$ concludes the proof. □

This implies that with high probability, Algorithm 42 (given a list L meeting its prerequisites) succeeds in correctly labelling all data points, assuming $\Omega(\Delta)$ separation between component means.

Lemma 127. *Suppose every pair $i, i' \in [k]$, $i \ne i''$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 34\Delta$. Then with probability at least $1 - \delta - n\exp(-\Omega(\Delta^2))$, Algorithm 42 (assuming its preconditions) outputs a correct clustering of all points (up to label permutation).*

Proof. Assume the result of Lemma 126 and that multiplication through $\mathbf{G}^\top$ preserves all pairwise distances between $\mathbf{X} \cup L$ and itself up to a 1 ± 0.1 factor throughout this proof.

We first prove that for any two X_j, $X_{j'}$ associated to the same $i \in [k]$, Line 4 of Algorithm 42 sets $X_j \sim X_{j'}$. To see this, Lemma 126 and the triangle inequality give $\|m(j) - m(j')\|_2 \le 14\Delta$, so this will pass Line 4 by the Johnson-Lindenstrauss guarantee. Next, suppose X_j is associated with $i \in [k]$ and $X_{j'}$ is associated with $i' \in [k]$ with $i \ne i'$, and suppose for contradiction $X_j \sim X_{j'}$. By the Johnson-Lindenstrauss guarantee, $\|m(j) - m(j')\|_2 \le 20\Delta$, which yields by Lemma 126 and the triangle inequality that $\|\mu_i - \mu_{i'}\|_2 \le 34\Delta$, contradicting the separation assumption. □

We conclude with the following guarantee on ClusterUniformGMM.

Corollary 30. *Suppose every pair $i, i' \in [k]$, $i \ne i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{k}\log k)$ for an appropriate constant. There is an algorithm drawing $n = \Theta(dk)$ samples from the mixture $\sum_{i\in[k]} \frac{1}{k}\,\mathrm{dist}_i$ where dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and returns a correct clustering of all points (up to label permutation) with probability at least*

$$1 - \delta - n\exp\left(-\Omega\left(k\log^2 k\right)\right) - k\exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d\log^4 n \log^4\frac{n}{\delta} + k^2\log^4\frac{n}{\delta} + nk\log\frac{n}{\delta}\right).$$

Proof. We begin by stating the algorithm. We take $\frac{1}{10}$ of the dataset and run FastMultifilter[7] on it to produce L satisfying the prerequisites of Algorithm 42, and then cluster the remaining $\frac{9}{10}$ of

[7] If $\alpha^{-1} = \log^{o(1)} d$, we instead run the algorithm in Section G.4 to obtain the desired error guarantee, which fits within the runtime budget. Similarly, if $\alpha^{-1} = \Omega(d)$, we instead run the algorithm in Section G.1 which fits within the runtime budget by Proposition 9 of that paper.

the dataset using L via ClusterUniformGMM; then, we take a disjoint $\frac{1}{10}$ and cluster the remaining $\frac{9}{10}$ using ClusterUniformGMM. We then match labels based on which clusters overlap on at least $\frac{1}{2}$ of their points between these two runs. The runtime follows from Theorem 91, and the runtime of ClusterUniformGMM, which is clearly $O(nk \log \frac{n}{\delta})$ since Line 3 dominates, as Line 4 can be greedily implemented using distance comparisons between only L once the map m has been formed.

Next, for correctness, Theorem 51 and Proposition B.1 of [118] (which says Assumption 13 is met for both FastMultifilter runs with probability $\geq 1 - \exp(\Omega(d))$) imply both runs of FastMultifilter correctly return lists satisfying the precondition of Algorithm 42; here we note that the dataset partition ensures independence of lists used and datasets clustered. Then, Lemma 127 implies both clusterings are completely correct on $\frac{9}{10}$ of the data. The conclusion follows from standard binomial concentration, which implies that the $\frac{8}{10}$ of the data which was held-out contains at least $\frac{1}{2}$ the points associated with each $i \in [k]$ in the overall dataset, with probability at least $1 - \exp(-\Omega(d))$. □

We remark that for n which grows super-exponentially in k (such that the failure probability guarantee of Corollary 30 becomes vacuous), it is straightforward to obtain an appropriate high-probability guarantee for clustering all points by assuming that the minimum pairwise cluster separation scales as $\sqrt{\log n}$. A similar remark also applies to Corollary 33.

8.5.2 Robustly clustering (sub-)Gaussian mixture models

In this section, we generalize Corollary 30 to non-uniform corrupted mixture models. In particular, we consider an adversarially corrupted mixture model

$$\mathcal{M} = (1 - \epsilon) \sum_{i \in [k]} \alpha_i \operatorname*{dist}_i + \epsilon \operatorname*{dist}_{\text{adv}}, \tag{8.30}$$

where for all $i \in [k]$, dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions. Moreover, for some fixed known α, we assume all $\alpha_i \geq \alpha$ and $\epsilon \leq \frac{\alpha}{4}$. By definition of α, note that we must have $\alpha = O(\frac{1}{k})$. In this section, we assume our list-decoding subroutine $\mathcal{A}$ returns a list of size $O(\alpha^{-1})$, and guarantees estimation error Δ. We now state our algorithm.

We again refer to X_j as associated with some $i \in [k]$ if it is a draw from dist_i, and as "adversarial" if it is drawn from dist_{adv}. We begin with the following consequences of Lemma 126.

Corollary 31. *With probability at least $1 - \delta - n\exp(-\Omega(\Delta^2)) - k\exp(-\Omega(d))$, both of the following events hold. For all $i \in [k]$, every $\hat{\mu} \in L$ with $\|\hat{\mu} - \mu_i\|_2 \leq 7\Delta$ is in L'. Moreover, every $\hat{\mu} \in L'$ has $\|\hat{\mu} - \mu_i\|_2 \leq 25\Delta$ for some $i \in [k]$.*

Proof. By standard binomial concentration, for every $i \in [k]$, the set of $j \in [n]$ associated with i has size at least $0.9\alpha n$ with probability $1 - k\exp(-\Omega(d))$. Condition on this event, all pairwise distances between $\mathbf{X} \cup L$ and itself being preserved up to 1 ± 0.1 by multiplication through $\mathbf{G}^\top$, and the conclusion of Lemma 125 for the remainder of this proof.

Algorithm 43: ClusterRobustGMM$(\mathbf{X}, L, \Delta, k, \delta, \alpha)$

Input: $\mathbf{X} = \{X_j\}_{j\in[n]} \sim (1-\epsilon)\sum_{i\in[k]} \alpha_i \operatorname{dist}_i + \epsilon \operatorname{dist}_{\text{adv}}$ where dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, all $\alpha_i \ge \alpha$, $\epsilon \le \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \le \Delta$ for $\Delta = \Omega(\sqrt{\alpha^{-1}})$, $\delta \in (0,1)$;

$\mathbf{G} \in \mathbb{R}^{d\times c} \leftarrow$ entrywise $\pm\frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log\frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [6]);

Let $m : [n] \to L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\left\|\mathbf{G}^\top(X_j - \hat{\mu})\right\|_2$;

$\mathcal{S}_{\hat{\mu}} \leftarrow \{j \in [n] \mid m(j) = \hat{\mu}\}$ for all $\hat{\mu} \in L$, $\mathcal{B}_{\hat{\mu}} \leftarrow \bigcup_{\hat{\mu}'\in L \mid \|\mathbf{G}^\top(\hat{\mu}-\hat{\mu}')\|_2 \le 16\Delta} \mathcal{S}_{\hat{\mu}'}$;

$L' \leftarrow \{\hat{\mu} \in L \mid |\mathcal{B}_{\hat{\mu}}| \ge 0.9\alpha n\}$;

Define an equivalence relation $\sim$ on $\mathbf{X}'$ by $X_i \sim X_j$ iff $\left\|\mathbf{G}^\top(m(i) - m(j))\right\|_2 \le 55\Delta$, for $\mathbf{X}' := \{X_i \in \mathbf{X} \mid m(i) \in L'\}$; if this is not an equivalence relation, then return any labeling;

Return: Labeling of $\mathbf{X}'$ associated with $\sim$, along with $\mathbf{X} \setminus \mathbf{X}'$ as "unlabeled";

We begin with the first claim: let $\|\hat{\mu} - \mu_i\|_2 \le 7\Delta$. For every X_j associated with i, Lemma 126 shows $\|m(j) - \mu_i\|_2 \le 7\Delta$, implying by the Johnson-Lindenstrauss guarantee and triangle inequality, $\left\|\mathbf{G}^\top(m(j) - \hat{\mu})\right\|_2 \le 16\Delta$. Hence X_j counts towards $\mathcal{B}_{\hat{\mu}}$, which then captures all points associated with i, so $\hat{\mu} \in L'$. For the second claim, suppose $\|\hat{\mu} - \mu_i\|_2 > 25\Delta$; then, no $\hat{\mu}' \in L$ with $\|\hat{\mu}' - \mu_i\|_2 \le 7\Delta$ will count towards $\mathcal{B}_{\hat{\mu}}$, since such $\hat{\mu}'$ has $\left\|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\right\|_2 > 16\Delta$. By Lemma 126, the only points that can contribute to $\mathcal{B}_{\hat{\mu}}$ are then the ones drawn from $\mathcal{D}_{\text{adv}}$, which by standard binomial concentration will be less than $0.6\alpha n$ with probability $1 - \exp(-\Omega(d))$, and hence $\hat{\mu} \notin L'$. □

The following conclusion then follows immediately in the vein of Lemma 127.

Corollary 32. *Suppose every pair $i, i' \in [k]$, $i \ne i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 55\Delta$. Then with probability at least $1 - \delta - n\exp(-\Omega(\Delta^2)) - k\exp(-\Omega(d))$, Algorithm 43 (assuming its preconditions) outputs a correct clustering of all points associated with some component $i \in [k]$ (up to label permutation).*

Proof. The proof is identical to Lemma 127, where we note for each $\hat{\mu} \in L'$, there is a unique $i \in [k]$ promised by Corollary 31 where $\|\hat{\mu} - \mu_i\|_2 \le 25\Delta$ (by our separation assumption). Thus, $\sim$ defines an equivalence relation, since every $\hat{\mu} \in L'$ is mapped to the unique equivalence class associated with $\hat{\mu}_i$. This successfully recovers the true clusters, since every point X_j associated with some $i \in [k]$ will be in $\mathbf{X}'$ by Corollary 31 and Lemma 126, and thus it will be classified correctly by $\sim$. □

Finally, we give a complete guarantee on ClusterRobustGMM.

Corollary 33. *Suppose every pair $i, i' \in [k]$, $i \ne i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}}\log\alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1-\epsilon)\sum_{i\in[k]} \alpha_i \operatorname{dist}_i + \epsilon \operatorname{dist}_{\text{adv}}$ where dist_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and returns a correct clustering of all points drawn from some dist_i (up to label permutation) with probability at least*

$$1 - \delta - n\exp\left(-\Omega\left(\alpha^{-1}\log^2\alpha^{-1}\right)\right) - k\exp\left(-\Omega(d)\right).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. The proof is the same as Corollary 30, where we note that the mislabeled points from $\mathrm{dist}_{\mathrm{adv}}$ can only affect the overlapping labels between the two runs by at most a $1.1\epsilon \leq \frac{1.1}{4}\alpha$ fraction, a minority compared to the number of correct labels (due to the true mixture), which in both runs will contribute at least $\frac{4}{5}\alpha_i$ of the points assigned by $\sim$ to the true clustering. Hence, between runs the clusters assigned by $\sim$ to the points drawn from dist_i will overlap on at least half their points, and we can use this agreement to correctly label all points not drawn from $\mathrm{dist}_{\mathrm{adv}}$ (where throughout, we conditioned on all high-probability events in Corollary 30's proof holding). □

8.5.3 Mixture models with bounded fourth moments

In this section, we consider non-uniform corrupted mixture models under a weaker distributional assumption based on bounded fourth moments. We will no longer be able to correctly cluster all (non-adversarial) points, but will instead guarantee that amongst non-adversarial points, at least a $1-o(\alpha)$ fraction are correctly classified. Concretely, we consider the adversarially corrupted mixture model (8.30), where for all $i \in [k]$, and some constant $C = O(1)$,

$$\mathbb{E}_{X \sim \mathrm{dist}_i}\left[\langle v, X - \mu_i\rangle^4\right] \leq C \text{ for all } v \in \mathbb{R}^d,\ \|v\|_2 = 1. \tag{8.31}$$

We remark that this fourth-moment bound also implies the covariance is bounded by $O(1)\mathbf{I}$ for all components, by Jensen's inequality. We again assume that all $\alpha_i \geq \alpha$, and $\epsilon \leq \frac{\alpha}{4}$. Our algorithm for this setting will be exactly the same as ClusterRobustGMM, but for convenience we restate it under the new distributional assumptions. We also assume $\Delta = \omega(\sqrt{\alpha^{-1}})$ for notational simplicity.

Algorithm 44: ClusterRobustBFMM$(\mathbf{X}, L, \Delta, k, \delta, \alpha)$

1 **Input:** $\mathbf{X} = \{X_j\}_{j\in[n]} \sim (1-\epsilon)\sum_{i\in[k]} \alpha_i \,\mathrm{dist}_i + \epsilon\,\mathrm{dist}_{\mathrm{adv}}$ where dist_i has mean μ_i and satisfies (8.31), all $\alpha_i \geq \alpha$, $\epsilon \leq \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \omega(\sqrt{\alpha^{-1}})$, $\delta \in (0,1)$;

2 $\mathbf{G} \in \mathbb{R}^{d\times c} \leftarrow$ entrywise $\pm\frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [6]);

3 Let $m : [n] \to L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\left\|\mathbf{G}^\top(X_j - \hat{\mu})\right\|_2$;

4 $\mathcal{S}_{\hat{\mu}} \leftarrow \{j \in [n] \mid m(j) = \hat{\mu}\}$ for all $\hat{\mu} \in L$, $\mathcal{B}_{\hat{\mu}} \leftarrow \bigcup_{\hat{\mu}' \in L \mid \|\mathbf{G}^\top(\hat{\mu}-\hat{\mu}')\|_2 \leq 15\Delta} \mathcal{S}_{\hat{\mu}'}$;

5 $L' \leftarrow \{\hat{\mu} \in L \mid |\mathcal{B}_{\hat{\mu}}| \geq 0.9\alpha n\}$;

6 Define an equivalence relation $\sim$ on $\mathbf{X}'$ by $X_i \sim X_j$ iff $\left\|\mathbf{G}^\top(m(i) - m(j))\right\|_2 \leq 55\Delta$, for $\mathbf{X}' := \{X_i \in \mathbf{X} \mid m(i) \in L'\}$; if this is not an equivalence relation, then return any labeling;

7 **Return:** Labeling of $\mathbf{X}'$ associated with $\sim$, along with $\mathbf{X} \setminus \mathbf{X}'$ as "unlabeled";

Our analysis of ClusterRobustBFMM follows from the following key observation: define $X_j \in \mathbf{X}$ as "adversarial" if it is drawn from dist_{adv}, and say it is "pseudo-adversarial" if it is drawn from some component $i \in [k]$, but satisfies $\|m(j) - \mu_i\|_2 > 7\Delta$. Then, by the bounded fourth moment assumption (8.31), there are few pseudo-adversarial points, made rigorous as follows.

Lemma 128. *With probability at least $1 - \delta - \exp(-\Omega(d))$, the number of pseudo-adversarial points in $\mathbf{X}$ is $o(\alpha n)$.*

Proof. We first bound the probability that some $X_j \sim \text{dist}_i$ will be pseudo-adversarial. By the proof of Lemma 126, if X_j were pseudo-adversarial, it must be the case that in the direction v corresponding to $m(j) - \hat{\mu}_i$, $\langle v, X_j \rangle > \langle v, \mu_i \rangle + \Delta$. However, by (8.31) and Markov,

$$\Pr_{X \sim \text{dist}_i} \left[\langle v, X - \mu_i \rangle^4 \geq \Delta^4 \right] \leq \frac{C}{\Delta^4} = o\left(\alpha^2\right).$$

Union bounding over all possible directions v (one for each of the $O(\alpha^{-1})$ elements of L other than $\hat{\mu}_i$), this implies the chance that X drawn from *any* dist_i is pseudo-adversarial is $o(\alpha)$. Hence, the expected number of pseudo-adversarial points in our entire dataset is $o(\alpha n)$, and the conclusion follows by applying standard binomial concentration. □

Now, Corollary 31 holds for ClusterRobustBFMM simply by lumping together the pseudo-adversarial points and the adversarial points in its proof (in particular, no hypothesis more than 25Δ away from a true mean can capture any points in the dataset other than adversarial and pseudo-adversarial ones). We now state analogs of Corollaries 32 and 33.

Corollary 34. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 55\Delta$. Then with probability at least $1 - \delta - n\exp(-\Omega(\Delta^2)) - k\exp(-\Omega(d))$, Algorithm 44 (assuming its preconditions) outputs a correct clustering of a $1 - o(\alpha)$ proportion of points associated with some component $i \in [k]$ (up to label permutation).*

Proof. The proof is identical to Corollary 32, where we only guarantee correctly labeling points which are not pseudo-adversarial; $\sim$ will correctly cluster all such points by separation and Corollary 31. □

Corollary 35. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1 - \epsilon) \sum_{i \in [k]} \alpha_i \, \text{dist}_i + \epsilon \, \text{dist}_{\text{adv}}$ where dist_i has mean μ_i and satisfies the bounded fourth moment condition (8.31), and returns a correct clustering of a $1 - o(\alpha)$ fraction of all points drawn from some dist_i (up to label permutation) with probability at least*

$$1 - \delta - k\exp\left(-\Omega(d)\right).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. The proof is identical to Corollary 33, using Corollary 34 instead of Corollary 32, and using that the pseudo-adversarial points will not substantially affect cluster overlap guarantees. □

8.5.4 Bounded-covariance mixture models

In this section, we finally handle the case of non-uniform corrupted mixture models under only a bounded-covariance assumption; in particular, we study (8.30) where every component dist_i for $i \in [k]$ has covariance bounded by $\mathbf{I}$. We again assume that all $\alpha_i \geq \alpha$, and $\epsilon \leq \frac{\alpha}{4}$.

Algorithm 45: ClusterRobustBCMM$(\mathbf{X}, L, \Delta, k, \delta, \alpha)$

1 **Input:** $\mathbf{X} = \{X_j\}_{j\in[n]} \sim (1-\epsilon)\sum_{i\in[k]} \alpha_i \,\mathrm{dist}_i + \epsilon\,\mathrm{dist}_{\mathrm{adv}}$ where dist_i has mean μ_i and has covariance bounded by $\mathbf{I}$, all $\alpha_i \geq \alpha$, $\epsilon \leq \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$, $\delta \in (0,1)$;

2 $\mathbf{G} \in \mathbb{R}^{d\times c} \leftarrow$ entrywise $\pm\frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [6]);

3 $\mathbf{P} \leftarrow \mathbf{L}^\top(\mathbf{L}\mathbf{L}^\top)^{-1}\mathbf{L}$ (i.e. projection matrix onto the subspace spanned by L) where $\mathbf{L} \in \mathbb{R}^{O(\alpha^{-1})\times d}$ is the vertical concatenation of L;

4 $\widetilde{X}_j \leftarrow \mathbf{G}^\top\mathbf{P}X_j$ for all $j \in [n]$, $\tilde{\mu}_i \leftarrow \mathbf{G}^\top\hat{\mu}_i$ for all $\hat{\mu}_i \in L$;

5 $\widetilde{L} \leftarrow \left\{\hat{\mu}_i \in L \,\middle|\, |\mathcal{S}(\hat{\mu}_i)| \geq \frac{\alpha n}{2}\right\}$, where $\mathcal{S}(\hat{\mu}_i) := \left\{X_j \in \mathbf{X} \,\middle|\, \left\|\widetilde{X}_j - \tilde{\mu}_i\right\|_2 \leq 2.5\Delta\right\}$;

6 Define an equivalence relation $\sim$ on $\widetilde{L}$ by $\hat{\mu}_i \sim \hat{\mu}_j$ iff $\|\tilde{\mu}_i - \tilde{\mu}_j\|_2 \leq 20\Delta$; if this is not an equivalence relation, then return any labeling;

7 **Return:** Labeling of $\mathbf{X}$ where $\widetilde{X}_j$ and $\widetilde{X}_{j'}$ have the same label iff $\widetilde{X}_j \in \mathcal{S}(\hat{\mu}_i)$ and $\widetilde{X}_{j'} \in \mathcal{S}(\hat{\mu}_{i'})$ for $\hat{\mu}_i \sim \hat{\mu}_{i'}$;

We briefly describe Algorithm 45. Line 3 forms the projection matrix onto the span of L. Line 5 "prunes" the set L to a set $\widetilde{L}$ where we only keep candidates with enough data points close by (in the subspace spanned by L). Line 6 then appropriately partitions the candidates, and Line 7 partitions the dataset based on the candidate partition. In the following proof we use $\mathbf{P}\hat{\mu} = \hat{\mu}$ for all $\hat{\mu} \in L$, and that all μ_i have a point in the subspace projected to by $\mathbf{P}$ at most Δ away.

Lemma 129. *With probability at least $1 - \delta - k\exp(-\Omega(d))$, if $\Delta = \Omega(\sqrt{\alpha^{-1}}\log\alpha^{-1})$, and every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 20\Delta$, Algorithm 45 returns a correct clustering of a $1 - o(1)$ fraction of all pointns drawn from some dist_i. The algorithm runs in time*

$$O\left(n \cdot \left(d + \frac{1}{\alpha}\right) \cdot \log \frac{n}{\delta}\right).$$

Proof. Throughout this proof, condition on the event that distances between $\mathbf{P}\mathbf{X} \cup L$ are preserved up to 1 ± 0.1 by multiplication through $\mathbf{G}^\top$ and that there are at most $\frac{\alpha n}{3}$ adversarial points; we will union bound this conditioning with an event of probability $1 - k\exp(-\Omega(d))$.

We first claim that any $\hat{\mu} \in L$ satisfying $\|\hat{\mu} - \mu_i\|_2 \le \Delta$ for some $i \in [k]$ is contained in $\widetilde{L}$, with probability at least $1 - \exp(-\Omega(d))$: call this claim †. To see †, with probability at least $1 - \exp(-\Omega(d))$, a $\frac{9}{10}$ fraction of $X_j \in \mathbf{X}$ sampled from dist_i satisfy $\|\mathbf{P}(X_j - \mu_i)\|_2 \le \Delta$. This follows since Chebyshev's inequality, $\Delta = \omega(\sqrt{\alpha^{-1}})$, and $\mathbf{P}$ projecting onto a $O(\alpha^{-1})$-dimensional subspace imply the probability $\|\mathbf{P}(X_j - \mu_i)\|_2 \le \Delta$ is $o(1)$. By triangle inequality and $\mathbf{P} \preceq \mathbf{I}$, for such X_j,

$$\|\mathbf{P}(X_j - \hat{\mu})\|_2 \le \|\mathbf{P}(X_j - \mu_i)\|_2 + \|\mathbf{P}(\hat{\mu} - \mu_i)\|_2 \le \Delta + \|\hat{\mu} - \mu_i\|_2 \le 2\Delta.$$

Since $\mathbf{G}^\top$ preserves distances to a 1.1 factor, this implies $\left\|\widetilde{X}_j - \tilde{\mu}\right\|_2 \le 2.5\Delta$ as desired.

We next claim that any $\hat{\mu} \in L$ satisfying $\|\hat{\mu} - \mu_i\|_2 > 7\Delta$ for all $i \in [k]$ will not be contained in $\widetilde{L}$, with probability at least $1 - \exp(-\Omega(d))$. To see this, we claim $\mathcal{S}(\hat{\mu})$ can only contain an $o(\alpha)$ fraction of the points drawn from dist_i; summing over all dist_i, and adding in all $\le \frac{\alpha n}{3}$ adversarial points, shows $|\mathcal{S}(\hat{\mu})| < \frac{\alpha n}{2}$. This latter claim follows by first observing that for $X_j \in \mathcal{S}(\hat{\mu})$,

$$\begin{aligned}\|\mathbf{P}(X_j - \mu_i)\|_2 &\ge \|\mu_i - \hat{\mu}\|_2 - \|\mu_i - \mathbf{P}\mu_i\|_2 - \|\mathbf{P}(X_j - \hat{\mu})\|_2 \\ &> 7\Delta - \Delta - 3\Delta = 3\Delta \ge \|\mathbf{P}(X_j - \hat{\mu})\|_2.\end{aligned}$$

The first inequality used that $\mathbf{P}\hat{\mu} = \hat{\mu}$, and the triangle inequality twice. In the second inequality, we used $\|\mu_i - \mathbf{P}\mu_i\|_2 \le \Delta$ since the subspace projected to by $\mathbf{P}$ contains a point at most Δ away from μ_i, and $\|\mathbf{P}(X_j - \hat{\mu})\|_2 \le 3\Delta$ by the assumption on $\mathbf{G}$ and $X_j \in \mathcal{S}(\hat{\mu})$. This implies that the projection of X_j is closer to $\mathbf{P}\hat{\mu} = \hat{\mu}$ than $\mathbf{P}\mu_i$. Letting dist_i' be the projection of dist_i by $\mathbf{P}$, $X \sim \mathrm{dist}_i'$ is closer to $\hat{\mu}$ than $\mathbf{P}\mu_i$ with probability at most $\frac{1}{\Omega(\Delta^2)} = o(\alpha)$ by arguments in Lemma 126 and Chebyshev. Hence, the probability that $X_j \in \mathcal{S}(\hat{\mu})$ is $o(\alpha)$, as desired.

Now, under the success of all above conditioning events, by the minimum separation assumption of 20Δ, each surviving $\hat{\mu} \in \widetilde{L}$ has a unique μ_i such that $\|\hat{\mu} - \mu_i\|_2 \le 7\Delta$, and each designated $\hat{\mu}_i$ with $\|\hat{\mu}_i - \mu_i\|_2 \le \Delta$ survives. Hence, the equivalence partition succeeds and captures all $\hat{\mu} \in \widetilde{L}$ at distance at most 7Δ from a μ_i. Moreover, for every pair $\hat{\mu}, \hat{\mu}' \in \widetilde{L}$ such that $\|\hat{\mu} - \mu_i\|_2 \le 7\Delta$ and $\|\hat{\mu}' - \mu_{i'}\|_2 \le 7\Delta$ for some $i \ne i'$,

$$X \in \mathcal{S}(\hat{\mu}) \cap \mathcal{S}(\hat{\mu}') \implies \|\hat{\mu} - \hat{\mu}'\|_2 \le 6\Delta.$$

Since the minimum separation between μ_i and $\mu_{i'}$ is 20Δ, this is a contradiction by the triangle inequality. So, no sets $\mathcal{S}(\hat{\mu})$ and $\mathcal{S}(\hat{\mu}')$ intersect, where the hypotheses are close to different means. Thus the labeling in Line 7 is well-defined. It remains to show that if $X_j \sim \mathrm{dist}_i$, we will have $X_j \in \mathcal{S}(\hat{\mu}_i)$ with probability $1 - o(1)$ (and hence a $1 - o(1)$ fraction of points is labeled correctly).

This was in fact shown earlier, when we proved the claim †.

Finally, it is clear that all operations can be performed in the desired runtime: $\mathbf{G}^\top \mathbf{P}$ can be computed explicitly in time $O(d^2\alpha^{-1} + d^2 \log \frac{n}{\delta}) = O(nd + d^2 \log \frac{n}{\delta})$ via naïve matrix multiplication, and all projection and distance computations (multiplying all points and hypotheses by $\mathbf{G}^\top \mathbf{P}$, computing all $\mathcal{S}(\hat{\mu})$, and checking all pairwise distances in Line 6) take time $O(n \cdot (d + \frac{1}{\alpha}) \cdot \log \frac{n}{\delta})$. □

Corollary 36. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1 - \epsilon) \sum_{i \in [k]} \alpha_i \operatorname{dist}_i + \epsilon \operatorname{dist}_{\text{adv}}$ where dist_i has mean μ_i and covariance bounded by $\mathbf{I}$, and returns a correct clustering of a $1 - \epsilon_1$ fraction of all points drawn from some dist_i (up to label permutation) for any fixed $\epsilon_1 > 0$, with probability at least*

$$1 - \delta - k \exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. Since the goal is to correctly label a $1 - \epsilon_1$ fraction of all points, we can use a $\frac{\epsilon_1}{2}$ fraction of our dataset as holdout for learning an independent list L. We then use this list to cluster a $1 - \frac{\epsilon_1}{2}$ fraction of the remaining points via Lemma 129. □

8.6 Robust regression

Parameter estimation in generalized linear models, such as linear and logistic regression problems, is among the most fundamental and well-studied statistical optimization problems. It serves as the primary workhorse in statistical studies arising from a variety of disciplines, ranging from economics [489], biology [524], and the social sciences [249]. Formally, given a *link function* $\gamma : \mathbb{R}^2 \to \mathbb{R}$ and a dataset of covariates and labels $\{(X_i, y_i)\}_{i \in [n]} \subset \mathbb{R}^d \times \mathbb{R}$ drawn from an underlying distribution $\mathcal{D}_{Xy}$, the problem of statistical (generalized linear) regression asks to

$$\text{estimate } \theta^\star := \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left\{ \mathbb{E}_{(X,y) \sim \mathcal{D}_{Xy}} \left[\gamma(\langle \theta, X \rangle, y) \right] \right\}. \tag{8.32}$$

For example, when $\gamma(v, y) = \frac{1}{2}(v - y)^2$, (8.32) corresponds to (statistical) linear regression. The problem (8.32) also has an interpretation as computing a maximum likelihood estimate for a parameterized distributional model for data generation, and indeed is only tractable under certain distributional assumptions, since we only have access to samples from $\mathcal{D}_{Xy}$ rather than the underlying distribution itself (see e.g. [53] for tractability results in the linear regression setting).

However, in modern settings, these strong distributional assumptions may fail to hold. In practically relevant settings, regression is often performed on massive datasets, where the data comes from a poorly-understood distribution and has not been thoroughly vetted or cleaned of outliers. This has prompted the study of robust regression, i.e. regression under weak distributional or corruption assumptions. In this work, we study the problem of regression (8.32) in the *strong contamination model*. In this model, we assume the data points we receive are independently drawn from $\mathcal{D}_{Xy}$, but that an arbitrary ϵ-fraction of the samples are then adversarially *contaminated* or replaced. The strong contamination model has recently drawn interest in the algorithmic statistics and learning communities for several reasons. Firstly, it is a *flexible* model of corruption and can be used to study both truly adversarial data poisoning attacks (where e.g. part of the dataset is sourced from malicious respondents), as well as model misspecification, where the generative $\mathcal{D}_{Xy}$ does not exactly satisfy our distributional assumptions, but is close in total variation to a distribution that does. Furthermore, a line of work building upon [175, 337] (discussed in our survey of prior work in Section 8.6.2) has achieved remarkable positive results for mean estimation and related problems under strong contamination, with statistical guarantees scaling independently of the dimension d. This dimension-free error promise is important in modern high-dimensional settings.

8.6.1 Our results

We give multiple *nearly-linear time algorithms*[8] for problem (8.32) under the strong contamination model, with improved statistical or runtime guarantees compared to the state-of-the-art. Prior algorithms for (8.32) under the strong contamination model typically followed one of two frameworks. The first, which we refer to as *robust gradient descent*, was pioneered by [447], and is based on reframing (8.32) as a problem where we have noisy gradient access to an unknown function we wish to optimize, coupled with the design of a noisy gradient oracle based on a robust mean estimation primitive. The second, which we refer to as *Sever*, originated in work of [176], and uses the guarantees of stationary point finders such as stochastic gradient descent to repeatedly perform outlier removal. In this work, we show that both approaches can be sped up dramatically, and give two complementary types of algorithms within these frameworks.

Robust acceleration. Our first contribution is to demonstrate that within the noisy gradient estimation framework for minimizing well-conditioned regression problems of the form (8.32), an *accelerated* rate of optimization can be achieved, answering an open question asked by [447]. We demonstrate the following result for smooth statistical regression problems, where we assume the uncorrupted data is drawn from $\mathcal{D}_{Xy}$ with marginals $\mathcal{D}_X$ and $\mathcal{D}_y$, $\mathcal{D}_X$ has support in $\mathbb{R}^d$, and $\widetilde{O}$ hides polylogarithmic factors in problem parameters (cf. Section 8.3.1 for technical definitions). Finally, we remove κ dependences in sample complexity statements, as they are subsumed by ϵ dependences

[8] Throughout, we reserve the description "nearly-linear" for runtimes scaling linearly in the dataset size nd, and polynomially in ϵ^{-1} and the condition number, up to a polylogarithmic overhead in problem parameters.

due to assumed bounds on $\epsilon\kappa$ or $\epsilon\kappa^2$.

Theorem 52 (informal, see Theorem 58)**.** *Suppose $\gamma : \mathbb{R}^2 \to \mathbb{R}$ is such that $\gamma_y(v) := \gamma(v, y)$ is convex and has (absolute) first and second derivatives at most 1 for all y in the support of $\mathcal{D}_y$, and $\mathcal{D}_X$ has second moment matrix $\mathbf{\Sigma}^\star \preceq L \cdot \mathbf{I}$. For some $\mu \geq 0$, let $\kappa = \max(1, \frac{L}{\mu})$ and let*

$$\theta^\star_{\text{reg}} := \text{argmin}_{\theta \in \mathbb{R}^d} \left\{ \mathbb{E}_{(X,y)\sim\mathcal{D}_{Xy}} \left\{ \gamma\left(\langle \theta, X \rangle, y \right) \right\} + \frac{\mu}{2} \left\| \theta \right\|_2^2 \right\}$$

be the solution to the true regularized statistical regression problem.[9] *There is an algorithm that given $n := \widetilde{O}(\frac{d}{\epsilon})$ ϵ-corrupted samples from $\mathcal{D}_{Xy}$, for $\epsilon\kappa^2$ at most an absolute constant, runs in time $\widetilde{O}(nd\sqrt{\kappa})$ and obtains θ with $\left\| \theta - \theta^\star_{\text{reg}} \right\|_2 = O\left(\sqrt{\frac{\kappa\epsilon}{\mu}} \right)$ with probability at least $1 - \delta$.*

A canonical example of a link function γ satisfying the assumptions of Theorem 52 is the logit function $\gamma(v, y) = \log(1 + \exp(-vy))$, when the labels y are ± 1. To contextualize Theorem 52, the earlier work [447] obtains a similar statistical guarantee in its setting, using $\widetilde{O}(\kappa)$ calls to a *noisy gradient oracle*, which they implement via a subroutine inspired by works on robust mean estimation. At the time of its initial dissemination, nearly-linear time robust mean estimation algorithms were not known; since then, [134] showed that for the case of linear regression (see Theorem 54 for the formal setup, as the linear regression link function is not Lipschitz), the framework was amenable to mean estimation techniques of [131], and gave an algorithm running in time $\widetilde{O}(nd\kappa\epsilon^{-6})$. Theorem 52 represents an improvement to these results on two fronts: we apply tools from the mean-estimation algorithm of [194] to remove the $\text{poly}(\epsilon^{-1})$ runtime dependence for a general class of regression problems, and we achieve an iteration count of $\widetilde{O}(\sqrt{\kappa})$, matching the accelerated gradient descent runtime of [417] for smooth optimization in the non-robust setting. We remark that the application of tools inspired by [194] is fairly straightforward, and not a primary contribution of our work compared to the accelerated dependence on κ.

We demonstrate the generality of our acceleration framework by demonstrating that it applies to optimizing the *Moreau envelope* for Lipschitz, but possibly non-smooth, link functions γ; a canonical example of such a function is the hinge loss $\gamma(v, y) = \max(0, 1 - vy)$ with ± 1 labels, used in training support vector machines. The Moreau envelope is a well-studied smooth approximation to a non-smooth function which everywhere additively approximates the original function if it is Lipschitz (see e.g. [485]), and in the non-robust setting many state-of-the-art rates for Lipschitz optimization are known to be attained by accelerated optimization of an appropriate Moreau envelope [502]. We show that even without explicit access to the Moreau envelope, we can obtain approximate minimizers to it through our robust acceleration framework.

[9] To simplify our bounds and avoid estimation error for non-strongly convex statistical regression problems scaling with the initial search radius (which may be dimension-dependent), we focus on regularized regression problems. There is a substantial line of work on reductions between rates for strongly convex and convex smooth optimization in the non-robust setting, see e.g. [551], and we defer an analogous exploration in the robust setting to future work.

Theorem 53 (informal, see Theorem 59)**.** *Suppose $\gamma : \mathbb{R}^2 \to \mathbb{R}$ is such that $\gamma_y(v) := \gamma(v, y)$ is convex and has (absolute) first derivative at most 1 for all y in the support of $\mathcal{D}_y$, and $\mathcal{D}_X$ has bounded second moment matrix. For some $\mu, \lambda \geq 0$, let $\kappa = \max(1, \frac{1}{\lambda\mu})$ and let*

$$\theta^\star_{\text{env}} := \text{argmin}_{\theta \in \mathbb{R}^d} \left\{ F^\star_\lambda(\theta) + \frac{\mu}{2} \|\theta\|_2^2 \right\}, \text{ where } F^\star_\lambda(\theta) := \inf_{\theta'} \left\{ F^\star(\theta') + \frac{1}{2\lambda} \|\theta - \theta'\|_2^2 \right\}$$
$$\text{is the Moreau envelope of } F^\star(\theta) := \mathbb{E}_{(X,y)\sim\mathcal{D}_{Xy}} \left\{ \gamma\left(\langle \theta, X \rangle, y \right) \right\}.$$

There is an algorithm that given $n := \widetilde{O}(\frac{d}{\epsilon})$ ϵ-corrupted samples from $\mathcal{D}_{Xy}$, for $\epsilon\kappa^2$ at most an absolute constant, runs in time $\widetilde{O}(\frac{nd\sqrt{\kappa}}{\epsilon})$ and obtains θ with $\left\|\theta - \theta^\star_{\text{reg}}\right\|_2 = O\left(\sqrt{\frac{\kappa\epsilon}{\mu}}\right)$ with probability at least $1 - \delta$.

To obtain this result, we give a nearly-linear time construction of a noisy gradient oracle for the Moreau envelope, which may be of independent interest; we note similar gradient oracle constructions in different settings have been developed in the optimization literature (see e.g. [110]).

Robust linear regression. The specific problem of robust linear regression is perhaps the most ubiquitous example of statistical regression [321, 315, 188, 550, 134, 53]. Amongst the algorithms developed for this problem, the only nearly-linear time algorithm is the recent work of [134]. For an instance of robust linear regression with noise variance bounded by σ^2 and covariate second moment matrix $\mathbf{\Sigma}^\star := \mathbb{E}_{X\sim\mathcal{D}_X}[XX^\top]$, the algorithms of [176, 447, 134] attain distance to the true regression minimizer $\theta^\star$ scaling as $\sigma\kappa\sqrt{\epsilon}$ in the $\mathbf{\Sigma}^\star$ norm (the "Mahalanobis distance") under a bounded 4^{th} moment assumption. We measure error in the $\mathbf{\Sigma}^\star$ norm as it is scale invariant and the natural norm in which to measure the underlying (quadratic) statistical regression error.[10] We give one result (Theorem 54) which improves the runtime of [176, 447, 134] under the noisy gradient descent framework, and one result (Theorem 55) which improves its estimation rate, under the Sever framework.

We first demonstrate that directly applying our robust acceleration framework leads to a similar estimation guarantee as [176, 447, 134] under the same assumptions.

Theorem 54 (informal, see Theorem 57)**.** *Suppose $\mathcal{D}_X$ is a 2-to-4 hypercontractive distribution with second moment matrix $\mathbf{\Sigma}^\star = \mathbb{E}_{X\sim\mathcal{D}_X}[XX^\top]$ satisfying $\mu \cdot \mathbf{I} \preceq \mathbf{\Sigma}^\star \preceq L \cdot \mathbf{I}$, and $y \sim \mathcal{D}_y$ is generated as $\langle \theta^\star, X \rangle + \delta$, for $\delta \sim \mathcal{D}_\delta$ with variance at most σ^2 independent of X. Let $\kappa := \frac{L}{\mu}$. There is an algorithm,* RobustAccel*, that given $n := \widetilde{O}(\frac{d}{\epsilon})$ ϵ-corrupted samples from $\mathcal{D}_{Xy}$, for $\epsilon\kappa^2$ at most an absolute constant, runs in time $\widetilde{O}(nd\sqrt{\kappa})$ and obtains θ with $\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} = O(\sigma\kappa\sqrt{\epsilon})$ with probability $1 - \delta$.*

We give a formal definition of 2-to-4 hypercontractivity in Section 8.3.1; as a lower bound of [53] shows, attaining estimation rates for robust linear regression scaling polynomially in ϵ is impossible under only bounded second moments, and such a 4^{th} moment bound is the minimal assumption under

[10] Some prior works gave ℓ_2 norm guarantees, which we have translated for comparison.

which robust estimation is known to be possible. Theorem 57 matches the distribution assumptions and error of [134], while obtaining an accelerated runtime.

Interestingly, under the 4^{th} moment bound used in Theorem 57, [53] showed that the information-theoretically optimal rate of estimation in the $\mathbf{\Sigma}^\star$ norm is *independent* of κ, and presented a matching upper bound under an analogous, but more stringent, distributional assumption.[11] We remark that thus far robust linear regression algorithms have broadly fallen under two categories: The first category (e.g. [321, 550, 53]), based on the sum-of-squares paradigm for algorithm design, sacrifices practicality to obtain improved error rates by paying a large runtime and sample complexity overhead (as well as requiring stronger distributional assumptions). The second (e.g. [176, 447, 134]), which opts for more practical approaches to algorithm design, has been bottlenecked at Mahalanobis distance $O(\sigma\kappa\sqrt{\epsilon})$ and the requirement that $\epsilon\kappa^2 = O(1)$.

We present a nearly-linear time method for robust linear regression overcoming this bottleneck for the first time amongst non-sum-of-squares algorithms, and attaining improved statistical performance compared to Theorem 54 while only requiring $\epsilon\kappa = O(1)$.

Theorem 55 (informal, see Theorem 56)**.** *Suppose $\mathcal{D}_X$ is a 2-to-4 hypercontractive distribution with second moment matrix $\mathbf{\Sigma}^\star = \mathbb{E}_{X\sim\mathcal{D}_X}[XX^\top]$ satisfying $\mu\cdot\mathbf{I} \preceq \mathbf{\Sigma}^\star \preceq L\cdot\mathbf{I}$, and $y \sim \mathcal{D}_y$ is generated as $\langle\theta^\star, X\rangle + \delta$, for $\delta\sim\mathcal{D}_\delta$, a 2-to-4 hypercontractive distribution with variance at most σ^2 independent of X. Let $\kappa := \frac{L}{\mu}$. There is an algorithm,* FastRegression*, that given $n := \widetilde{O}((\frac{d^2}{\epsilon^3} + \frac{d}{\epsilon^4}))$ ϵ-corrupted samples from $\mathcal{D}_{Xy}$, for $\epsilon\kappa$ at most an absolute constant, uses $\widetilde{O}(\frac{1}{\epsilon})$ calls to an empirical risk minimization routine[12] and $\widetilde{O}(\frac{nd}{\epsilon})$ additional runtime, and obtains θ with $\|\theta-\theta^\star\|_{\mathbf{\Sigma}^\star} = O(\sigma\sqrt{\kappa\epsilon})$ with probability at least $\frac{9}{10}$.*

This second algorithm does require more resources than that of Theorem 54: the sample complexity scales quadratically in d, and the runtime is never faster. Further, we make the slightly stronger assumption of hypercontractive noise for the uncorrupted samples. On the other hand, the improved dependence on the condition number in the error can be significant for distributions in practice, which may be far from isotropic. All told, Theorem 55 presents an intermediate tradeoff inheriting some statistical gains of the sum-of-squares approach (albeit still depending on κ) without sacrificing a nearly-linear runtime. Interestingly, we obtain Theorem 55 by reinterpreting an identifiability proof used in the algorithm of [53], and combining it with tools inspired by the Sever framework. Our sample complexity for Theorem 55 dramatically improves that of [176]'s original linear regression algorithm in the Sever framework for moderate ϵ, which used $\widetilde{O}(\frac{d^5}{\epsilon^2})$ samples (in addition to beating their weaker error guarantee). We elaborate on these points further in Section 8.2.2; we believe it is an interesting open problem to understand if the worse sample complexity

[11] The algorithm of [53] requires $\mathcal{D}_X$ to be *certifiably hypercontractive*, an algebraic condition frequently required by the sum-of-squares algorithmic paradigm to apply to robust statistical estimation problems.

[12] The empirical risk minimization algorithm used is up to the practitioner; its runtime will never scale worse than $\widetilde{O}(nd\sqrt{\kappa})$ by applying (non-robust) accelerated gradient descent, but can be substantially better if recent advances in stochastic gradient methods are used, e.g. [18].

of Theorem 55 is truly necessary for the improved statistical guarantees.

Table 8.1: Robust linear regression results in the $\kappa \gg 1$ regime. All listed results assume 2-to-4 hypercontractivity and independent noise (although some also give rates for non-independent noise). Error guarantees are in Mahalanobis distance. We omit polylogarithmic factors for simplicity.

Reference	Runtime	Error guarantee	Range of ϵ	Comments
[321]	$\mathrm{poly}(d)$	$\sigma\epsilon^{\frac{1}{4}}$	N/A	Certifiable hypercontractivity
[550]	$\mathrm{poly}(d)$	$\sigma\epsilon^{\frac{1}{2}}$	N/A	Certifiable hypercontractivity
[53]	$\mathrm{poly}(d)$	$\sigma\epsilon^{\frac{3}{4}}$	N/A	Certifiable hypercontractivity
[176]	$\mathrm{poly}(d)$	$\sigma\kappa\epsilon^{\frac{1}{2}}$	$O(\kappa^{-2})$	Linear sample size
[134]	$nd\kappa \cdot \mathrm{poly}(\epsilon^{-1})$	$\sigma\kappa\epsilon^{\frac{1}{2}}$	$O(\kappa^{-2})$	Linear sample size
Theorem 54	$nd\sqrt{\kappa}$	$\sigma\kappa\epsilon^{\frac{1}{2}}$	$O(\kappa^{-2})$	Linear sample size
Theorem 55	$nd \cdot \mathrm{poly}(\epsilon^{-1})$	$\sigma(\kappa\epsilon)^{\frac{1}{2}}$	$O(\kappa^{-1})$	Quadratic sample size

8.6.2 Prior work

We give a general overview contextualizing our work in this section, and defer the comparison of specific technical components we develop in this work to relevant discussions.

The study of learning in the presence of adversarial noise is known as *robust statistics*, with a long history dating back over 60 years [31, 512, 278, 513, 279]. Despite this, the first efficient algorithms with near-optimal error for many fundamental high dimensional robust statistics problems were only recently developed [175, 337, 177]. Since these works, efficient robust estimators have been developed for a variety of more complex problems; a full survey of this field is beyond our scope, and we defer a more comprehensive overview to [182, 360, 492].

Our results sit within the line of work in this field on robust stochastic optimization. The first works which achieved dimension-independent error rates with efficient algorithms for the problems we consider are the aforementioned works of [447, 176]. Similar problems were previously considered in [118, 54]. In [118], the authors consider a setting where a majority of the data is corrupted, and the goal is to output a short list of hypotheses so that at least one is close to the true regressor. However, because most of their data is corrupted, they achieve weaker statistical rates; in particular, their techniques do not achieve vanishing error as the fraction of error goes to zero. In [54], the authors consider a somewhat different model with stronger assumptions on the structure of the functions. In particular, they assume that the uncorrupted covariates are Gaussian, and are primarily concerned with the case where the regressors are sparse. Their main goal is to achieve sublinear sample complexities by leveraging sparsity. We also remark that the algorithms in [118, 54] are also much more cumbersome, requiring heavy-duty machinery such as black-box SDP solvers and cutting plane methods, and as a result are more computationally intense than those considered in [447, 177].

There has been a large body of subsequent work on the special case of robust linear regression [321, 315, 188, 550, 134, 53]; however, the majority of this line of work focuses on achieving improved error rates under additional distributional assumptions by using the sum-of-squares hierachy. As a result, their algorithms are likely impractical in high dimensions, and require large (albeit polynomial) sample complexity and runtime. Of particular interest to us is [134], who combine the framework of [447] with the robust mean estimation algorithm of [131] to achieve nearly-linear runtimes in the problem dimension and the number of samples. Our Theorem 54 can be thought of as the natural accelerated version of [134], with an additional ϵ^{-6} runtime overhead removed using more sophisticated mean estimation techniques.

8.6.3 Techniques

We now describe the techniques we use to obtain the accelerated rates of Theorems 52, 53, and 54 as well as the robust linear regression algorithm of Theorem 55.

Robust acceleration. Our robust acceleration framework is based on the following abstract formulation of an optimization problem: there is an unknown function $F^\star : \mathbb{R}^d \to \mathbb{R}$ with minimizer $\theta^\star$ which is L-smooth and μ-strongly convex, and we wish to estimate $\theta^\star$, but our only mode of accessing $F^\star$ is through a *noisy gradient oracle* $\mathcal{O}_{\text{ng}}$. Namely, for some σ, ϵ, we can query $\mathcal{O}_{\text{ng}}$ at any point $\theta \in \mathbb{R}^d$ with an upper bound $R \geq \|\theta - \theta^\star\|_2$ and receive an estimate $G(\theta)$ such that

$$\|G(\theta) - \nabla F^\star(\theta)\|_2 = O\left(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R\right). \tag{8.33}$$

In other words, we receive gradients perturbed by both fixed additive noise, and multiplicative noise depending on the distance to $\theta^\star$. The prior works [176, 447, 134] observed that by using tools from robust mean estimation, appropriate noisy gradient oracles could be constructed for the functions $F^\star(\theta) = \mathbb{E}_{(X,y)\sim\mathcal{D}_{Xy}}[\sigma(\langle\theta, X\rangle - y)]$ arising from the distributional assumptions in Theorems 52, 53, and 54. Our first contribution is speeding up the implementation of $\mathcal{O}_{\text{ng}}$ to run in nearly-linear time $\widetilde{O}(nd)$, leveraging recent advances by [194] for robust mean estimation.

Our second, and more technically involved, contribution is demonstrating that accelerated runtimes are achievable under the noisy gradient oracle access model of (8.33). Designing accelerated algorithms under noisy gradient access is an extremely well-studied problem, and there are both strong positive results [162, 400, 209, 142, 395, 102] as well as negative results [172] showing that under certain noise models, accelerated gradient descent may be outperformed by unaccelerated methods. Indeed, it was asked (motivated by these negative results) as an open question in [447] whether an accelerated rate was possible under the noise model (8.33).

Our accelerated algorithm runs in logarithmically many phases, where we halve the distance to the optimizer (while it is above a certain noise floor depending on the additive error in (8.33)) in each phase. The subroutine we design for implementing each phase is a robust accelerated "outer

loop" tolerant to noisy gradient access in the manner provided by our oracle $\mathcal{O}_{\text{ng}}$. By carefully balancing the accuracy of subproblem solutions, the multiplicative error in our gradient estimates within the accelerated outer loop, and the drift of the phase's iterates (which may venture further from $\theta^\star$ than our initial iterate upper bound), we show that above the noise floor we can halve the distance to $\theta^\star$ in $\widetilde{O}(\sqrt{\kappa})$ queries to $\mathcal{O}_{\text{ng}}$; recursing on this guarantee yields our complete algorithm.

To obtain Theorem 53, we demonstrate that for Lipschitz functions $F^\star$ admitting a *radiusless* noisy gradient oracle, i.e. one which satisfies (8.33) with no dependence on R, we can further efficiently construct a noisy gradient oracle for the Moreau envelope of $F^\star$ using projected subgradient descent. This construction enables applying our robust accelerated method to Lipschitz regression problems.

Our acceleration framework crucially tolerates both additive and multiplicative guarantees for gradient estimation. While it is possible that arguments of other noisy acceleration frameworks e.g. [142] may be extended to capture our gradient noise model, we give a self-contained derivation specialized to our specific oracle access for convenience. We view our result as a proof-of-concept that acceleration is possible under this noise model; we believe a unified study of acceleration under noise models encompassing (8.33) warrants further exploration, and defer it to interesting future work.

Robust linear regression. For the special case of linear regression, as discussed earlier, the robust optimization methods of [176, 447, 134] attain Mahalanobis distance scaling as $\sigma\kappa\sqrt{\epsilon}$ from the true minimizer. Directly plugging in deterministic conditions proven by [134] to hold under an appropriate statistical model into our robust gradient descent framework, we obtain a similar guarantee (Theorem 54) at an accelerated rate. In this technical overview, we now focus on how we obtain the improvements of Theorem 55.

At a high level, prior works lose two factors of $\sqrt{\kappa}$ in their error guarantees because of two *norm conversions* from the $\mathbf{\Sigma}^\star$ norm to the ℓ_2 norm: one in gradient space, and one in parameter space. Because we do not have access to the true covariance $\mathbf{\Sigma}^\star$, it is natural to perform both gradient estimation and the gradient descent procedure itself in the ℓ_2 norm. When the ℓ_2 guarantees of both subroutines are converted back to the $\mathbf{\Sigma}^\star$ norm, the error rate is lossy by a factor of κ.

We give a different approach to robust linear regression which bypasses this barrier in parameter space, saving a factor of $\sqrt{\kappa}$ in our error rate. In particular, we measure progress of our parameter estimates entirely in the $\mathbf{\Sigma}^\star$ norm in our analysis, which removes the need for an additional norm conversion. Our starting point is the following *identifiability proof* guarantee of [53], which we slightly repurpose for our needs. Let $w \in \Delta^n$ be entrywise less than $\frac{1}{n}1,1$ such that $\|w_G\|_1 \geq 1 - 4\epsilon$ where G is our "uncorrupted" data, and let $\theta \in \mathbb{R}^d$. We demonstrate in Proposition 37 that

$$\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} = O\left(\sigma\sqrt{\kappa\epsilon} + \sqrt{\left\|\text{Cov}_w\left(\{g_i(\theta)\}_{i\in[n]}\right)\right\|_{\text{op}}\frac{\epsilon}{\mu}} + \sqrt{\epsilon}\,\|\nabla F_w(\theta)\|_{(\mathbf{\Sigma}^\star)^{-1}}\right). \tag{8.34}$$

In the above display, $g_i(\theta)$ is the empirical gradient of the squared loss at our i^{th} data point, $\mathrm{Cov}_w(\cdot)$ is the empirical second moment matrix of its argument under the weighting w, and F_w is the empirical risk under w. The guarantee (8.34) suggests a natural approach to estimation: if we can *simultaneously* verify that θ is an approximate minimizer to F_w, and that the empirical second moment of gradients at θ (according to w) are small, then we have a proof that θ and $\theta^\star$ are close.

Prior work by [53] used this approach to obtain a polynomial-time estimator by solving a joint optimization problem in (w, θ), via an appropriate semidefinite program relaxation. However, in designing near-linear time algorithms, we cannot afford to use said relaxation. This raises a chicken-and-egg issue: for fixed w, it is straightforward to make $\|\nabla F_w(\theta)\|_{(\mathbf{\Sigma}^\star)^{-1}}$ small, by setting θ to the empirical risk minimizer (ERM) of F_w. Likewise, for fixed θ, known *filtering* techniques rapidly decrease $\|\mathrm{Cov}_w(\{g_i(\theta)\})_{i\in[n]}\|_{\mathrm{op}}$ while preserving most of w_G, by using that the second moment restricted to uncorrupted points has a small operator norm as a certificate for outlier removal. However, performing either of these subroutines to guarantee one of our sufficient conditions passes (small operator norm or gradient norm) may adversely affect the quality of the other.

We circumvent this chicken-and-egg problem by introducing a *third* potential, namely the actual function value $F_w(\theta)$. In particular, notice that the two subroutines we described earlier (down-weighting w or setting θ to the ERM) both decrease this third potential. Our linear regression algorithm is an alternating procedure which iteratively filters w based on the gradients at the current θ (to make the operator norm small), and sets θ to the ERM of F_w (to zero out the gradient norm). We further show that if the ERM step does not make significant function progress (our third potential), then it was not strictly necessary to make progress according to (8.34), since the gradient norm was already small. This gives a dimension-independent bound on the number of times we could have alternated, via tracking function progress, yielding Theorem 55.

8.7 Preliminaries: robust regression

We give the notation used throughout the sections dedicated to robust regression in Section 8.7.1, and set up the statistical model we consider in Section 8.7.2. In Section 8.7.3, we give the deterministic regularity assumptions used by our regression algorithm in Section 8.8. In Section 8.7.4, we give the deterministic regularity assumptions used by our stochastic optimization algorithms in Sections 8.9 and 8.10. Finally, in Section 8.7.5, we state a nearly-linear time procedure for robustly decreasing the operator norm of the second moment matrix of a set of vectors. Some proofs are deferred to the appendices.

8.7.1 Notation

General notation. For $d \in \mathbb{N}$ we let $[d] := \{j \mid j \in \mathbb{N}, 1 \le j \le d\}$. The ℓ_p norm of a vector argument is denoted $\|\cdot\|_p$, where $\|\cdot\|_\infty$ is the element with largest absolute value; when the argument

is a symmetric matrix, we overload this to mean the Schatten-p norm. The all-ones vector (of appropriate dimension from context) is denoted $_{1,1}$. The (solid) probability simplex is denoted $\Delta^n := \{w \in \mathbb{R}^n_{\geq 0}, \|w\|_1 \leq 1\}$. We use $\widetilde{O}$ to suppress logarithmic factors in dimensions, distance ratios, the problem condition number κ, the inverse corruption parameter ϵ^{-1}, and the inverse failure probability. For $v \in \mathbb{R}^n$ and $S \subseteq [n]$, we let $v_S \in \mathbb{R}^n$ denote v with coordinates in $[n] \setminus S$ zeroed out. For a set S, we call S_1, S_2 a *bipartition* of S if $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$.

Matrices. Matrices are denoted in boldface. We denote the zero and identity matrices (of appropriate dimension) by $\mathbf{0}$ and $\mathbf{I}$. The $d \times d$ symmetric matrices are $\mathbb{S}^d$, and the $d \times d$ positive semidefinite cone is $\mathbb{S}^d_{\geq 0}$. For $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d$ we write $\mathbf{A} \preceq \mathbf{B}$ to mean $\mathbf{B} - \mathbf{A} \in \mathbb{S}^d_{\geq 0}$. The largest and smallest eigenvalue and trace of a symmetric matrix are respectively denoted $\lambda_{\max}(\cdot)$, $\lambda_{\min}(\cdot)$, and $\mathrm{Tr}(\cdot)$. The inner product on $\mathbb{S}^d$ is $\langle \mathbf{A}, \mathbf{B} \rangle := \mathrm{Tr}(\mathbf{AB})$. For positive definite $\mathbf{M}$, we define the induced norm $\|v\|_{\mathbf{M}} := \sqrt{v^\top \mathbf{M} v}$. We use $\|\cdot\|_{\mathrm{op}}$ to mean the ℓ_2-ℓ_2 operator norm of a matrix; when the argument is symmetric, it is synonymous with $\lambda_{\max}$, and otherwise is the largest singular value.

Functions. The gradient and Hessian of a twice-differentiable function are denoted ∇ and ∇^2. We say differentiable $f : \mathbb{R}^d \to \mathbb{R}$ is λ-Lipschitz in a quadratic norm $\|\cdot\|_{\mathbf{M}}$ if $\|\nabla f(\theta)\|_{\mathbf{M}^{-1}} \leq \lambda$ for all $\theta \in \mathbb{R}^d$. We say twice-differentiable $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex in $\|\cdot\|_{\mathbf{M}}$ if

$$\mu \mathbf{M} \preceq \nabla^2 f(\theta) \preceq L\mathbf{M}, \text{ for all } \theta \in \mathbb{R}^d.$$

When $\mathbf{M}$ is not specified, we assume $\mathbf{M} = \mathbf{I}$ (i.e. the norm in question is ℓ_2). For any $\mathbf{M}$, smoothness and strong convexity imply the following bounds for all $\theta, \theta' \in \mathbb{R}^d$,

$$f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{\mu}{2} \|\theta' - \theta\|^2_{\mathbf{M}} \leq f(\theta') \leq f(\theta) + \langle \nabla f(\theta), \theta' - \theta \rangle + \frac{L}{2} \|\theta' - \theta\|^2_{\mathbf{M}}.$$

It is well-known that L-smoothness of function f implies L-Lipschitzness of the function gradient ∇f, i.e. $\|\nabla f(\theta) - \nabla f(\theta')\|_{\mathbf{M}^{-1}} \leq L \|\theta - \theta'\|_{\mathbf{M}}$ for all $\theta, \theta' \in \mathbb{R}^d$. For any f which is L-smooth and μ-strongly convex in $\|\cdot\|_{\mathbf{M}}$, with $\theta^* := \mathrm{argmin}_{\theta \in \mathbb{R}^d} f(\theta)$, it is straightforward to show

$$\frac{1}{2L} \|\nabla f(\theta)\|^2_{\mathbf{M}^{-1}} \leq f(\theta) - f(\theta^\star) \leq \frac{1}{2\mu} \|\nabla f(\theta)\|^2_{\mathbf{M}^{-1}} \text{ for all } \theta \in \mathbb{R}^d.$$

Distributions. The multivariate Gaussian distribution with mean μ and covariance $\mathbf{\Sigma}$ is denoted $\mathcal{N}(\mu, \mathbf{\Sigma})$. For weights $w \in \mathbb{R}^n_{\geq 0}$ and a set of vectors $\mathbf{X} := \{X_i\}_{i \in [n]}$, we let

$$\mu_w(\mathbf{X}) := \sum_{i \in [n]} \frac{w_i}{\|w\|_1} X_i, \ \mathrm{Cov}_{w, \bar{X}}(\mathbf{X}) := \sum_{i \in [n]} \frac{w_i}{\|w\|_1} \left(X_i - \bar{X}\right) \left(X_i - \bar{X}\right)^\top.$$

be the empirical mean and (centered) covariance matrix; when $\bar{X}$ is not specified, it is the zeroes vector. Draws from the uniform distribution on $\{X_i\}_{i \in [n]}$ are denoted $X \sim_{\mathrm{unif}} \mathbf{X}$. We say distribution

$\mathcal{D}$ supported on $\mathbb{R}^d$ is 2-to-4 hypercontractive with parameter $C_{2\to 4}$ if for all $v \in \mathbb{R}^d$,

$$\mathbb{E}_{X\sim\mathcal{D}}\left[\langle X, v\rangle^4\right] \le C_{2\to 4}\mathbb{E}_{X\sim\mathcal{D}}\left[\langle X, v\rangle^2\right]^2.$$

We will refer to this property as being $C_{2\to 4}$-hypercontractive for short; by massaging the definition, we observe $C_{2\to 4}$-hypercontractivity is preserved under linear transformations of the distribution.

Filtering. We will make much use of the following algorithmic technique, which refer to as *filtering*. In the filtering paradigm, we have an index set $[n]$, and a fixed, unknown bipartition $[n] = G \cup B$, $G \cap B = \emptyset$. The set G is a "good" set of indices that we wish to keep, and the set B is a set of "bad" indices which we would like to remove. The algorithm maintains a set of weights $w \in \Delta^n$ (with the goal of producing a weight vector which is close to the uniform distribution on G). These weights are iteratively updated according to "scores" $\tau \in \mathbb{R}^n_{\ge 0}$; the goal of filtering is to assign large scores to coordinates in B and small scores to coordinates in G, so that the bad coordinates can be filtered out according to their scores. Concretely, we use the following definition.

Definition 30 (saturated weights)**.** *We say weights $w \in \Delta^n$ are* c-saturated *with respect to the bipartition $G \cup B = [n]$ if $w \le \frac{1}{n}1_{1,1}$ entrywise, and*

$$\left\|\left[\frac{1}{n}1_{1,1} - w\right]_G\right\|_1 \le \left\|\left[\frac{1}{n}1_{1,1} - w\right]_B\right\|_1 + c.$$

If $c = 0$, we refer to w as simply saturated*.*

In words, w is saturated if its difference from the uniform distribution has more weight on B than G (in the context of our algorithm, if we have started with uniform weights and produced a saturated w, then we have removed more mass from B than G). We allow for a "fudge factor" of an additive c to relax the above definition, which will come in handy in our linear regression applications.

Definition 31 (safe scores)**.** *Suppose $G \cup B$ is a bipartition of $[n]$, and suppose $w \in \Delta^n$ is a set of weights. We call a set of scores $\tau = \{\tau_i\}_{i\in[n]} \in \mathbb{R}^n_{\ge 0}$* safe with respect to w *if it satisfies*

$$\langle w_G, \tau\rangle \le \langle w_B, \tau\rangle.$$

The following simple lemma (implicit in prior works [177, 118, 360, 492]) is the crux of the filtering paradigm, relating these two definitions.

Lemma 130. *Suppose $w \in \Delta^n$ is saturated, and $\tau \in \mathbb{R}^n_{\ge 0}$ is safe with respect to w. Defining w' by*

$$w'_i \leftarrow \left(1 - \frac{\tau_i}{\tau_{\max}}\right) w_i \text{ for all } i \in [n], \text{ and } \tau_{\max} := \max_{i\in[n]|w_i\neq 0} \tau_i,$$

then $w' \in \Delta^n$ is also saturated.

Proof. If $G \cup B = [n]$ is the bipartition with good coordinates G, then by definition of safe scores,

$$\|[w - w']_G\|_1 = \frac{1}{\tau_{\max}} \langle w_G, \tau \rangle \leq \frac{1}{\tau_{\max}} \langle w_G, \tau \rangle \leq \|[w - w']_B\|_1 .$$

Now, since w is saturated, and since $w' \leq w \leq \frac{1}{n} 1_{,1}$ by definition of saturation,

$$\left\| \left[\tfrac{1}{n} 1_{,1} - w' \right]_G \right\|_1 = \left\| \left[\tfrac{1}{n} 1_{,1} - w \right]_G \right\|_1 + \|[w - w']_G\|_1 \leq \left\| \left[\tfrac{1}{n} 1_{,1} - w \right]_B \right\|_1 + \|[w - w']_B\|_1 = \left\| \left[\tfrac{1}{n} 1_{,1} - w' \right]_G \right\|_1 .$$

□

We will also frequently using the following simple fact.

Lemma 131. *Suppose $w \in \Delta^n$ is c-saturated with respect to bipartition $[n] = G \cup B$, and suppose the bad set $|B| \leq \epsilon n$. Let $\tilde{w} = \frac{w}{\|w\|_1}$ be the distribution with probabilities proportional to w and $w^\star_G = \frac{1}{|G|} \mathbf{1}_G$ be uniform over G. Then, $\|\tilde{w} - w^\star_G\|_1 \leq 6\epsilon + 2c$.*

Proof. By the definition of saturation, since there is only ϵ mass to remove from the coordinates of B on $\frac{1}{n} 1_{,1}$, clearly $\|w\|_1 \geq 1 - 2\epsilon - c$. By the triangle inequality, we have

$$\|\tilde{w} - w^\star_G\|_1 \leq \|\tilde{w} - w\|_1 + \left\| w - \tfrac{1}{n} 1_{,1} \right\|_1 + \left\| \tfrac{1}{n} 1_{,1} - w^\star_G \right\|_1 .$$

By definition of $\tilde{w}$, $\|\tilde{w} - w\|_1 = 1 - \|w\|_1 \leq 2\epsilon + c$. By saturation, $\left\| w - \frac{1}{n} 1_{,1} \right\|_1 \leq 2\epsilon + c$. Finally, since $|G| \geq (1 - \epsilon)n$, $\left\| \frac{1}{n} 1_{,1} - w^\star_G \right\|_1 \leq 2\epsilon$. Combining these pieces yields the claim. □

8.7.2 Our statistical models

We provide provable guarantees for optimization problems captured by the following statistical model.

Model 1 (stochastic optimization in the strong contamination model)**.** *For $\mathcal{D}_f$ a distribution over functions $f : \mathbb{R}^d \to \mathbb{R}$, our goal is to optimize $\mathbb{E}_{f \sim \mathcal{D}_f}[f(\theta)]$. We are given access to n samples $\{f_i\}_{i \in [n]}$ produced as follows:*

1. *Functions $\{\tilde{f}_i\}_{i \in [n]}$ are drawn independently from $\mathcal{D}_f$.*
2. *An arbitrary subset $B \subset [n]$ of the samples is replaced with arbitrary functions from* $\mathrm{supp}(\mathcal{D}_f)$.
3. *For each $i \in [n]$, if $i \in B$ we observe f_i as the corrupted sample, otherwise, we observe $f_i = \tilde{f}_i$.*

We call B the corrupted *samples. When $|B| = \epsilon n$, we say $\{f_i\}_{i \in [n]}$ is drawn* ϵ-corrupted from $\mathcal{D}_f$.

Throughout we use the convention that $G \cup B = [n]$ is the bipartition of sample coordinates with B the corrupted samples and G the "good" samples. For simplicity, we assume throughout that

$\epsilon = \frac{|B|}{n}$ is smaller than some globally fixed constant. We will also frequently use the notation g_i to mean ∇f_i for all $i \in [n]$. When $\mathcal{D}_f$ is clear from context, we denote the "true average function" by

$$F^\star(\theta) := \mathbb{E}_{f \sim \mathcal{D}_f}\left[f(\theta)\right].$$

For $w \in \Delta^n$ and functions $\{f_i\}_{i\in[n]}$, the (unnormalized) weighted empirical average function is

$$F_w(\theta) := \sum_{i\in[n]} w_i f_i(\theta). \tag{8.35}$$

We will use $w_G^\star$ to denote the uniform distribution over G, $w_G^\star := \frac{1}{|G|}\mathbf{1}_G$, and we use F_G as shorthand for the function $F_{w_G^\star}$. The goal of robust parameter estimation is to estimate the true optimizer, which we always denote by $\theta^\star := \operatorname{argmin}_{\theta\in\mathbb{R}^d} F^\star(\theta)$. For example, the problem estimating the mean of $\mathcal{D}$ can be expressed by choosing $f_i(\theta) = \frac{1}{2}\left\|\theta - X_i\right\|_2^2$ for $X_i \sim \mathcal{D}$. In the uncorrupted setting (i.e. $\epsilon = 0$), a typical strategy (given reasonable regularity assumptions on $\mathcal{D}_f$) is to choose the estimator $\theta_G := \operatorname{argmin}_{\theta\in\mathbb{R}^d} F_G(\theta)$. The challenge is to obtain comparable estimation performance to θ_G which is robust to an ϵ-fraction of unknown corruptions.

Our focus is optimizing *generalized linear models*. In particular, throughout we will work only with $\mathcal{D}_f$ of the following form.

Model 2 (generalized linear model)**.** *A* generalized linear model *is a distribution over functions $f_i : \mathbb{R}^d \to \mathbb{R}$ which is defined by a joint distribution $\mathcal{D}_{Xy}$ over pairs $\{X_i, y_i\} \in \mathbb{R}^d \times \mathbb{R}$ and a* link function *$\gamma : \mathbb{R}^2 \to \mathbb{R}$, so that samples $f_i \sim \mathcal{D}_f$ are generated as*

$$f_i(\theta) := \gamma\left(\left\langle X_i, \theta\right\rangle, y_i\right), \quad \text{for } (X_i, y_i) \sim \mathcal{D}_{Xy}. \tag{8.36}$$

Note that observing $\{f_i\}_{i\in[n]}$ is equivalent to observing the dataset $\{X_i, y_i\}_{i\in[n]}$ when γ is known.

For instance, when $\gamma(v, y) = \frac{1}{2}(v - y)^2$, this is the problem of (statistical) linear regression. Further, when $\gamma(v, y) = \log(1 + \exp(-vy))$, our problem is logistic regression, and when $\gamma(v, y) = \max(0, 1 - vy)$, it is fitting a support vector machine. We refer to the X and y marginals over $\mathcal{D}_{Xy}$ respectively by $\mathcal{D}_X$ and $\mathcal{D}_y$, and we denote $\mathbf{\Sigma}^\star := \mathbb{E}_{X\sim\mathcal{D}_X}\left[XX^\top\right]$ when $\mathcal{D}_X$ is clear from context.

8.7.3 Linear regression

In Section 8.8 and (part of) Section 8.9, we develop algorithms for the well-studied special case of the generalized linear model, Model 2 wherein $\gamma(v, y) = \frac{1}{2}(v - y)^2$, i.e. a statistical variant of linear regression. We obtain guarantees under the following model and regularity assumptions for $\mathcal{D}_{Xy}$.

Model 3 (distributional regularity for linear regression)**.** *Given distributions $\mathcal{D}_X$ and $\mathcal{D}_\delta$ over $\mathbb{R}^d, \mathbb{R}^d$ respectively, the distribution $\mathcal{D}_{Xy}$ over $\mathbb{R}^d \times \mathbb{R}$ is sampled as follows: for an underlying vector*

$\theta^\star \in \mathbb{R}^d$, independently sample $X \sim \mathcal{D}_X$ and $\delta \sim \mathcal{D}_\delta$, and set $y \leftarrow \langle \theta^\star, X\rangle + \delta$. Further, $\mathcal{D}_X$ and $\mathcal{D}_\delta$ satisfy the following regularity assumptions.

1. *For $\mathbf{\Sigma}^\star = \mathbb{E}_{\mathcal{D}_X}[XX^\top]$ and $0 < \mu < L$, we have $\mu\mathbf{I} \preceq \mathbf{\Sigma}^\star \preceq L\mathbf{I}$.*
2. *$\mathcal{D}_X$ is $C_{2\to 4}$-hypercontractive for a constant $C_{2\to 4}$.*
3. *$\mathcal{D}_\delta$ is a $C_{2\to 4}$-hypercontractive distribution with mean zero and variance $\le \sigma^2$.*

For $\{(X_i, y_i)\}_{i\in[n]}$ (in particular, overloading to include $i \in B$), we use the notation $\delta_i := y_i - \langle X_i, \theta^\star\rangle$. We also use the following notation when discussing linear regression:

$$f_i(\theta) := \frac{1}{2}\left(\langle X_i, \theta\rangle - y_i\right)^2, \; g_i(\theta) := \nabla f_i(\theta) = X_i\left(\langle X_i, \theta\rangle - y_i\right). \tag{8.37}$$

We will denote the condition number of $\mathbf{\Sigma}^\star$ by $\kappa := \frac{L}{\mu}$ throughout. Under Model 3, it is immediate from the first-order optimality condition that for

$$F^\star(\theta) := \mathbb{E}_{X,y\sim\mathcal{D}_{Xy}}\left[\frac{1}{2}\left(\langle X, \theta\rangle - y\right)\right]^2,$$

the optimizer $\mathrm{argmin}_{\theta\in\mathbb{R}^d} F^\star(\theta)$ is exactly $\theta^\star$.

In our setting, following the description in Section 8.7.2 we independently draw $\{(X_i, y_i)\}_{i\in G} \sim \mathcal{D}_{Xy}$ for $|G| = (1-\epsilon)n$ and observe $\{(X_i, y_i)\}_{i\in[n]}$ where $[n] = G \cup B$ and $\{(X_i, y_i)\}_{i\in B}$ are arbitrarily chosen. We will frequently refer to $\{X_i\}_{i\in[n]}$ as $\mathbf{X} \in \mathbb{R}^{n\times d}$. Under Model 3, recent work [53] obtained the following results.

Proposition 32 ([53], Theorem 1.7, Theorem 1.9, Theorem 1.2)**.** *For Models 1 and 3, the minimax optimal error rate for estimators $\hat\theta$ is*

$$\left\|\hat\theta - \theta^\star\right\|_{\mathbf{\Sigma}^\star} = O\left(\sigma\epsilon^{\frac{3}{4}}\right).$$

When the distribution $\mathcal{D}_X$ is further certifiably hypercontractive *in the sum-of-squares proof system, there is a* poly(d)*-time estimator requiring* poly(d) *samples achieving this rate with high probability. Moreover, without the hypercontractivity condition in Model 3, even when $\mu, L = \Theta(1)$ it is information-theoretically impossible to attain an error rate depending polynomially on ϵ.*

The algorithmic result of [53] (and all known techniques with error rate $\ll \sqrt{\epsilon}$) crucially requires that the distributions are sum-of-squares certifiably hypercontractive, which is a stronger assumption than (standard) hypercontractivity. There is evidence that the problem of certifying $2 \to 4$ hypercontractivity is computationally intractable in general (under e.g. the small-set expansion hypothesis, see [58, 78]). Even for certifiably hypercontractive distributions, known algorithms require use spectral estimators of higher-order moment matrices, and thus more samples and increased runtime complexity. Hence, error $\sqrt{\epsilon}$ is a standing barrier for fast algorithms.

In Section 8.8, whenever we discuss robust linear regression we operate in Models 1 and 3. These assumptions imply that the data $\{X_i, y_i\}_{i\in[n]}$ will satisfy the following deterministic conditions with probability $\frac{9}{10}$. For convenience we work with these deterministic conditions directly in our proofs.

Assumption 7 (deterministic regularity for linear regression)**.** *Let ϵ be sufficiently small, and let $r \in (0, \epsilon^2)$. Assume $\{(X_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{R}$ is (ϵ, r)-*good for linear regression *(or (ϵ, r)-good if context is clear), which means there is a partition $[n] = G \cup B$ with $|G| \geq (1-\epsilon)n$ which satisfies:*

1. *For any $w \in \Delta^n$ with $\|w_G\|_1 \geq 1 - 4\epsilon$, $\frac{1}{2}\mathbf{\Sigma}^\star \preceq \mathrm{Cov}_{w_G}(\mathbf{X}) \preceq \frac{3}{2}\mathbf{\Sigma}^\star$.*

2. *There is a constant C_{est} such that for all $\theta \in \mathbb{R}^d$, there exists a $G' \subseteq G$ satisfying $|G'| \geq (1-r)|G|$ such that for all ϵ-saturated $w \in \Delta^n$, if we let $\tilde{w} := \frac{w_{G'}}{\|w_{G'}\|_1}$,*

$$\|\nabla F_{\tilde{w}}(\theta) - \nabla F^\star(\theta)\|_2 \leq C_{\mathrm{est}}\sqrt{L\epsilon}\left(\sigma + \|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star}\right), \tag{8.38}$$

$$\left\|\mathrm{Cov}_{\tilde{w}}\left(\{g_i(\theta)\}_{i\in G'}\right)\right\|_{\mathrm{op}} \leq C_{\mathrm{est}} L\left(\|\theta - \theta^\star\|^2_{\mathbf{\Sigma}^\star} + \sigma^2\right). \tag{8.39}$$

3. *There is a constant C_{ub} such that*

$$\sum_{i\in G}\frac{1}{|G|}f_i(\theta^\star) = \frac{1}{2|G|}\sum_{i\in G}\left(\langle X_i, \theta^\star\rangle - y_i\right)^2 \leq C_{\mathrm{ub}}\sigma^2.$$

We defer the proof of the following claim, which establishes the probabilistic validity of Assumption 7 (up to adjusting constants in definitions) under the statistical Models 1 and 3, to Appendix G.6.

Proposition 33. *Let $\alpha \geq 1$ and let $\epsilon > 0$ be sufficiently small. Let $\{(X_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{R}$ be an ϵ-corrupted set of samples from a distribution $\mathcal{D}_{Xy}$ as in Model 3. Then, if*

$$n = O\left(\frac{d\alpha^2 \log d}{\epsilon^4} + \frac{d^2\alpha^{1.5}\log(d/\epsilon)}{\epsilon^3}\right),$$

the set $\{(X_i, y_i)\}_{i\in[n]}$ is $(2\epsilon, \frac{\epsilon^2}{\alpha})$-good for linear regression with probability at least $\frac{9}{10}$.

Remark 9. *We remark that the gaurantees of Proposition 33 may be recovered with $n = O(d\log d/\epsilon^4 + d^2 \log d \log\frac{1}{\epsilon})$ samples when the X_i are further assumed to be subgaussian.*

We observe that Assumption 7 implies the following useful bound.

Lemma 132. *Let $w \in \Delta^n$ be ϵ-saturated with respect to bipartition $[n] = G \cup B$, let $G^\star \subseteq G$ be the subset in Assumption 7.2 corresponding to $\theta^\star$, and let $\tilde{w} = \frac{w_{G^\star}}{\|w_{G^\star}\|_1}$. Let $\theta_{\tilde{w}} = \mathrm{argmin}_{\theta\in\mathbb{R}^d} F_{\tilde{w}}(\theta)$ be the empirical minimizer of $F_{\tilde{w}}$. Then,*

$$\|\theta_{\tilde{w}} - \theta^\star\|_{\mathbf{\Sigma}^\star} \leq 4C_{\mathrm{est}}\sigma\sqrt{\kappa\epsilon}.$$

Proof. Applying Assumption 7.2 with $\theta = \theta^\star$, we have that

$$\|\nabla F_{\tilde{w}}(\theta^\star)\|_2 \le C_{\text{est}}\sqrt{L\epsilon}\sigma.$$

However, applying Assumption 7.1 on the weights $w_{G^\star}$ implies that $F_{\tilde{w}}$ is $\frac{1}{2}$-strongly convex in the $\mathbf{\Sigma}^\star$ norm (since $w_{G^\star}$ removes at most ϵ^2 mass from w_G) and minimized by $\theta_{\tilde{w}}$, and hence by using consequences of strong convexity and $(\mathbf{\Sigma}^\star)^{-1} \preceq \frac{1}{\mu}\mathbf{I}$,

$$\|\nabla F_{\tilde{w}}(\theta^\star)\|_2 \ge \sqrt{\mu}\,\|\nabla F_{\tilde{w}}(\theta^\star)\|_{(\mathbf{\Sigma}^\star)^{-1}} \ge \frac{\sqrt{\mu}}{4}\,\|\theta_{\tilde{w}} - \theta^\star\|_{\mathbf{\Sigma}^\star}\,.$$

□

Finally, in Section 8.9, when we develop an alternative approach to linear regression based on the robust gradient descent framework, we require a slightly weaker set of distributional assumptions and deterministic implications, which we now state.

Model 4 (distributional regularity for linear regression, gradient descent setting)**.** *Given distributions $\mathcal{D}_X$ and $\mathcal{D}_\delta$ over $\mathbb{R}^d, \mathbb{R}^d$ respectively, the distribution $\mathcal{D}_{Xy}$ over $\mathbb{R}^d \times \mathbb{R}$ is sampled as follows: for an underlying vector $\theta^\star \in \mathbb{R}^d$, independently sample $X \sim \mathcal{D}_X$ and $\delta \sim \mathcal{D}_\delta$, and set $y \leftarrow \langle \theta^\star, X\rangle + \delta$. Further, $\mathcal{D}_X$ and $\mathcal{D}_\delta$ satisfy the following regularity assumptions.*

1. *For $\mathbf{\Sigma}^\star = \mathbb{E}_{\mathcal{D}_X}[XX^\top]$ and $0 < \mu < L$, we have $\mu\mathbf{I} \preceq \mathbf{\Sigma}^\star \preceq L\mathbf{I}$.*
2. *$\mathcal{D}_X$ is $C_{2\to 4}$-hypercontractive for a constant $C_{2\to 4}$.*
3. *$\mathcal{D}_\delta$ is a distribution with mean zero and variance $\le \sigma^2$.*

The main difference between Model 3 and Model 4 is that the latter no longer requires hypercontractive noise. This corresponds to the following deterministic assumption.

Assumption 8 (deterministic regularity for linear regression, gradient descent setting)**.** *Let ϵ be sufficiently small. For every fixed $\theta \in \mathbb{R}^d$, there is a partition $[n] = G_\theta \cup B_\theta$ with $|G_\theta| \ge (1-\epsilon)n$ which satisfies: there is a constant C_{est} such that for $\tilde{w} := \frac{w_{G_\theta}}{\|w_{G_\theta}\|_1}$,*

$$\|\nabla F_{\tilde{w}}(\theta) - \nabla F^\star(\theta)\|_2 \le C_{\text{est}}\sqrt{L\epsilon}\,(\sigma + \|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star})\,, \tag{8.40}$$

$$\left\|\text{Cov}_{\tilde{w}}\left(\{g_i(\theta)\}_{i\in G_\theta}\right)\right\|_{\text{op}} \le C_{\text{est}} L\left(\|\theta - \theta^\star\|^2_{\mathbf{\Sigma}^\star} + \sigma^2\right). \tag{8.41}$$

The main difference between Assumption 7 and Assumption 8 is that the latter provides gradient bounds using a different set G_θ for each θ (as opposed to the former, which uses the same set G for all θ). The upshot is that the corresponding required sample complexity is lower.

Proposition 34 ([134], Lemma 5.5)**.** *Let $\epsilon > 0$ be sufficiently small. Let $\{(X_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{R}$ be an $O(\epsilon)$-corrupted set of samples from a distribution $\mathcal{D}_{Xy}$ as in Model 8. Then if*

$$n = O\left(\frac{d\log(d/\epsilon)}{\epsilon}\right),$$

Assumption 8 holds with probability at least $\frac{9}{10}$.

8.7.4 Regularity assumptions: Lipschitz and smooth stochastic optimization

In Section 8.9, we develop algorithms for the special case of Model 2 when all $\gamma_{y_i}(v) := \gamma(v, y_i)$, as viewed as a function of its first variable, satisfy

$$\left|\gamma'_{y_i}(v)\right| \le 1,\ 0 \le \gamma''_{y_i}(v) \le 1 \text{ for all } v \in \mathbb{R},\ i \in [n].$$

In other words, all γ_{y_i} are 1-smooth and 1-Lipschitz. A canonical example is when all $y_i = \pm 1$ are positive or negative labels, and γ is the logistic loss function, $\gamma(v, y) = \log(1 + \exp(-vy))$. In this setting, we will focus on approximating the (population) regularized optimizer,

$$\theta^\star_{\text{reg}} := \text{argmin}_{\theta\in\mathbb{R}^d}\left\{F^\star(\theta) + \frac{\mu}{2}\left\|\theta\right\|_2^2\right\},\ \text{where } F^\star(\theta) := \mathbb{E}_{f\sim\mathcal{D}_f}\left[f(\theta)\right]. \tag{8.42}$$

Here, $\mu \in \mathbb{R}_{\ge 0}$ controls the amount of regularization, and is used to introduce some amount of strong convexity in the problem. Following Section 8.7.2, the distribution $\mathcal{D}_f$ over sampled functions is directly dependent on a dataset distribution, $\mathcal{D}_{Xy}$, through the relationship in Model 2. Concretely, we make the following regularity assumptions about the distribution $\mathcal{D}_{Xy}$ and its induced $\mathcal{D}_f$.

Model 5 (distributional regularity for smooth GLMs)**.** *$\mathcal{D}_{Xy}$, supported on $\mathbb{R}^d \times \mathbb{R}$, its marginals $\mathcal{D}_X$, $\mathcal{D}_y$, and its induced function distribution $\mathcal{D}_f$, have the following properties.*

1. *Letting the second moment matrix of $\mathcal{D}_X$ be $\mathbf{\Sigma}^\star$ and $0 < L$, we have $\mathbf{\Sigma}^\star \preceq L\mathbf{I}$.*
2. *There is a link function $\gamma : \mathbb{R}^2 \to \mathbb{R}$, such that for all y in the support of $\mathcal{D}_y$, $\gamma_y(v) := \gamma(v, y)$ satisfies $\left|\gamma'_y(v)\right| \le 1$ and $0 \le \gamma''_y(v) \le 1$ for all $v \in \mathbb{R}$.*
3. *The distribution of $f \sim \mathcal{D}_f$ is generated as follows: for $(X, y) \sim \mathcal{D}_{Xy}$, $f(\theta) = \gamma(\langle X, \theta\rangle, y)$.*

In Section 8.10, we further develop algorithms for the special case of Model 2 when all $\gamma_{y_i}(v) := \gamma(v, y_i)$ satisfy only a first-derivative bound,

$$\left|\gamma'_{y_i}(v)\right| \le 1 \text{ for all } v \in \mathbb{R}, i \in [n].$$

In other words, all γ_{y_i} are 1-Lipschitz (but possibly non-smooth). A canonical example is when all $y_i = \pm 1$ are positive or negative labels, and γ is the support vector machine loss function (hinge loss), $\gamma(v, y) = \max(0, 1 - vy)$. In this setting, we will focus on approximating the (population) regularized optimizer of the *Moreau envelope*,

$$\theta^\star_{\text{env}} := \text{argmin}_{\theta \in \mathbb{R}^d} \left\{ F^\star_\gamma(\theta) + \frac{\mu}{2} \|\theta\|_2^2 \right\}, \text{ where } F^\star(\theta) = \mathbb{E}_{f \sim \mathcal{D}_f}[f(\theta)],$$
$$\text{and } F^\star_\lambda(\theta) := \inf_{\theta'} \left\{ F^\star(\theta') + \frac{1}{2\lambda} \|\theta - \theta'\|_2^2 \right\}.$$

The Moreau envelope is extremely well-studied [435], and can be viewed as a smooth approximation to a non-smooth function. We choose to focus on optimizing the Moreau envelope because it is amenable to acceleration techniques, trading off approximation for smoothness.

Model 6 (distributional regularity for Lipschitz GLMs)**.** *$\mathcal{D}_{Xy}$, supported on $\mathbb{R}^d \times \mathbb{R}$, its marginals $\mathcal{D}_X$, $\mathcal{D}_y$, and its induced function distribution $\mathcal{D}_f$, have the following properties.*

1. *Letting the second moment matrix of $\mathcal{D}_X$ be $\mathbf{\Sigma}^\star$ and $0 < L$, we have $\mathbf{\Sigma}^\star \preceq L\mathbf{I}$.*
2. *There is a link function $\gamma : \mathbb{R}^2 \to \mathbb{R}$, such that for all y in the support of $\mathcal{D}_y$, $\gamma_y(v) := \gamma(v, y)$ satisfies $\left|\gamma_y'(v)\right| \leq 1$ for all $v \in \mathbb{R}$.*
3. *The distribution of $f \sim \mathcal{D}_f$ is generated as follows: for $(X, y) \sim \mathcal{D}_{Xy}$, $f(\theta) = \gamma(\langle X, \theta \rangle, y)$.*

In other words, Model 6 is Model 5 without the smoothness assumption. Under the weaker Model 6, [176] showed that we can make the following simplifying deterministic assumptions about our observed dataset (which also extends to Model 5, as it a superset of conditions).

Assumption 9 (deterministic regularity for Lipschitz regression)**.** *The set $\{(X_i, y_i)\}_{i \in [n]} \subset \mathbb{R}^d \times \mathbb{R}$, and the link function $\gamma : \mathbb{R}^2 \to \mathbb{R}$, have the following properties, for $[n] = G \cup B$.*

1. *Letting $w^\star_G := \frac{1}{|G|} \mathbf{1}_G$, $\text{Cov}_{w^\star_G}(\mathbf{X}) \preceq \frac{3}{2} L\mathbf{I}$.*
2. *There is a constant C_{est} such that for all $\theta \in \mathbb{R}^d$ and all saturated $w \in \Delta^n$, letting $\tilde{w} := \frac{w_G}{\|w_G\|_1}$,*

$$\|\nabla F_{\tilde{w}}(\theta) - \nabla F^\star(\theta)\|_2 \leq C_{\text{est}} \sqrt{L\epsilon},$$
$$\left\|\text{Cov}_{\tilde{w}}\left(\{g_i(\theta)\}_{i \in G}\right)\right\|_{\text{op}} \leq C_{\text{est}} L.$$

Proposition 35 ([176], Proposition C.3)**.** *Let $\epsilon > 0$ be sufficiently small. Let $\{(X_i, y_i)\}_{i \in [n]} \subset \mathbb{R}^d \times \mathbb{R}$ be an ϵ-corrupted set of samples from a distribution $\mathcal{D}_{Xy}$ as in Model 6. Then, if*

$$n = O\left(\frac{d \log(d/\epsilon)}{\epsilon}\right),$$

Assumption 9 holds with probability at least $\frac{9}{10}$.

Finally, we make the useful observation that $F^\star$ under Model 6 is also Lipschitz.

Lemma 133. *Under Model 6, $F^\star = \mathbb{E}_{f\sim\mathcal{D}_f}[f]$ is $\sqrt{L}$-Lipschitz.*

Proof. We wish to prove $\|\nabla F^\star(\theta)\|_2 \le \sqrt{L}$ for all $\theta \in \mathbb{R}^d$. By nonnegativity of covariance,

$$\mathbb{E}_{f\sim\mathcal{D}_f}\left[(\nabla f(\theta))(\nabla f(\theta))^\top\right] \succeq \left(\mathbb{E}_{f\sim\mathcal{D}_f}[\nabla f(\theta)]\right)\left(\mathbb{E}_{f\sim\mathcal{D}_f}[\nabla f(\theta)]\right)^\top = (\nabla F^\star(\theta))(\nabla F^\star(\theta))^\top.$$

Since the right-hand side of the above display is rank-one, $\|\nabla F^\star(\theta)\|_2^2$ is at most the largest eigenvalue of the gradient second moment matrix, so it suffices to show the left-hand side is $\preceq L\mathbf{I}$: assuming $f \sim \mathcal{D}_f$ is associated with $(X, y) \sim \mathcal{D}_{Xy}$,

$$\mathbb{E}_{f\sim\mathcal{D}_f}\left[(\nabla f(\theta))(\nabla f(\theta))^\top\right] = \mathbb{E}_{X,y\sim\mathcal{D}_{Xy}}\left[X\left(\gamma'\left(\langle X,\theta\rangle, y\right)\right)^2 X^\top\right] \preceq \mathbf{\Sigma}^\star \preceq L\mathbf{I}.$$

□

8.7.5 Robustly decreasing the covariance operator norm

In this section, we describe a procedure, FastCovFilter, which takes as input a set of vectors $\mathbf{V} := \{v_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$ such that for an (unknown) bipartition $[n] = G \cup B$, it is promised that $\mathbb{E}_{i\sim_{\text{unif}}G}\left[v_i v_i^\top\right]$ is bounded in operator norm by R. Given this promise, FastCovFilter takes as input a set of saturated weights $w \in \Delta^n$ and performs a sequence of safe weight removals to obtain a new saturated $w' \in \Delta^n$, such that $\sum_{i\in[n]} w_i' v_i v_i^\top$ is bounded in operator norm by $O(R)$. Moreover, the procedure runs in nearly-linear time in the description size of $\mathbf{V}$. We formally describe the guarantees of FastCovFilter here as Proposition 36, and defer the proof to Appendix G.6.

Proposition 36. *There is an algorithm,* FastCovFilter *(Algorithm 92), taking inputs $\mathbf{V} := \{v_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$, saturated weights $w \in \Delta^n$ with respect to bipartition $[n] = G \cup B$ with $|B| = \epsilon n$, $\delta \in (0, 1]$, and $R \ge 0$ with the promise that*

$$\left\|\sum_{i\in G}\frac{1}{|G|}v_i v_i^\top\right\|_{\text{op}} \le R.$$

Then, with probability at least $1-\delta$, FastCovFilter *returns saturated $w' \in \Delta^n$ such that*

$$\left\|\sum_{i\in[n]} w_i' v_i v_i^\top\right\|_{\text{op}} \le 5R.$$

The runtime of FastCovFilter *is*

$$O\left(nd\log^3(n)\log\left(\frac{n}{\delta}\right)\right).$$

An algorithm with similar guarantees to FastCovFilter is implicit in the work [194], but we give a self-contained exposition in this work for completeness. Our approach in designing FastCovFilter

is to use a matrix multiplicative weights-based potential function, along with the filtering approach suggested by Lemma 130, to rapidly decrease the quadratic form of the empirical second moment matrix in a number of carefully-chosen directions, and argue this quickly decreases the potential. We remark that this potential-based approach was developed earlier in this chapter (see Section 8.3).

8.8 Linear regression

Throughout this section, we operate under Models 1 and 3, and Assumption 7. Namely, there is a dataset $\{X_i, y_i\}_{i\in[n]}$ and an unknown bipartition $[n] = G \cup B$, such that $\{X_i, y_i\}_{i\in G}$ were draws from a distribution $\mathcal{D}_{Xy}$, and we wish to estimate

$$\theta^\star := \mathrm{argmin}_{\theta\in\mathbb{R}^d} F^\star(\theta), \text{ where } F^\star(\theta) := \mathbb{E}_{X,y\sim\mathcal{D}_{Xy}}\left[\frac{1}{2}\left(\langle X,\theta\rangle - y\right)^2\right].$$

We follow notation (8.35), (8.37) in this section, and define $G^\star \subseteq G$ as the subset given by Assumption 7.2 for the true minimizer $\theta^\star$. We also define $B^\star := [n] \setminus G^\star$, so $B^\star \supseteq B$.

We begin in Section 8.8.1 with a preliminary on filtering under a weaker assumption than the safety condition in Definition 29; in particular, Assumption 7 is not quite compatible with Definition 29 because G' can change based on θ, but will not affect saturation by more than constants. In Section 8.8.2, we then state a general "identifiability proof" showing that controlling certain quantities such as the operator norm of the gradient covariances and near-optimality of a current estimate θ, with respect to some weights $w \in \Delta^n$, suffices to bound closeness of θ and $\theta^\star$. This identifiability proof is motivated by an analogous argument in [53], and will guide our algorithm design. Next, we give a self-contained oracle which rapidly halves the distance to $\theta^\star$ outside of a sufficiently large radius in Section 8.8.3, and analyze the final phase (once this radius is reached) in Section 8.8.4. We put the pieces together to give our main result and full algorithm in Section 8.8.5.

8.8.1 Filtering under $(\epsilon, \frac{\epsilon^2}{\alpha})$-goodness

Throughout this section, we will globally fix a value (for a sufficiently large constant)

$$\alpha = O\left(\max\left(1, \epsilon\kappa\log\frac{R_0}{\sigma}\right)\right). \tag{8.43}$$

In this definition, R_0 is an initial distance bound on $\|\theta_0 - \theta^\star\|_{\mathbf{\Sigma}^\star}$ we will provide to our final algorithm (see the statement of Theorem 56). We operate under Assumption 7 such that our dataset is $(\epsilon, \frac{\epsilon^2}{\alpha})$-good, which inflates the sample complexity of Proposition 33 by a factor depending on α.

At a high level, this technical complication is to ensure that throughout the algorithm we never remove more than 3ϵ mass from G, and that our weights are always ϵ-saturated with respect to $G \cup B$, allowing for inductive application of Assumption 7. Formally, we demonstrate the following

(simple) generalization of Lemma 130, using safety definitions on different sets.

Lemma 134. *Suppose Assumption 7 holds with $r = \frac{\epsilon^2}{\alpha}$. Consider any algorithm which performs the following weight removal.*

1. $w_0 \leftarrow \frac{1}{n} 1_n$
2. *For $0 \le t < T$:*
   (a) $[w_{t+1}]_i \leftarrow \left(1 - \frac{\tau_i}{\tau_{\max}}\right)[w_t]_i$ *for all $i \in [n]$ and $\tau_{\max} := \max_{i \in [n] | w_i \neq 0} \tau_i$, for $\{\tau_i\}_{i \in [n]}$ safe with respect to w_t and a bipartition G'_t, $[n] \setminus G'_t$ where $G'_t \subseteq G$, $|G'_t| \ge (1 - \frac{\epsilon^2}{\alpha})|G|$.*

Suppose the number of distinct sets G'_t throughout the algorithm is bounded by $\frac{\alpha}{2\epsilon}$. Then throughout the algorithm, w_t is ϵ-saturated with respect to the bipartition $G \cup B$.

Proof. Consider some distinct set G'. The proof of Lemma 130 demonstrates that under these assumptions, in every iteration t using G' for weight removal, the amount of mass removed from G' is less than the amount removed from $[n] \setminus G'$. Moreover, the amount of mass removed from G in these iterations can be at most the amount removed from G', plus the weight assigned to the *entire difference* $G \setminus G'$, which is at most a $\frac{\epsilon^2}{\alpha}$ fraction. Similarly, the amount of mass removed from B is at least the amount of mass removed from $[n] \setminus G'$, minus their set difference, which is again at most $\frac{\epsilon^2}{\alpha}$. Combining over all distinct sets, this deviation is at most ϵ. □

It will be straightforward to verify that throughout this section, all weight removals we perform will be of the form in Lemma 134, and that we never perform weight removals with respect to more than $\frac{\alpha}{2\epsilon}$ distinct sets. Hence, we will always assume that any weights w we discuss are ϵ-saturated with respect to the bipartition $G \cup B$, and thus has $\|w_G\|_1 \ge 1 - 3\epsilon$, allowing for application of Assumption 7. Finally, we remark we sometimes will apply Assumption 7.1 with weight vectors $w_{G^\star}$ instead of w_G; since their difference is $O(\epsilon^2) < \epsilon$, this is a correct application for any $\|w_G\|_1 \ge 1 - 3\epsilon$.

8.8.2 Identifiability proof for linear regression

In this section, we give an identifiability proof similar to that appearing in [53] which demonstrates, for a given weight-parameter estimate pair $(w, \theta) \in \Delta^n \times \mathbb{R}^d$, verifiable technical conditions on this pair which certify a bound on $\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star}$. In short, Proposition 37 will show that if both of the quantities

$$\|\nabla F_w(\theta)\|_{(\mathbf{\Sigma}^\star)^{-1}}, \ \left\|\mathrm{Cov}_w\left(\{g_i(\theta)\}_{i \in [n]}\right)\right\|_{\mathrm{op}} \tag{8.44}$$

are simultaneously controlled, then we obtain a distance bound to $\theta_{G^\star}$.

Proposition 37. *Let $w \in \Delta^n$ be ϵ-saturated with respect to the bipartition $[n] = G \cup B$, and let $\theta \in \mathbb{R}^d$. Assuming $\epsilon\kappa$ is sufficiently small, there is a universal constant C_{id} such that*

$$\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} \le C_{\mathrm{id}} \left(\sqrt{\epsilon} \left(\sigma\sqrt{\kappa} + \sqrt{\frac{\left\| \mathrm{Cov}_w \left(\{g_i(\theta)\}_{i\in[n]} \right) \right\|_{\mathrm{op}}}{\mu}} \right) + \|\nabla F_w(\theta)\|_{(\mathbf{\Sigma}^\star)^{-1}} \right).$$

Proof. Let G' be the set promised by Assumption 7.2 for the point θ. Throughout this proof, we define $G_\theta := G' \cap G^\star$, and we let $w_\theta^\star := \frac{1}{|G_\theta|}\mathbf{1}_{G_\theta}$ be the uniform weights on G_θ. Finally, let $\hat{\theta}$ minimize $F_{w_\theta^\star}$. By applying Lemma 132 on the weights $w_\theta^\star$, we have that $\|\hat{\theta} - \theta^\star\|_{\mathbf{\Sigma}^\star} = O(\sigma\sqrt{\kappa\epsilon})$. Thus, in the remainder of the proof we focus on bounding $\|\theta - \hat{\theta}\|_{\mathbf{\Sigma}^\star}$ by the required quantity.

Let $\mathcal{C}(w_\theta^\star, \tilde{w})$ supported on $[n] \times [n]$ be an optimal *coupling* between $i \sim w_\theta^\star$ and $j \sim \tilde{w}$, where $\tilde{w} := \frac{w}{\|w\|_1}$ is the distribution proportional to w; we denote the coupling as $\mathcal{C}$ for short. For a pair $(i, j) \in [n] \times [n]$ sampled from $\mathcal{C}$, we let $\mathbf{1}_{i=j}$ be the indicator of the event $i = j$, and similarly define $\mathbf{1}_{i\neq j}$. From the total variation characterization of coupling, we have by Lemma 131 that

$$\mathbb{E}_{i,j\sim\mathcal{C}} [\mathbf{1}_{i\neq j}] = \|\tilde{w} - w_\theta^\star\|_1 \le 9\epsilon.$$

Here we used that $\|w_\theta^\star - w_G^\star\|_1 = O(\epsilon^2)$, where $w_G^\star$ is the uniform distribution on G, as in Lemma 131. Now, let $v = \theta - \hat{\theta}$. We have by Assumption 7.1 that

$$\mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, X_i \rangle \left\langle X_i, \theta - \hat{\theta} \right\rangle \right] = \left\langle \theta - \hat{\theta}, \mathbb{E}_{i\sim w_\theta^\star} \left[X_i X_i^\top \right] (\theta - \hat{\theta}) \right\rangle \ge \frac{1}{2} \left\| \theta - \hat{\theta} \right\|_{\mathbf{\Sigma}^\star}^2. \tag{8.45}$$

On the other hand,

$$\begin{aligned}
\mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, X_i \rangle \left\langle X_i, \theta - \hat{\theta} \right\rangle \right] &= \mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, X_i \rangle \left(\langle X_i, \theta \rangle - y_i \right) \right] + \mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, X_i \rangle \left(y_i - \left\langle X_i, \hat{\theta} \right\rangle \right) \right] \\
&= \mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, g_i(\theta) \rangle \right] - \mathbb{E}_{i\sim w_\theta^\star} \left[\left\langle v, g_i(\hat{\theta}) \right\rangle \right] = \mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, g_i(\theta) \rangle \right].
\end{aligned}$$

Here, we used that $\mathbb{E}_{i\sim w_\theta^\star} \left[g_i(\hat{\theta}) \right] = \nabla F_{w_\theta^\star}(\hat{\theta}) = 0$ by definition of $\hat{\theta}$. Continuing,

$$\begin{aligned}
\mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, X_i \rangle \left\langle X_i, \theta - \hat{\theta} \right\rangle \right] &= \mathbb{E}_{i,j\sim\mathcal{C}} \left[\langle v, g_i(\theta) \rangle \mathbf{1}_{i=j} \right] + \mathbb{E}_{i,j\sim\mathcal{C}} \left[\langle v, g_i(\theta) \rangle \mathbf{1}_{i\neq j} \right] \\
&= \mathbb{E}_{j\sim\tilde{w}} \left[\langle v, g_j(\theta) \rangle \right] - \mathbb{E}_{i,j\sim\mathcal{C}} \left[\langle v, g_j(\theta) \rangle \mathbf{1}_{i\neq j} \right] + \mathbb{E}_{i,j\sim\mathcal{C}} \left[\langle v, g_i(\theta) \rangle \mathbf{1}_{i\neq j} \right] \\
&\le \langle v, \nabla F_{\tilde{w}}(\theta) \rangle + 3\sqrt{\epsilon} \left(\mathbb{E}_{j\sim\tilde{w}} \left[\langle v, g_j(\theta) \rangle^2 \right]^{\frac{1}{2}} + \mathbb{E}_{i\sim w_\theta^\star} \left[\langle v, g_i(\theta) \rangle^2 \right]^{\frac{1}{2}} \right).
\end{aligned} \tag{8.46}$$

In the last line, we used Cauchy-Schwarz and $\mathbb{E}_{i,j\sim\mathcal{C}}[\mathbf{1}_{i\neq j}^2] \le 9\epsilon$ to deal with the second and third

terms. To bound the term corresponding to $i \sim w_\theta^\star$,

$$\mathbb{E}_{i\sim w_\theta^\star}\left[\langle v, g_i(\theta)\rangle^2\right] \le \|v\|_2^2 \left\|\mathrm{Cov}_{w_\theta^\star}\left(\{g_i(\theta)\}_{i\in[n]}\right)\right\|_{\mathrm{op}} \le 1.1 C_{\mathrm{est}}\kappa \|v\|_{\mathbf{\Sigma}^\star}^2 \left(\|\theta-\theta^\star\|_{\mathbf{\Sigma}^\star}^2 + \sigma^2\right).$$

The first inequality is by definition of $\|\cdot\|_{\mathrm{op}}$, and the second used $\|v\|_2^2 \ge \frac{1}{\mu}\|v\|_{\mathbf{\Sigma}^\star}^2$ and Assumption 7.2, since G_θ is a subset of the set G' promised by Assumption 7.2 (where we adjusted by a constant for normalization). We similarly arrive at the bound

$$\mathbb{E}_{j\sim\tilde{w}}\left[\langle v, g_j(\theta)\rangle^2\right] \le \frac{\|v\|_{\mathbf{\Sigma}^\star}^2}{\mu}\left\|\mathrm{Cov}_w\left(\{g_i(\theta)\}_{i\in[n]}\right)\right\|_{\mathrm{op}}.$$

Now plugging the above displays into (8.46) and combining with (8.45), as well as using $\|\theta-\theta^\star\|_{\mathbf{\Sigma}^\star}^2 + \sigma^2 = O(\|\theta-\hat{\theta}\|_{\mathbf{\Sigma}^\star}^2 + \sigma^2)$ by the triangle inequality and our earlier bound on $\|\theta^\star-\hat{\theta}\|_{\mathbf{\Sigma}^\star}$,

$$\begin{aligned}\left\|\theta-\hat{\theta}\right\|_{\mathbf{\Sigma}^\star}^2 \le \left\|\theta-\hat{\theta}\right\|_{\mathbf{\Sigma}^\star}\|\nabla F_{\tilde{w}}(\theta)\|_{(\mathbf{\Sigma}^\star)^{-1}} + O\left(\sqrt{\epsilon\kappa}\right)\left(\sigma\left\|\theta-\hat{\theta}\right\|_{\mathbf{\Sigma}^\star} + \left\|\theta-\hat{\theta}\right\|_{\mathbf{\Sigma}^\star}^2\right)\\ + O\left(\sqrt{\epsilon}\right)\left(\left\|\theta-\hat{\theta}\right\|_{\mathbf{\Sigma}^\star}\sqrt{\frac{\left\|\mathrm{Cov}_w\left(\{g_i(\theta)\}_{i\in[n]}\right)\right\|_{\mathrm{op}}}{\mu}}\right).\end{aligned}$$

In the first line, we used Cauchy-Schwarz to bound the term $\langle v, \nabla F_{\tilde{w}}(\theta)\rangle$. Dividing through by $\|\theta-\hat{\theta}\|_{\mathbf{\Sigma}^\star}$, and using that $\epsilon\kappa$ is sufficiently small, yields the conclusion. □

Proposition 37 suggests a natural approach. On the one hand, if θ is an (approximate) minimizer to F_w, the first quantity in (8.44) will be small. On the other hand, for a fixed $\theta \in \mathbb{R}^d$, we can filter on w using our subroutine FastCovFilter until the second quantity in (8.44) is small. The main challenge is accomplishing both bounds simultaneously. To this end, we show that the number of times we have to repeat this process of filtering and then computing an approximate minimizer is bounded, using F_w as a potential function; by preprocessing F_w so that is smooth at all points, if θ is an approximate minimizer for the F_w attained *after* filtering on gradients at θ, then we can exit the subroutine. Otherwise, we argue we make substantial function progress by calling an empirical risk minimizer to terminate quickly. We make this strategy formal in the following sections.

8.8.3 Halving the distance to $\theta^\star$

In this section, we design a procedure, HalfRadiusLinReg, with the following guarantee. Suppose that we have ϵ-saturated $\bar{w} \in \Delta^n$ and a parameter $\bar{\theta} \in \mathbb{R}^d$, as well as scalar R with the promise

$$\left\|\bar{\theta}-\theta^\star\right\|_{\mathbf{\Sigma}^\star} \le R.$$

The goal of HalfRadiusLinReg is to return a new θ with $\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} \le \frac{1}{2}R$; we require that $R = \Omega(\sigma)$ for a sufficiently large constant in this section. We do so by using saturated weights to guide a potential analysis, crucially using Proposition 37. Before stating HalfRadiusLinReg, we require two additional helper tools. The first is an approximate optimization procedure.

Definition 32. *We call $\mathcal{O}_{\mathrm{ERM}}$ a γ-approximate ERM oracle if on input $F : \mathbb{R}^d \to \mathbb{R}$ it returns a point $\hat{\theta}$ such that $F(\hat{\theta}) - F(\theta_F^\star) \le \gamma$, for $\theta_F^\star := \mathrm{argmin}_{\theta \in \mathbb{R}^d} F(\theta)$.*

The second controls the initial error, which we use to yield distance bounds via strong convexity.

Lemma 135. *There is an algorithm, FunctionFilter, which takes as input ϵ-saturated $w \in \Delta^n$, $\theta \in \mathbb{R}^d$, and $R \ge \|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star}$, and produces ϵ-saturated $w' \in \Delta^n$ such that $F_{w'}(\theta) \le 2C_{\mathrm{ub}}(\sigma^2 + R^2)$, in time $O(nd + n \log \frac{D}{R^2})$, where D is a bound on the largest $\frac{1}{2}(\langle X_i, \theta\rangle - y_i)^2$ for any nonzero w_i.*

Proof. Define $\tau_i := f_i(\theta) = \frac{1}{2}(\langle X_i, \theta\rangle - y_i)^2$. Assumption 7.1 shows that $F_{w_{G^\star}}$ is $\frac{3}{2}$-smooth in the $\mathbf{\Sigma}^\star$ norm, since its Hessian is exactly $\|w_{G^\star}\|_1 \operatorname{Cov}_{w_{G^\star}}(\mathbf{X}) \preceq \frac{3}{2}\mathbf{\Sigma}^\star$. Moreover, letting $\hat{\theta}$ minimize $F_{w_{G^\star}}$, by Lemma 132, the triangle inequality and the assumed bound R,

$$\left\|\theta - \hat{\theta}\right\|_{\mathbf{\Sigma}^\star} \le R + \left\|\theta^\star - \hat{\theta}\right\|_{\mathbf{\Sigma}^\star} \le R + \sigma.$$

Hence, assuming C_{ub} is large enough and $R \ge \sigma$, smoothness demonstrates

$$\sum_{i \in G^\star} w_i \tau_i = F_{w_{G^\star}}(\theta) \le F_{w_{G^\star}}(\hat{\theta}) + \frac{3}{4}\left\|\theta - \hat{\theta}\right\|_{\mathbf{\Sigma}^\star}^2 \le C_{\mathrm{ub}}(\sigma^2 + R^2),$$

Next, letting $\tau_{\max} := \max_{i \in [n] | w_i \ne 0} \tau_i$ and $K \in \mathbb{N} \cup \{0\}$ be smallest such that

$$\sum_{i \in [n]} \left(1 - \frac{\tau_i}{\tau_{\max}}\right)^K w_i \tau_i \le 2C_{\mathrm{ub}}(\sigma^2 + R^2),$$

each of the first K weight removals according to the scores τ are safe with respect to $G^\star \cup B^\star$, and Lemma 134 implies we can output ϵ-saturated $w_i' = \left(1 - \frac{\tau_i}{\tau_{\max}}\right)^K w_i$ entrywise. It remains to binary search for K; given access to the scores τ, checking if a given K passes the above display takes $O(n)$ time. We can upper bound K by the following inequality:

$$\sum_{i \in [n]} \left(1 - \frac{\tau_i}{\tau_{\max}}\right)^K w_i \tau_i \le \sum_{i \in [n]} \exp\left(-\frac{K\tau_i}{\tau_{\max}}\right) w_i \tau_i \le \frac{1}{eK} \sum_{i \in [n]} w_i \tau_{\max} \le \frac{\tau_{\max}}{K}.$$

Here the second inequality used $x \exp(-Cx) \le \frac{1}{eC}$ for all nonnegative x, C, where we chose $C = \frac{K}{\tau_{\max}}$ and $x = \tau_i$. Hence, $K = O(\frac{D}{R^2})$. The runtime follows as computing scores takes time $O(nd)$. □

We remark that every time we use FunctionFilter is a weight removal of the form in Lemma 134, which is safe with respect to the bipartition $G^\star \cup B^\star$. This accounts for one distinct set throughout.

Algorithm 46: HalfRadiusLinReg$(\mathbf{X}, y, \bar{w}, \bar{\theta}, R, D, \mathcal{O}_{\mathrm{ERM}}, \delta)$

Input: Dataset $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$, $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$, satisfying Assumption 7, ϵ-saturated $\bar{w} \in \Delta^n$ with respect to bipartition $[n] = G \cup B$ such that $\mathbf{X}^\top \mathbf{diag}\,(\bar{w})\,\mathbf{X} \preceq 8L\mathbf{I}$, $\bar{\theta} \in \mathbb{R}^d$ with $\left\|\bar{\theta} - \theta^\star\right\|_{\mathbf{\Sigma}^\star} \le R$, $\delta \in (0,1)$, $O(\frac{\sigma^2}{\kappa})$-approximate ERM oracle $\mathcal{O}_{\mathrm{ERM}}$,

$$D \ge \max_{i\in[n]} \frac{1}{2}(\langle X_i, \bar{\theta}\rangle - y_i)^2. \tag{8.47}$$

;

Output: With probability $\ge 1-\delta$, saturated w with respect to $G \cup B$, and θ with

$$\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} \le \frac{1}{2}R.$$

;

$t \leftarrow 0$, $w^{(0)} \leftarrow$ FunctionFilter$(w, \bar{\theta}, R, D)$, $\Delta_0 \leftarrow \infty$;

while $\Delta_t > \frac{R^2}{512\kappa C_{\mathrm{id}}^2}$ **do**

  $\theta^{(t)} \leftarrow \mathcal{O}_{\mathrm{ERM}}(F_{w^{(t)}})$;

  $w^{(t+1)} \leftarrow$ FastCovFilter $\left(\left\{g_i\left(\theta^{(t)}\right)\right\}_{i\in[n]}, w^{(t)}, \frac{\delta}{O(\kappa)}, 40C_{\mathrm{est}}C_{\mathrm{ub}}LR^2\right)$;

  $\Delta_{t+1} \leftarrow F_{w^{(t+1)}}\left(\theta^{(t)}\right) - F_{w^{(t+1)}}\left(\theta^{(t+1)}\right)$;

  $t \leftarrow t+1$;

Return: $\left(w^{(t)}, \theta^{(t-1)}\right)$

Lemma 136. HalfRadiusLinReg *is correct, i.e. if its preconditions are met, it successfully returns* (w, θ) *such that* $w \in \Delta^n$ *is* ϵ*-saturated and* $\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} \le \frac{1}{2}R$*. It runs in* $O(\kappa)$ *calls to* $\mathcal{O}_{\mathrm{ERM}}$*, plus*

$$O\left(n\log\left(\frac{D}{R^2}\right) + nd\kappa\log^3(n)\log\left(\frac{n\kappa}{\delta}\right)\right) \text{ additional time.}$$

Proof. We discuss correctness and runtime separately.

Correctness. The first step of our correctness proof is to show that throughout the algorithm,

$$\left\|\theta^{(t)} - \theta^\star\right\|_{\mathbf{\Sigma}^\star} \le 6\sqrt{C_{\mathrm{ub}}}R. \tag{8.48}$$

To see this, consider iteration t and suppose $w^{(t)}$ is ϵ-saturated. Let $G' \subseteq G$ be the set promised by Assumption 7.2 for the pair $(w^{(t)}, \theta^{(t)})$, and let $G_t = G' \cap G^\star$. At the beginning of the algorithm we applied FunctionFilter (see Lemma 135), and the minimum value of F_w is monotone nonincreasing as w is decreasing, and $\mathcal{O}_{\mathrm{ERM}}$ only decreases function value (else Line 4 would fail), so since $R \ge \sigma$,

$$F_{w^{(t)}_{G_t}}\left(\theta^{(t)}\right) \le F_{w^{(t)}}\left(\theta^{(t)}\right) \le 4C_{\mathrm{ub}}R^2.$$

On the other hand, $F_{w^{(t)}_{G_t}}$ is $\frac{1}{3}$-strongly convex in the $\mathbf{\Sigma}^\star$ norm by Assumption 7.1 (adjusting for the

normalization factor), and hence letting $\theta_t^\star$ be the minimizer of $F_{w_{G_t}^{(t)}}$, we have

$$4C_{\mathrm{ub}}R^2 \geq F_{w_{G_t}^{(t)}}\left(\theta^{(t)}\right) \geq F_{w_{G_t}^{(t)}}\left(\theta^{(t)}\right) - F_{w_{G_t}^{(t)}}\left(\theta_t^\star\right) \geq \frac{1}{6}\left\|\theta^{(t)} - \theta_t^\star\right\|_{\mathbf{\Sigma}^\star}^2 \implies \left\|\theta^{(t)} - \theta_t^\star\right\|_{\mathbf{\Sigma}^\star} \leq 5\sqrt{C_{\mathrm{ub}}}R.$$

Finally, by using Lemma 132 on $\theta_t^\star$, we have $\|\theta_t^\star - \theta^\star\|_{\mathbf{\Sigma}^\star} \leq 4C_{\mathrm{est}}\sigma\sqrt{\kappa\epsilon} \leq R$. From this and the above display, the triangle inequality yields (8.48). Now, (8.48) with Assumption 7.2 shows that for all t,

$$\left\|\mathrm{Cov}_{\frac{1}{|G_t|}\mathbf{1}_{G_t}}\left(\left\{g_i(\theta^{(t)})\right\}_{i\in G_t}\right)\right\|_{\mathrm{op}} \leq 40C_{\mathrm{est}}C_{\mathrm{ub}}LR^2.$$

We argue later in this proof that there are at most $O(\kappa)$ loops throughout the algorithm. This shows all calls to FastCovFilter are safe with respect to the bipartition G_t, $[n] \setminus G_t$ (adjusting the definition of ϵ by a constant in Proposition 36), and thus taking a union bound the algorithm succeeds with probability at least $1 - \delta$. Condition on this for the remainder of the proof.

The success of all calls to FastCovFilter implies that in every iteration t (following Proposition 36),

$$\left\|\mathrm{Cov}_{w^{(t+1)}}\left(\left\{g_i(\theta^{(t)})\right\}_{i\in[n]}\right)\right\|_{\mathrm{op}} = O\left(LR^2\right). \tag{8.49}$$

Next, in any iteration where $\Delta_{t+1} \leq \frac{R^2}{512\kappa C_{\mathrm{id}}^2}$, we claim that

$$\left\|\nabla F_{w^{(t+1)}}\left(\theta^{(t)}\right)\right\|_{(\mathbf{\Sigma}^\star)^{-1}} \leq \frac{R}{4C_{\mathrm{id}}}. \tag{8.50}$$

This is because $F_{w^{(t+1)}}$ is 8κ-smooth in the $\mathbf{\Sigma}^\star$ norm, since $\nabla^2 F_{w^{(t+1)}} = \mathbf{X}^\top \mathbf{diag}\left(w^{(t+1)}\right)\mathbf{X} \preceq 8L\mathbf{I}$ by assumption, and $8\kappa\mathbf{\Sigma}^\star \succeq 8L\mathbf{I}$ by assumption, so by the guarantees of $\mathcal{O}_{\mathrm{ERM}}$,

$$\begin{aligned}\frac{1}{16\kappa}\left\|\nabla F_{w^{(t+1)}}\left(\theta^{(t)}\right)\right\|_{(\mathbf{\Sigma}^\star)^{-1}}^2 &\leq F_{w^{(t+1)}}\left(\theta^{(t)}\right) - \min_{\theta\in\mathbb{R}^d} F_{w^{(t+1)}}(\theta) \\ &\leq \Delta_{t+1} + O\left(\frac{\sigma^2}{\kappa}\right) \leq \frac{R^2}{256\kappa C_{\mathrm{id}}^2}.\end{aligned}$$

Rearranging indeed yields (8.50). Now by combining (8.49) and (8.50) in Proposition 37, we see that if $\Delta_{t+1} \leq \frac{R^2}{512\kappa C_{\mathrm{id}}^2}$ in an iteration, we obtain the desired $\left\|\theta^{(t)} - \theta^\star\right\|_{\mathbf{\Sigma}^\star} \leq \frac{1}{2}R$.

Runtime. We first observe that the loop in Lines 4-9 of Algorithm 46 can only run $O(\kappa)$ times. This is because FunctionFilter decreases the initial function value until it is $O(R^2)$, and every loop decreases the function value by $\Omega(\frac{R^2}{\kappa})$. Hence, the cost of the whole algorithm is one call to FunctionFilter, and $O(\kappa)$ calls to $\mathcal{O}_{\mathrm{ERM}}$, FastCovFilter, and two function value computations, which fit into the allotted runtime budget by Lemma 135 and Proposition 36.

□

Finally, we remark that throughout Algorithm 46, we only filtered weight based on FunctionFilter

and FastCovFilter, with respect to at most $O(\kappa)$ distinct sets (in the manner described by Lemma 134): the sets $\{G_t\}$ used in correctness calls to FastCovFilter, and the set $G^\star$ for the one call to FunctionFilter.

8.8.4 Last phase analysis

In this section, we give a slight variant of Algorithm 46 which applies when $R \le C_{\mathrm{lp}}\sigma$ for a universal constant C_{lp}. It will have a somewhat more stringent termination condition, because we require the gradient term in Proposition 37 to be $O(\sigma\sqrt{\epsilon\kappa})$, but otherwise is identical.

Algorithm 47: LastPhase$(\mathbf{X}, y, \bar{w}, \bar{\theta}, D, \mathcal{O}_{\mathrm{ERM}}, \delta)$

Input: Dataset $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$, $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$, satisfying Assumption 7, ϵ-saturated $\bar{w} \in \Delta^n$ with respect to bipartition $[n] = G \cup B$ such that $\mathbf{X}^\top \mathbf{diag}(\bar{w})\,\mathbf{X} \preceq 8L\mathbf{I}$, $\bar{\theta} \in \mathbb{R}^d$ with $\left\|\bar{\theta} - \theta^\star\right\|_{\mathbf{\Sigma}^\star} \le C_{\mathrm{lp}}\sigma$, $\delta \in (0,1)$, $O(\sigma^2\epsilon)$-approximate ERM oracle $\mathcal{O}_{\mathrm{ERM}}$,

$$D \ge \max_{i\in[n]} \frac{1}{2}(\langle X_i, \bar{\theta}\rangle - y_i)^2. \tag{8.51}$$

;

Output: With probability $\ge 1-\delta$, saturated w with respect to $G \cup B$, and θ with

$$\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} = O\left(\sigma\sqrt{\kappa\epsilon}\right).$$

;

$t \leftarrow 0$, $w^{(0)} \leftarrow$ FunctionFilter$(w, \bar{\theta}, C_{\mathrm{lp}}\sigma, D)$, $\Delta_0 \leftarrow \infty$;

while $\Delta_t > \sigma^2\epsilon$ **do**

  $\theta^{(t)} \leftarrow \mathcal{O}_{\mathrm{ERM}}(F_{w^{(t)}})$;

  $w^{(t+1)} \leftarrow$ FastCovFilter$\left(\left\{g_i\left(\theta^{(t)}\right)\right\}_{i\in[n]}, w^{(t)}, O(\delta\epsilon), 40C_{\mathrm{est}}C_{\mathrm{ub}}C_{\mathrm{lp}}^2 L\sigma^2\right)$;

  $\Delta_{t+1} \leftarrow F_{w^{(t+1)}}\left(\theta^{(t)}\right) - F_{w^{(t+1)}}\left(\theta^{(t+1)}\right)$;

  $t \leftarrow t+1$;

Return: $\left(w^{(t)}, \theta^{(t-1)}\right)$;

Lemma 137. LastPhase *is correct, i.e. if its preconditions are met, it successfully returns (w,θ) such that $w \in \Delta^n$ is ϵ-saturated and $\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} = O(\sigma\sqrt{\kappa\epsilon})$. It runs in $O(\frac{1}{\epsilon})$ calls to $\mathcal{O}_{\mathrm{ERM}}$, plus*

$$O\left(n\log\left(\frac{D}{R^2}\right) + \frac{nd}{\epsilon}\log^3(n)\log\left(\frac{n}{\delta\epsilon}\right)\right) \text{ additional time.}$$

Proof. On the correctness side, the analysis is nearly identical to Lemma 136; the same logic applies to yield an analogous bound to (8.48), which shows that all iterates are within $O(\sigma)$ from the minimizer, so all calls to FastCovFilter are correct. This implies that (analogous to (8.49)) the gradient operator norm is always bounded by $O(L\sigma^2)$. Similarly, since the threshold for termination

is when the function decrease is $\sigma^2\epsilon$, we have that on the terminating iteration,

$$\frac{1}{16\kappa}\left\|\nabla F_{w^{(t+1)}}\left(\theta^{(t)}\right)\right\|^2_{(\mathbf{\Sigma}^\star)^{-1}} = O(\sigma^2\epsilon),$$

and combining this with the operator norm bound in Proposition 37 yields the conclusion. On the runtime side, the analysis is the same as Lemma 136, except that there are now $O(\frac{1}{\epsilon})$ iterations. □

Again, we remark here that throughout Algorithm 47, we filtered weight with respect to at most $O(\epsilon^{-1})$ distinct sets (in the manner described by Lemma 134).

8.8.5 Full algorithm

Before we give our full algorithm, we require some preliminary pruning procedures on the dataset.

Lemma 138. *Under Assumption 7, for all $i \in G$, $\|X_i\|_2 \le \sqrt{2Ln}$.*

Proof. Suppose otherwise for some $i \in G$. Then, since $\mathrm{Cov}_{w_G}(\mathbf{X}) \succeq \frac{1}{|G|}X_iX_i^\top \succeq \frac{1}{n}X_iX_i^\top$, $\mathrm{Cov}_{w_G}(\mathbf{X})$ has an eigenvalue larger than $\frac{3}{2}L$ (certified by $\frac{X_i}{\|X_i\|_2}$), contradicting Assumption 7.1. □

We next give a bound on D required by Algorithms 46 and 47, assuming a bound on $\left\|\bar{\theta}-\theta^\star\right\|_{\mathbf{\Sigma}^\star}$.

Lemma 139. *Suppose all $\{X_i\}_{i\in[n]}$ satisfy $\|X_i\|_2 \le \sqrt{2Ln}$, and we have a bound $\left\|\bar{\theta}-\theta^\star\right\|_{\mathbf{\Sigma}^\star} \le R$. Under Assumption 7, it suffices to set D in* HalfRadiusLinReg *or* LastPhase *to*

$$D = 2nC_{\mathrm{ub}}\sigma^2 + 2\kappa nR^2. \tag{8.52}$$

Proof. First, under Assumption 7 we have that for all $i \in G$,

$$|\langle X_i, \theta^\star\rangle - y_i| \le \sqrt{2nC_{\mathrm{ub}}\sigma^2}.$$

Applying Cauchy-Schwarz, we have that for all $i \in G$, since $\left\|\bar{\theta}-\theta^\star\right\|_2 \le \frac{1}{\sqrt{\mu}}\left\|\bar{\theta}-\theta^\star\right\|_{\mathbf{\Sigma}^\star} \le \frac{R}{\sqrt{\mu}}$,

$$\left|\left\langle X_i, \bar{\theta}\right\rangle - y_i\right| \le \sqrt{2nC_{\mathrm{ub}}\sigma^2} + \|X_i\|_2\left\|\bar{\theta}-\theta^\star\right\|_2 \le \sqrt{2nC_{\mathrm{ub}}\sigma^2} + \sqrt{2\kappa n}R.$$

□

Finally, we give our full algorithm for regression, FastRegression, below.

Theorem 56. *In Models 1 and 3, under Assumption 7 with $r = \frac{\epsilon^2}{\alpha}$ for α as in (8.43), supposing $\epsilon\kappa$ is sufficiently small, given $\theta_0 \in \mathbb{R}^d$ and $\|\theta_0-\theta^\star\|_{\mathbf{\Sigma}^\star} \le R_0$,* FastRegression *returns θ with $\|\theta-\theta^\star\|_{\mathbf{\Sigma}^\star} = O(\sigma\sqrt{\kappa\epsilon})$ with probability at least $1-\delta$. The algorithm runs in $O\left(\frac{1}{\epsilon}+\kappa\log\frac{R_0}{\sigma}\right)$ calls to a $O(\sigma^2\epsilon)$-approximate ERM oracle, and*

$$O\left(\left(nd\log^3(n)\log\left(\frac{nR_0}{\sigma\delta\epsilon}\right)\right)\left(\frac{1}{\epsilon}+\kappa\log\frac{R_0}{\sigma}\right)\right) \text{ additional time.}$$

Algorithm 48: FastRegression($\mathbf{X}, y, \theta_0, R_0, \mathcal{O}_{\mathrm{ERM}}, \delta$)

Input: Dataset $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$, $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$, satisfying Models 1 and 3, and satisfying Assumptions 7, $\theta_0 \in \mathbb{R}^d$ with $\|\theta_0 - \theta^\star\|_{\mathbf{\Sigma}^\star} \le R_0$, $\delta \in (0,1)$, $O(\sigma^2\epsilon)$-approximate ERM oracle $\mathcal{O}_{\mathrm{ERM}}$.;

Output: With probability $\ge 1-\delta$, θ with

$$\|\theta - \theta^\star\|_{\mathbf{\Sigma}^\star} = O\left(\sigma\sqrt{\kappa\epsilon}\right).$$

;

Remove all (X_i, y_i) with $\|X_i\|_2 > \sqrt{2nL}$, $n \leftarrow$ new dataset size;

$w \leftarrow$ FastCovFilter($\mathbf{X}, \frac{1}{n}1_{,1}, \frac{\delta}{3}, \frac{3}{2}L, 2nL$), $R \leftarrow R_0 + 4C_{\mathrm{est}}\sigma\sqrt{\kappa\epsilon}$, $\theta \leftarrow \theta_0$;

$T \leftarrow O(\log\frac{R_0}{\sigma})$ for a sufficiently large constant;

while $R > C_{\mathrm{lp}}\sigma$ **do**

  $D \leftarrow$ value in (8.52) with current setting of R ;

  Remove all (X_i, y_i) not satisfying bound (8.47), $n \leftarrow$ new dataset size;

  $(w, \theta) \leftarrow$ HalfRadiusLinReg($\mathbf{X}, y, w, \theta, R, D, \mathcal{O}_{\mathrm{ERM}}, \frac{\delta}{3T}$);

  $R \leftarrow \frac{1}{2}R$;

$D \leftarrow$ value in (8.52) with current setting of R ;

Remove all (X_i, y_i) not satisfying bound (8.47), $n \leftarrow$ new dataset size;

$(w, \theta) \leftarrow$ HalfRadiusLinReg($\mathbf{X}, y, w, \theta, D, \mathcal{O}_{\mathrm{ERM}}, \frac{\delta}{3}$);

Return: θ;

Proof. First, correctness of Lines 3, 8, and 13 of the algorithm follow from Lemmas 138 and 139. Also, Line 4 ensures that throughout the algorithm we have $\mathbf{X}^\top \mathbf{diag}\,(w)\,\mathbf{X} \preceq 8L\mathbf{I}$, so the preconditions of HalfRadiusLinReg and LastPhase are met. Finally, the initial setting of R is correct by Lemma 132 and the assumed bound R_0. The correctness and runtime then follow from applying Lemma 136 T times and Lemma 137 once. The failure probability comes from union bounding over the one call to FastCovFilter, the T calls to HalfRadiusLinReg, and the one call to LastPhase.

Finally, for completeness we check that the promise of Section 8.8.1 is kept by Algorithm 48. There are at most $\log(\frac{R_0}{\sigma})$ calls to HalfRadiusLinReg, and one call to LastPhase. Combined, this accounts for at most $O(\frac{1}{\epsilon} + \kappa\log\frac{R_0}{\sigma})$ distinct sets we filtered with respect to, in the manner described by Lemma 134. For a sufficiently large α in (8.43), this is indeed at most $\frac{\alpha}{2\epsilon}$ distinct sets. □

We conclude this section by noting that the guarantees of Proposition 33 imply the sample complexity required for Algorithm 48 to succeed with probability at least $\frac{9}{10} - \delta$ is $n = \widetilde{O}(\frac{d}{\epsilon^4} + \frac{d^2}{\epsilon^3})$.

8.9 Robust acceleration

In this section, we give a general-purpose algorithm for solving statistical optimization problems with a finite condition number κ under the strong contamination model. We study the following abstract problem: we wish to minimize a function $F : \mathbb{R}^d \to \mathbb{R}$ which is L-smooth and μ-strongly

convex with minimizer $\theta_F^\star$, but we only have black-box access to F through a *noisy gradient oracle* $\mathcal{O}_{\mathrm{ng}}$. In particular, we can query $\mathcal{O}_{\mathrm{ng}}$ at any point $\theta \in \mathbb{R}^d$ with a parameter $R \geq \|\theta - \theta_F^\star\|_2$ and receive $G(\theta)$ such that for a universal constant C_{ng},

$$\|G(\theta) - \nabla F(\theta)\|_2 \leq C_{\mathrm{ng}} \left(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R\right).$$

Notably, our algorithm is *accelerated*, running in a number of iterations depending on $\sqrt{\kappa}$ rather than κ. It applies to both the regression setting of Section 8.7.3 and the smooth stochastic optimization setting of Section 8.7.4. We demonstrate in Section 8.9.1 how to build a noisy gradient oracle for regression and smooth stochastic optimization settings. We then build in Section 8.9.2 a simple subroutine based on the robust gradient descent framework of [447] to approximately solve *regularized subproblems* encountered by our final algorithm. We put the pieces together and give our complete algorithm in Sections 8.9.3 and 8.9.4. Throughout we assume $\epsilon\kappa^2 < 1$ is sufficiently small.

8.9.1 Noisy gradient oracle

In this section, we build noisy gradient oracles for the problems in Sections 8.7.3 and 8.7.4. We now give a formal definition below; the remaining sections will access F through this abstraction.

Definition 33 (Noisy gradient oracle)**.** *We call $\mathcal{O}_{\mathrm{ng}}(\theta, R)$ a (L, σ, δ)*-noisy gradient oracle *for $F : \mathbb{R}^d \to \mathbb{R}$ with minimizer $\theta_F^\star$ if on query $\theta \in \mathbb{R}^d$ and given $R \geq \|\theta - \theta_F^\star\|_2$, with probability $\geq 1 - \delta$ it returns $G(\theta)$ satisfying for a universal constant C_{ng},*

$$\|G(\theta) - \nabla F(\theta)\|_2 \leq C_{\mathrm{ng}} \left(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R\right).$$

If the returned $G(\theta)$ always satisfies the stronger bound $\|G(\theta) - \nabla F(\theta)\|_2 \leq C_{\mathrm{ng}}\sqrt{L\epsilon}\sigma$, we call $\mathcal{O}_{\mathrm{ng}}$ a (L, σ, δ)-radiusless noisy gradient oracle*.*

Before developing our implementations, we state a useful identifiability result relating gradient estimation to controlling operator norms of gradient second moments for finite sum functions.

Lemma 140. *Suppose $F_G(\theta) = \frac{1}{|G|}\sum_{i \in G} f_i(\theta)$ for some functions $\{f_i\}_{i \in G}$, and let $w \in \Delta^n$ be saturated with respect to bipartition $[n] = G \cup B$. For $\tilde{w} := \frac{w}{\|w\|_1}$ and $w_G^\star = \frac{1}{|G|}\mathbf{1}_G$, we have*

$$\left\|\mathbb{E}_{i \sim w_G^\star}\left[\nabla f_i(\theta)\right] - \mathbb{E}_{j \sim \tilde{w}}\left[\nabla f_j(\theta)\right]\right\|_2 \leq \sqrt{24\epsilon}\left(\left\|\mathrm{Cov}_{w_G^\star}\left(\{\nabla f_i(\theta)\}_{i \in [n]}\right)\right\|_{\mathrm{op}}^{\frac{1}{2}} + \left\|\mathrm{Cov}_{\tilde{w}}\left(\{\nabla f_i(\theta)\}_{i \in [n]}\right)\right\|_{\mathrm{op}}^{\frac{1}{2}}\right).$$

Proof. Throughout this proof, we let $\mathcal{C}$ supported on $[n] \times [n]$ be an optimal coupling between $i \sim w_G^\star$

and $j \sim \tilde{w}$, and follow notation from Proposition 37. For some unit vector $v \in \mathbb{R}^d$, we have

$$\begin{aligned}\left\langle v, \mathbb{E}_{i\sim w_G^\star}\left[\nabla f_i(\theta)\right] - \mathbb{E}_{j\sim\tilde{w}}\left[\nabla f_j(\theta)\right]\right\rangle &= \mathbb{E}_{i,j\sim\mathcal{C}}\left[\langle v, \nabla f_i(\theta) - \nabla f_j(\theta)\rangle\right] \\ &= \mathbb{E}_{i,j\sim\mathcal{C}}\left[\langle v, \nabla f_i(\theta) - \nabla f_j(\theta)\rangle \mathbf{1}_{i\neq j}\right] \\ &\leq \sqrt{6\epsilon}\mathbb{E}_{i,j\sim\mathcal{C}}\left[\langle v, \nabla f_i(\theta) - \nabla f_j(\theta)\rangle^2\right]^{\frac{1}{2}} \\ &\leq \sqrt{24\epsilon}\left(\mathbb{E}_{i\sim w_G^\star}\left[\langle v, \nabla f_i(\theta)\rangle^2\right]^{\frac{1}{2}} + \mathbb{E}_{j\sim\tilde{w}}\left[\langle v, \nabla f_j(\theta)\rangle^2\right]^{\frac{1}{2}}\right).\end{aligned}$$

The conclusion follows from choosing v to be in the direction of $\mathbb{E}_{i\sim w_G^\star}\left[\nabla f_i(\theta)\right] - \mathbb{E}_{j\sim\tilde{w}}\left[\nabla f_j(\theta)\right]$, and using the definition of the operator norm. □

Lemma 140 implies that for approximating gradients of functions F which are "closely approximated" by an (unknown) finite sum function F_G, it suffices to find a weighting $\tilde{w}$ such that the operator norm of $\mathrm{Cov}_{\tilde{w}}$ applied to gradients is bounded. We now demonstrate applications of this strategy to linear regression and smooth stochastic optimization.

Corollary 37. *Consider a robust linear regression instance where we have sample access to datasets $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$ and $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$ under Models 1, 4, with sample size n corresponding to Proposition 34. For*

$$F(\theta) = F^\star(\theta) = \mathbb{E}_{X,y\sim\mathcal{D}_{Xy}}\left[\frac{1}{2}(\langle X_i, \theta\rangle - y_i)^2\right],$$

we can construct a (L, σ, δ)-noisy gradient oracle for F in $O\left(nd\log^3(n)\log^2\left(\frac{n}{\delta}\right)\right)$ time, using $O(\log\frac{1}{\delta})$ queries of samples from Proposition 34.

Proof. We first demonstrate how to construct a noisy gradient oracle with success probability $\geq \frac{8}{10}$. The algorithm is as follows: first, sample a dataset under Models 1, 4, according to Proposition 34. Then, at point $\theta \in \mathbb{R}^d$, with probability $\frac{9}{10}$ Assumption 8 gives us a set $G := G_\theta$ (where we drop the subscript for simplicity, as this proof only discusses a single θ) with $|G| = (1-\epsilon)$ such that (8.41) holds. If we have the promise $\|\theta - \theta^*\|_2 \leq R$, let $w \in \Delta^n$ be the output of

$$\mathsf{FastCovFilter}\left(\{g_i(\theta)\}_{i\in[n]}, \frac{1}{n}_{1,1}, \frac{9}{10}, C_{\mathrm{est}}\left(L\sigma^2 + LR^2\right)\right), \text{ where } g_i(\theta) := X_i\left(\langle X_i, \theta\rangle - y_i\right).$$

where $\mathsf{FastCovFilter}$ (Algorithm 92) is the algorithm of Proposition 36. We then output $\mathbb{E}_{j\sim\tilde{w}}\left[g_j(\theta)\right]$, where $\tilde{w} = \frac{w}{\|w\|_1}$. The runtime is $O(nd\log^4 n)$ from the bottleneck operation of running $\mathsf{FastCovFilter}$. The assumptions of $\mathsf{FastCovFilter}$, namely a bound on $\mathrm{Cov}_{w_G^\star}\left(\{g_i(\theta)\}_{i\in[n]}\right)$, are satisfied by Assumption 8.2. Guarantees of $\mathsf{FastCovFilter}$ and Lemma 140 then imply

$$\left\|\mathbb{E}_{i\sim w_G^\star}\left[g_i(\theta)\right] - \mathbb{E}_{j\sim\tilde{w}}\left[g_j(\theta)\right]\right\|_2 = O\left(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R\right).$$

The conclusion follows from Assumption 8.2 which bounds $\left\|\mathbb{E}_{i\sim w_G^\star}\left[g_i(\theta)\right] - \nabla F(\theta)\right\|_2$.

We now describe how to boost the success probability, by calling our sample access $T = O(\log \frac{1}{\delta})$ times. Let $G^\star := \nabla F(\theta)$ be the true gradient, and run the procedure described above T times, producing $\{G_t\}_{t\in[T]}$, such that each G_t satisfies $\|G_t - G^\star\|_2 \le C(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R)$ with probability at least $\frac{4}{5}$, for some constant C. By standard binomial concentration, with probability at least $1-\delta$, at least $\frac{3}{5}$ of the $\{G_t\}_{t\in[T]}$ will satisfy this bound; call such a satisfying G_t "good." We return any G_t which is within distance $2C(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R)$ from at least $\frac{3}{5}$ of the gradient estimates. Note this will never return any G_t with $\|G_t - G^\star\|_2 > 4C(\sqrt{L\epsilon}\sigma + L\sqrt{\epsilon}R)$, since the triangle inequality implies this G_t will miss all the good estimates, a contradiction since there is at most a $\frac{2}{5}$ fraction which is not good. Thus, this procedure satisfies the requirements with $C_{\text{ng}} = 4C$; the additional runtime overhead is $O(\log^2(\frac{1}{\delta}))$ distance comparisons between our gradient estimates. □

Corollary 38. *Consider a robust Lipschitz (not necessarily smooth) stochastic optimization instance where we have sample access to datasets* $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$ *and* $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$ *under Models 1, 2, 6 with sample size* n *corresponding to Proposition 35. For*

$$F(\theta) = F^\star(\theta) + \frac{\mu}{2}\|\theta\|_2^2 = \mathbb{E}_{f\sim\mathcal{D}_f}\left[f(\theta)\right] + \frac{\mu}{2}\|\theta\|_2^2,$$

we can construct a $(L, 1, \delta)$*-radiusless noisy gradient oracle for* F *in* $O\left(nd\log^3(n)\log^2\left(\frac{n}{\delta}\right)\right)$ *time, using* $O(\log\frac{1}{\delta})$ *queries of samples from Proposition 35.*

Proof. The proof follows identically to that of Corollary 37, where we use Assumption 9.2 in place of Assumption 8.2. □

In most of Sections 8.9.2, 8.9.3, and 8.9.4, we will no longer discuss any specifics of the unknown function $F : \mathbb{R}^d \to \mathbb{R}$ we wish to optimize, except that it is L-smooth, μ-strongly convex, has minimizer $\theta_F^\star$, and supports a noisy gradient oracle $\mathcal{O}_{\text{ng}}$. We will apply Corollaries 37 and 38 to derive concrete rates and sample complexities for specific applications at the conclusion of this section.

8.9.2 Proximal subproblems

In this section, we develop a subroutine which obtains approximate minimizers of certain regularized problems which we call proximal subproblems. Concretely, we now formally define the notion of a *noisy proximal oracle*, an abstraction we will use in the following Section 8.9.4.

Definition 34 (Noisy proximal oracle)**.** *We call* $\mathcal{O}_{\text{np}}$ *a* (σ, δ)-noisy proximal oracle *for* L*-smooth* $F : \mathbb{R}^d \to \mathbb{R}$ *with minimizer* $\theta_F^\star$ *if on query* $\bar\theta \in \mathbb{R}^d$ *and given* $R \ge \left\|\bar\theta - \theta_F^\star\right\|_2$*, with probability* $\ge 1-\delta$

it returns $\hat{\theta}$ satisfying for a universal constant C_{np},

$$\left\|\hat{\theta} - \theta^\star_{\bar{\theta}}\right\|_2 \le C_{\mathrm{np}}\left(\sigma\sqrt{\frac{\epsilon}{L}} + \sqrt{\epsilon}R\right), \text{ where } \theta^\star_{\bar{\theta}} := \mathrm{argmin}_{\theta\in\mathbb{R}^d}\left\{F(\theta) + \frac{L}{2}\left\|\theta - \bar{\theta}\right\|_2^2\right\}. \tag{8.53}$$

The main goal of this section is to give a reduction from implementation of a noisy proximal oracle to implementation of a noisy gradient oracle. We first require a standard helper result bounding the distance from $\theta^\star_F$ to $\theta^\star_{\bar{\theta}}$ by R, which we later use to show stability of our iterates.

Lemma 141. *Following definition* (8.53)*, $\left\|\theta^\star_{\bar{\theta}} - \theta^\star_F\right\|_2 \le R$.*

Proof. The first-order optimality condition of $\theta^\star_{\bar{\theta}}$ and the three-point identity

$$\langle c - b, a - c\rangle = \frac{1}{2}\left\|b - a\right\|_2^2 - \frac{1}{2}\left\|c - a\right\|_2^2 - \frac{1}{2}\left\|c - b\right\|_2^2 \tag{8.54}$$

yield the standard ℓ_2 mirror descent guarantee

$$0 \le F(\theta^\star_{\bar{\theta}}) - F(\theta^\star_F) \le \left\langle \nabla F\left(\theta^\star_{\bar{\theta}}\right), \theta^\star_{\bar{\theta}} - \theta^\star_F\right\rangle \le \frac{L}{2}\left\|\bar{\theta} - \theta^\star_F\right\|_2^2 - \frac{L}{2}\left\|\theta^\star_{\bar{\theta}} - \theta^\star_F\right\|_2^2 - \frac{L}{2}\left\|\theta^\star_{\bar{\theta}} - \bar{\theta}\right\|_2^2.$$

The first two inequalities used convexity of F and that $\theta^\star_F$ is the minimizer of F. Rearranging yields the desired conclusion, via $\left\|\theta^\star_{\bar{\theta}} - \theta^\star_F\right\|_2 \le \left\|\bar{\theta} - \theta^\star_F\right\|_2 \le R$. □

We can now define our noisy proximal oracle implementation, Algorithm 49.

Algorithm 49: NoisyProximalOracle$(\bar{\theta}, R, \mathcal{O}_{\mathrm{ng}}, \sigma, \delta)$

Input: $\mathcal{O}_{\mathrm{ng}}$, a $(L, \sigma, \frac{\delta}{T})$-noisy gradient oracle for L-smooth $F : \mathbb{R}^d \to \mathbb{R}$ with minimizer $\theta^\star_F$ for $T = O\left(\log\left(\frac{1}{\sqrt{\epsilon}} + \frac{R\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$, $\bar{\theta} \in \mathbb{R}^d$ with $\left\|\bar{\theta} - \theta^\star_F\right\|_2 \le R$, $\delta \in (0, 1)$;
Output: With probability $\ge 1 - \delta$, $\hat{\theta}$ satisfying (8.53) for a universal constant C_{np};
$T \leftarrow O\left(\log\left(\frac{1}{\sqrt{\epsilon}} + \frac{R\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$ for a sufficiently large constant, $t \leftarrow 0$, $\theta_0 \leftarrow \bar{\theta}$;
while $t < T$ *and* $F(\theta_t) + \frac{L}{2}\left\|\theta_t - \bar{\theta}\right\|_2^2 \le \frac{2}{3}(F(\theta_{t-1}) + \frac{L}{2}\left\|\theta_{t-1} - \bar{\theta}\right\|_2^2)$ **do**
  $g_t \leftarrow \mathcal{O}_{\mathrm{ng}}(\theta_t, 4R, \frac{\delta}{T}) + L(\theta_t - \bar{\theta})$;
  $\theta_{t+1} \leftarrow \theta_t - \frac{1}{2L}g_t$;
  $t \leftarrow t + 1$;
Return: θ_{t-1};

Proposition 38. NoisyProximalOracle *correctly implements a (σ, δ)-noisy proximal oracle for F in $T = O\left(\log\left(\frac{1}{\sqrt{\epsilon}} + \frac{R\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$ calls to a $(L, \sigma, \frac{\delta}{T})$-noisy gradient oracle $\mathcal{O}_{\mathrm{ng}}$, and $O(dT)$ additional time.*

Proof. The runtime is immediate from the description of Algorithm 49, so we focus on correctness. For notational simplicity, denote the function we wish to minimize (with minimizer $\theta^\star_{\bar{\theta}}$) by

$$F^{\bar{\theta}}(\theta) := F(\theta) + \frac{L}{2}\left\|\theta - \bar{\theta}\right\|_2^2.$$

By Lemma 141 and the assumption on $\theta_0 = \bar{\theta}$, we have that $\left\|\theta_0 - \theta^\star_{\bar{\theta}}\right\|_2 \le 2R$ via the triangle inequality. We first claim that for all $0 \le t < T$, it is the case that

$$\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 \le 3R \implies \|\theta_t - \theta^\star_F\|_2 \le 4R. \tag{8.55}$$

In order to show this bound, since $F^{\bar{\theta}}$ is L-strongly convex, it suffices to show

$$F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) \le 4LR^2 \implies \frac{L}{2}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 \le 4LR^2 \implies \left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 \le 3R.$$

Thus, it suffices to show that the function error at all points θ_t is bounded by $4LR^2$; this is true initially since $\left\|\theta_0 - \theta^\star_{\bar{\theta}}\right\|_2 \le 2R$ and $F^{\bar{\theta}}$ is $2L$-smooth by L-smoothness of F, and the termination condition guarantees that while the algorithm runs it is always a descent algorithm. Hence, all calls to $\mathcal{O}_{\text{ng}}$ are successful. This implies for the error of the gradient estimation, $e_t := g_t - \nabla F^{\bar{\theta}}(\theta_t)$, that

$$\|e_t\|_2 = \left\|\mathcal{O}_{\text{ng}}\left(\theta_t, 3R, \frac{\delta}{T}\right) - \nabla F(\theta_t)\right\|_2 \le C_{\text{ng}}\left(\sqrt{L\epsilon}\sigma + 4L\sqrt{\epsilon}R\right). \tag{8.56}$$

Next, we expand: since $F^{\bar{\theta}}$ is $2L$-smooth,

$$\begin{aligned}
F^{\bar{\theta}}(\theta_{t+1}) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) &\le F^{\bar{\theta}}(\theta_t) + \left\langle \nabla F^{\bar{\theta}}(\theta_t), \theta_{t+1} - \theta_t \right\rangle + L\|\theta_{t+1} - \theta_t\|_2^2 - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) \\
&= F^{\bar{\theta}}(\theta_t) - \frac{1}{2L}\left\langle \nabla F^{\bar{\theta}}(\theta_t), \nabla F^{\bar{\theta}}(\theta_t) + e_t \right\rangle + \frac{1}{4L}\left\|\nabla F^{\bar{\theta}}(\theta_t) + e_t\right\|_2^2 - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) \\
&= F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) - \frac{1}{4L}\left\|\nabla F^{\bar{\theta}}(\theta_t)\right\|_2^2 + \frac{1}{4L}\|e_t\|_2^2 \\
&\le \frac{1}{2}\left(F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right)\right) + \frac{1}{4L}\|e_t\|_2^2.
\end{aligned} \tag{8.57}$$

In the last line, we used that L-strong convexity of $F^{\bar{\theta}}$ implies

$$\frac{1}{2L}\left\|\nabla F^{\bar{\theta}}(\theta_t)\right\|_2^2 \ge F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right).$$

Continuing from (8.57), we have from (8.56) that

$$\begin{aligned}
F^{\bar{\theta}}(\theta_{t+1}) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right) &\le \frac{1}{2}\left(F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right)\right) + \frac{1}{4L}\|e_t\|_2^2 \\
&\le \frac{1}{2}\left(F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right)\right) + \frac{C_{\text{ng}}^2}{2}\left(\sigma^2\epsilon + 16LR^2\epsilon\right).
\end{aligned}$$

Thus, we will continue decreasing the suboptimality gap by a $\frac{1}{3}$ factor so long as

$$\frac{C_{\text{ng}}^2}{2}\left(\sigma^2\epsilon + 16LR^2\epsilon\right) \le \frac{1}{6}\left(F^{\bar{\theta}}(\theta_t) - F^{\bar{\theta}}\left(\theta^\star_{\bar{\theta}}\right)\right).$$

Suppose Line 4 terminates because the function value failed to improve. From the above,

$$\frac{L}{2}\left\|\theta_{t-1}-\theta_{\bar{\theta}}^{\star}\right\|_2^2 \leq F^{\bar{\theta}}\left(\theta_{t-1}\right)-F^{\bar{\theta}}\left(\theta_{\bar{\theta}}^{\star}\right) \leq 3C_{\mathrm{ng}}^2\left(\sigma^2\epsilon+16LR^2\epsilon\right).$$

Rearranging yields the desired conclusion. Otherwise, in T iterations we improve the function error from at most $4LR^2$ in the first iteration, to at most $3C_{\mathrm{ng}}^2\left(\sigma^2\epsilon+16LR^2\epsilon\right)$ (since it is decreasing by a constant factor each time), which again yields the desired conclusion by the above calculation. □

8.9.3 Halving the distance to $\theta_F^{\star}$

In this section, we give a subroutine used in our full algorithm, which halves the distance to $\theta_F^{\star}$, the minimizer of F, outside of a sufficiently large radius. Suppose that we have an initial point $\bar{\theta}\in\mathbb{R}^d$, as well as a sufficiently large scalar $R\geq 0$ with the promise that (for a universal constant C_{lp})

$$\left\|\bar{\theta}-\theta_F^{\star}\right\|_2 \leq R, \text{ where } R \geq C_{\mathrm{lp}}\sigma\sqrt{\frac{\kappa\epsilon}{\mu}}. \tag{8.58}$$

We now state a procedure, HalfRadiusAccel, which returns a new θ with $\|\theta-\theta_F^{\star}\|_2\leq\frac{1}{2}R$. In its statement, we define a sequence of scalars $\{a_t,A_t\}_{0\leq t<T}$ given by the recursions

$$A_0=0,\ A_t=La_t^2,\ A_{t+1}=A_t+a_{t+1}. \tag{8.59}$$

The following fact is well-known (see for instance Chapter 2.2 of [418]).

Fact 19. *For all $0\leq t<T$, $A_t=\Theta(\frac{t^2}{L})$ and $a_t=\Theta(\frac{t}{L})$.*

Algorithm 50: HalfRadiusAccel$(\bar{\theta},R,\mathcal{O}_{\mathrm{np}},\delta)$

1 **Input:** $\mathcal{O}_{\mathrm{np}}$, a $(\sigma,\frac{\delta}{O(\sqrt{\kappa})})$-noisy proximal oracle for L-smooth, μ-strongly convex $F:\mathbb{R}^d\to\mathbb{R}$ with minimizer $\theta_F^{\star}$, $\bar{\theta}\in\mathbb{R}^d$ and $R\in\mathbb{R}_{\geq 0}$ satisfying (8.58), $\delta\in(0,1)$;
2 **Output:** With probability $\geq 1-\delta$, $\theta\in\mathbb{R}^d$ with $\|\theta-\theta_F^{\star}\|_2\leq\frac{1}{2}R$;
3 $T\leftarrow O\left(\sqrt{\kappa}\right)$ for a sufficiently large constant, $t\leftarrow 0$, $\theta_0\leftarrow\bar{\theta}$, $v_0\leftarrow\bar{\theta}$;
4 **while** $t<T$ **do**
5 $\quad y_t\leftarrow\frac{A_t}{A_{t+1}}\theta_t+\frac{a_{t+1}}{A_{t+1}}v_t$;
6 $\quad\theta_{t+1}\leftarrow\mathcal{O}_{\mathrm{np}}(y_t,3R)$;
7 $\quad v_{t+1}\leftarrow\mathrm{argmin}_{v\in\mathbb{B}}\left\{a_{t+1}\left\langle y_t-\theta_{t+1},v\right\rangle+\frac{1}{2L}\left\|v-v_t\right\|_2^2\right\}$, where $\mathbb{B}:=\{v\mid\left\|v-\bar{\theta}\right\|_2\leq R\}$;
8 $\quad t\leftarrow t+1$;
9 **Return:** θ_t;

We assume for simplicity that we can exactly solve the constrained subproblem in Line 7 in each iteration in $O(d)$ time; by a standard binary search on a Lagrange multiplier, we can solve this problem to high accuracy (cf. Proposition 8, [109]) and it is straightforward to verify this will not

be the bottleneck operation compared to calling $\mathcal{O}_{\mathrm{np}}$. Before analyzing the algorithm, we require a helper argument which shows that all iterates lie close to $\theta_F^\star$.

Lemma 142. *In every iteration $0 \le t \le T$, $\|y_t - \theta_F^\star\|_2 \le 3R$.*

Proof. Because y_t is a convex combination of θ_t and v_t, and v_t is constrained to lie at distance at most $2R$ from $\theta_F^\star$ by the triangle inequality since v_t and $\theta_F^\star$ both lie in $\mathbb{B}$, it suffices to show that all $\|\theta_t - \theta_F^\star\|_2 \le 3R$. We show a slightly stronger statement: defining $D_t := 2R + tC_{\mathrm{np}}(\sigma\sqrt{\frac{\epsilon}{L}} + 3\sqrt{\epsilon}R)$ where C_{np} is the constant from Definition 34, we claim that

$$\|\theta_t - \theta_F^\star\|_2 \le D_t. \tag{8.60}$$

The conclusion then follows from

$$D_T = 2R + O\left(\sigma\sqrt{\frac{\epsilon\kappa}{L}} + \sqrt{\kappa\epsilon}R\right) \le 3R,$$

assuming $\kappa\epsilon$ is small enough and using the lower bound on R from (8.58). To show (8.60), assume inductively that $\|\theta_t - \theta_F^\star\|_2 \le D_t$ for some t, and observe by convexity that we also have $\|y_t - \theta_F^\star\|_2 \le D_t$. Letting $\theta_t^\star$ be the optimal solution to the proximal subproblem defining $\mathcal{O}_{\mathrm{np}}(y_t, 3R)$, we have from the proof of Lemma 141 that $\|\theta_t^\star - \theta_F^\star\|_2 \le \|y_t - \theta_F^\star\|_2 \le D_t$. The conclusion follows from the bound $\|\theta_t - \theta_t^\star\|_2 \le C_{\mathrm{np}}(\sigma\sqrt{\frac{\epsilon}{L}} + 3\sqrt{\epsilon}R)$, given by the guarantees of $\mathcal{O}_{\mathrm{np}}$. □

The distance bound of Lemma 142 implies all calls to $\mathcal{O}_{\mathrm{np}}$ are correct. We now give the main technical lemma of this section, which shows a potential bound on iterates of HalfRadiusAccel.

Lemma 143. *In every iteration $0 \le t \le T$, define*

$$E_t := F(\theta_t) - F(\theta_F^\star),\ D_t := \frac{1}{2}\|v_t - \theta_F^\star\|_2^2,\ \Phi_t := A_t E_t + D_t.$$

Then, for all $0 \le t < T$, for a universal constant C_{pot},

$$\Phi_{t+1} - \Phi_t \le C_{\mathrm{pot}}\left(t\sigma\sqrt{\frac{\epsilon}{L}}R + t\sqrt{\epsilon}R^2 + t^2\sigma^2\frac{\epsilon}{L} + t^2\epsilon R^2\right).$$

Proof. Throughout this proof, let $\theta_t^\star$ be the solution to the subproblem defining $\mathcal{O}_{\mathrm{np}}(y_t, 3R)$, namely

$$\theta_t^\star := \mathrm{argmin}_\theta\left\{F(\theta) + \frac{L}{2}\|\theta - y_t\|_2^2\right\}.$$

Fix an iteration t. Define for notational convenience $\rho := C_{\mathrm{np}}(\sigma\sqrt{\frac{1}{L}} + 3R)$, so the error guarantee of $\mathcal{O}_{\mathrm{np}}$ is that $\left\|\theta_{t+1}^\star - \theta_{t+1}\right\|_2 \le \sqrt{\epsilon}\rho$. The optimality conditions of $\theta_{t+1}^\star$ and v_{t+1} respectively show

$$\left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - u\right\rangle \le \frac{L}{2}\|y_t - u\|_2^2 - \frac{L}{2}\left\|\theta_{t+1}^\star - u\right\|_2^2 - \frac{L}{2}\left\|y_t - \theta_{t+1}^\star\right\|_2^2 \text{ for all } u \in \mathbb{R}^d, \tag{8.61}$$

where we used the identity (F.1), and (since $\theta_F^\star \in \mathbb{B}$)

$$a_{t+1} L \langle y_t - \theta_{t+1}, v_{t+1} - \theta_F^\star \rangle \le \frac{1}{2} \|v_t - \theta_F^\star\|_2^2 - \frac{1}{2} \|v_{t+1} - \theta_F^\star\|_2^2 - \frac{1}{2} \|v_t - v_{t+1}\|_2^2. \tag{8.62}$$

Moreover, define the point $q_t = \frac{A_t}{A_{t+1}} \theta_t + \frac{a_{t+1}}{A_{t+1}} v_{t+1}$. Rearranging,

$$v_{t+1} = \frac{1}{a_{t+1}} \left(A_{t+1} q_t - A_t \theta_t \right) = \theta_{t+1}^\star + \frac{A_t}{a_{t+1}} \left(\theta_{t+1}^\star - \theta_t \right) - \frac{A_{t+1}}{a_{t+1}} \left(\theta_{t+1}^\star - q_t \right).$$

By convexity of F, the above display yields

$$\begin{aligned}
\left\langle \nabla F\left(\theta_{t+1}^\star\right), v_{t+1} - \theta_F^\star \right\rangle &= \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - \theta_F^\star \right\rangle \\
&\quad + \frac{A_t}{a_{t+1}} \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - \theta_t \right\rangle - \frac{A_{t+1}}{a_{t+1}} \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - q_t \right\rangle \\
&\ge F\left(\theta_{t+1}^\star\right) - F\left(\theta_F^\star\right) + \frac{A_t}{a_{t+1}} \left(F\left(\theta_{t+1}^\star\right) - F\left(\theta_t\right) \right) \\
&\quad - \frac{A_{t+1}}{a_{t+1}} \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - q_t \right\rangle \\
&= \frac{A_{t+1}}{a_{t+1}} \left(F\left(\theta_{t+1}^\star\right) - F\left(\theta_F^\star\right) \right) - \frac{A_t}{a_{t+1}} \left(F\left(\theta_t\right) - F\left(\theta_F^\star\right) \right) \\
&\quad - \frac{A_{t+1}}{a_{t+1}} \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - q_t \right\rangle.
\end{aligned} \tag{8.63}$$

Next, since the proximal problem defining θ_{t+1} is $2L$-smooth, the guarantees of $\mathcal{O}_{\mathrm{np}}$ imply

$$\begin{aligned}
F\left(\theta_{t+1}\right) + \frac{L}{2} \left\|\theta_{t+1} - y_t\right\|_2^2 &\le F\left(\theta_{t+1}^\star\right) + \frac{L}{2} \left\|\theta_{t+1}^\star - y_t\right\|_2^2 + L \left\|\theta_{t+1} - \theta_{t+1}^\star\right\|_2^2 \\
&\le F\left(\theta_{t+1}^\star\right) + \frac{L}{2} \left\|\theta_{t+1}^\star - y_t\right\|_2^2 + L\epsilon\rho^2.
\end{aligned}$$

Substituting the above into (8.63) yields

$$\begin{aligned}
a_{t+1} \left\langle \nabla F\left(\theta_{t+1}^\star\right), v_{t+1} - \theta_F^\star \right\rangle &\ge A_{t+1} E_{t+1} - A_t E_t - A_{t+1} \left\langle \nabla F\left(\theta_{t+1}^\star\right), \theta_{t+1}^\star - q_t \right\rangle \\
&\quad + A_{t+1} \left(\frac{L}{2} \left\|\theta_{t+1} - y_t\right\|_2^2 - \frac{L}{2} \left\|\theta_{t+1}^\star - y_t\right\|_2^2 - L\epsilon\rho^2 \right).
\end{aligned} \tag{8.64}$$

Next, since (unconstrained) optimality of $\theta_{t+1}^\star$ implies $\theta_{t+1}^\star = y_t - \frac{1}{L} \nabla F(\theta_{t+1}^\star)$,

$$\begin{aligned}
a_{t+1} \left\langle \nabla F(\theta_{t+1}^\star), v_{t+1} - \theta_F^\star \right\rangle &= L \left\langle y_t - \theta_{t+1}^\star, v_{t+1} - \theta_F^\star \right\rangle \\
&\le a_{t+1} L \langle y_t - \theta_{t+1}, v_{t+1} - \theta_F^\star \rangle + a_{t+1} L \left\|\theta_{t+1}^\star - \theta_{t+1}\right\|_2 \|v_{t+1} - \theta_F^\star\|_2 \\
&\le a_{t+1} L \langle y_t - \theta_{t+1}, v_{t+1} - \theta_F^\star \rangle + 2 a_{t+1} L \sqrt{\epsilon} \rho R.
\end{aligned}$$

In the last inequality, we used that v_{t+1} and $\theta_F^\star$ both lie in $\mathbb{B}$. Finally, combining the above display,

(8.64), and (8.62) yields the bound

$$\begin{aligned}\Phi_{t+1} - \Phi_t &\le A_{t+1}\left\langle \nabla F\left(\theta^\star_{t+1}\right), \theta^\star_{t+1} - q_t\right\rangle - \frac{1}{2}\left\|v_t - v_{t+1}\right\|_2^2 + 2a_{t+1}L\sqrt{\epsilon}\rho R \\ &\quad - A_{t+1}\left(\frac{L}{2}\left\|\theta_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|\theta^\star_{t+1} - y_t\right\|_2^2 - L\epsilon\rho^2\right) \\ &= A_{t+1}\left(\left\langle \nabla F\left(\theta^\star_{t+1}\right), \theta^\star_{t+1} - q_t\right\rangle + \frac{L}{2}\left\|\theta^\star_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|\theta_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|y_t - q_t\right\|_2^2\right) \\ &\quad + 2a_{t+1}L\sqrt{\epsilon}\rho R + A_{t+1}L\epsilon\rho^2.\end{aligned}$$

In the second-to-last line, we used that $v_t - v_{t+1} = \frac{A_{t+1}}{a_{t+1}}(y_t - q_t)$ and $\frac{A_{t+1}^2}{a_{t+1}^2} = LA_{t+1}$ by using definitions. Now applying (8.61) with $u = q_t$ gives

$$\begin{aligned}\left\langle \nabla F\left(\theta^\star_{t+1}\right), \theta^\star_{t+1} - q_t\right\rangle + \frac{L}{2}\left\|\theta^\star_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|\theta_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|y_t - q_t\right\|_2^2 \\ \le -\frac{L}{2}\left\|\theta_{t+1} - y_t\right\|_2^2 - \frac{L}{2}\left\|\theta^\star_{t+1} - q_t\right\|_2^2 \le 0.\end{aligned}$$

Thus, applying Fact 19 shows the potential change is bounded by the desired

$$\begin{aligned}2a_{t+1}L\sqrt{\epsilon}\rho R + A_{t+1}L\epsilon\rho^2 &= O\left(t\sqrt{\epsilon}\rho R + t^2\epsilon\rho^2\right) \\ &= O\left(t\sigma\sqrt{\frac{\epsilon}{L}}R + t\sqrt{\epsilon}R^2 + t^2\sigma^2\frac{\epsilon}{L} + t^2\epsilon R^2\right).\end{aligned}$$

□

Finally, we are ready to analyze the output of HalfRadiusAccel.

Lemma 144. *With probability at least $1 - \delta$, the output θ_T of* HalfRadiusAccel *satisfies*

$$\left\|\theta_T - \theta^\star_F\right\|_2 \le \frac{1}{2}R.$$

Proof. The failure probability comes from union bounding over the failures of calls to $\mathcal{O}_{\mathrm{np}}$, so we discuss correctness assuming all calls succeed. By telescoping Lemma 143 over T iterations, we have

$$\Phi_T \le \Phi_0 + C_{\mathrm{pot}}\sum_{t\in[T]}\left(t\sigma\sqrt{\frac{\epsilon}{L}}R + t\sqrt{\epsilon}R^2 + t^2\sigma^2\frac{\epsilon}{L} + t^2\epsilon R^2\right).$$

It is clear from definition that $\Phi_0 \le \frac{1}{2}R^2$, so it remains to bound all other terms. By examination,

$$\sum_{t\in[T]} t\sigma\sqrt{\frac{\epsilon}{L}}R = O\left(\sigma\kappa\sqrt{\frac{\epsilon}{L}}R\right) = O\left(R^2\right),$$
$$\sum_{t\in[T]} t\sqrt{\epsilon}R^2 = O\left(\kappa\sqrt{\epsilon}R^2\right) = O\left(R^2\right),$$
$$\sum_{t\in[T]} t^2\sigma^2\frac{\epsilon}{L} = O\left(\kappa^{1.5}\sigma^2\frac{\epsilon}{L}\right) = O\left(R^2\right),$$
$$\sum_{t\in[T]} t^2\epsilon R^2 = O\left(\kappa^{1.5}\epsilon R^2\right) = O\left(R^2\right).$$

Each of the above lines follows from $\epsilon\kappa^2$ being sufficiently small, and the lower bound in (8.58). Thus for a large enough value of C_{lp} in (8.58), we have that $\Phi_T \le R^2$. Since $\Phi_T = A_T E_T + D_T \ge A_T E_T$, choosing a sufficiently large value of $T = \sqrt{\kappa}$ combined with Fact 19 yields

$$E_T = F\left(\Theta_T\right) - F\left(\theta_F^\star\right) \le \frac{R^2}{A_T} \le \frac{LR^2}{8\kappa} = \frac{\mu R^2}{8}.$$

The conclusion follows from strong convexity of F, which implies $E_T \ge \frac{\mu}{2}\left\|\theta_T - \theta_F^\star\right\|_2^2$.

A note on constants. To check there are no conflict of interests hidden in the constants of this proof, note first that conditional on the bound at time T being at most R^2, the number of iterations T can be chosen solely as a function of the constants in Fact 19. From this point, the constant C_{lp} in (8.58) can be chosen to ensure that the potential bound is indeed R^2. □

8.9.4 Full accelerated algorithm

We conclude this section with a statement of a complete accelerated algorithm, and its applications.

Proposition 39. RobustAccel *correctly returns θ satisfying* (8.65) *with probability $\ge 1-\delta$. It runs in $O\left(\sqrt{\kappa}\log^2\left(\frac{R_0\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$ calls to $\mathcal{O}_{\mathrm{ng}}$, and $O\left(\sqrt{\kappa}d\log^2\left(\frac{R_0\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$ additional time.*

Proof. Correctness is immediate by iterating the guarantees of Lemma 144 T times, and taking a union bound over all (at most N) calls to $\mathcal{O}_{\mathrm{ng}}$, the only source of randomness in the algorithm. The runtime follows from examining HalfRadiusAccel and NoisyProximalOracle, since all operations take $O(d)$ time other than calls to $\mathcal{O}_{\mathrm{ng}}$. □

By combining Proposition 39 with Corollary 37 and 38, we derive the following conclusions.

Theorem 57. *Under Models 1, 4, supposing $\epsilon\kappa^2$ is sufficiently small, given $\theta_0 \in \mathbb{R}^d$ and $\left\|\theta_0 - \theta^\star\right\|_{\mathbf{\Sigma}^\star} \le R_0$,* RobustAccel *using the noisy gradient oracle of Corollary 37 returns θ with $\left\|\theta - \theta^\star\right\|_{\mathbf{\Sigma}^\star} =$*

Algorithm 51: RobustAccel$(\theta_0, R_0, \mathcal{O}_{\mathrm{ng}}, \delta)$

Input: $\mathcal{O}_{\mathrm{ng}}$, a $(L, \sigma, \frac{\delta}{N})$-noisy gradient oracle for L-smooth, μ-strongly convex $F : \mathbb{R}^d \to \mathbb{R}$ with minimizer $\theta_F^\star$ for $N = O\left(\sqrt{\kappa}\log^2\left(\frac{R_0\sqrt{L}}{\sigma\sqrt{\epsilon}}\right)\right)$, $\theta_0 \in \mathbb{R}^d$ with $\|\theta_0 - \theta_F^\star\|_2 \le R_0$, $\delta \in (0,1)$;

Output: With probability $\ge 1-\delta$, $\theta \in \mathbb{R}^d$ with

$$\|\theta - \theta_F^\star\|_2 = O\left(\sigma\sqrt{\frac{\kappa\epsilon}{\mu}}\right). \tag{8.65}$$

;

$T \leftarrow O\left(\log\left(\frac{R_0\sqrt{\mu}}{\sigma\sqrt{\kappa\epsilon}}\right)\right)$ for a sufficiently large constant, $t \leftarrow 0$;

$\mathcal{O}_{\mathrm{np}} \leftarrow$ NoisyProximalOracle$(\cdot,\cdot,\mathcal{O}_{\mathrm{ng}},\sigma,\frac{\delta}{N})$;

while $t < T$ **do**

  $\theta_{t+1} \leftarrow$ HalfRadiusAccel$(\theta_t, R_t, \mathcal{O}_{\mathrm{np}}, \frac{\delta}{T})$;

  $R_{t+1} \leftarrow \frac{1}{2}R_t$;

  $t \leftarrow t+1$;

Return: θ_t;

$O\left(\sigma\kappa\sqrt{\epsilon}\right)$ with probability at least $1-\delta$. The algorithm runs in

$$O\left(nd\sqrt{\kappa}\log^2\left(\frac{R_0}{\sigma\sqrt{\epsilon}}\right)\log^3(n)\log^2\left(\frac{n\log\left(\frac{R_0}{\sigma}\right)}{\delta\epsilon}\right)\right) \text{ time,}$$

where n is the dataset size of Proposition 34. The sample complexity of the method is

$$O\left(\log\left(\frac{\log\left(\frac{R_0}{\sigma}\right)}{\delta\epsilon}\right)\cdot\left(\frac{d\log(d/\epsilon)}{\epsilon}\right)\right).$$

Theorem 58. *Under Models 1, 2, and 5, supposing $\epsilon\kappa^2$ is sufficiently small for $\kappa := \max(1, \frac{L}{\mu})$, given $\theta_0 \in \mathbb{R}^d$ and $\left\|\theta_0 - \theta_{\mathrm{reg}}^\star\right\|_2 \le R_0$,* RobustAccel *using the noisy gradient oracle of Corollary 38 returns θ with $\left\|\theta - \theta_{\mathrm{reg}}^\star\right\|_2 = O\left(\sqrt{\frac{\kappa\epsilon}{\mu}}\right)$ with probability at least $1-\delta$. The algorithm runs in*

$$O\left(nd\sqrt{\kappa}\log^2\left(\frac{R_0\sqrt{L+\mu}}{\epsilon}\right)\log^3(n)\log^2\left(\frac{n\log\left(R_0\sqrt{L+\mu}\right)}{\delta\epsilon}\right)\right) \text{ time,}$$

where n is the dataset size of Proposition 35. The sample complexity of the method is

$$O\left(\log\left(\frac{\log\left(R_0\sqrt{L+\mu}\right)}{\delta\epsilon}\right)\cdot\left(\frac{d\log(d/\epsilon)}{\epsilon}\right)\right).$$

We remark that Theorem 57 can afford to reuse the same samples to construct gradient estimates for all θ we query per Assumption 8: though the set G_θ may change per θ, our noisy estimator also

provides a per-θ guarantee (and hence does not require the set to be consistent across calls).

8.10 Lipschitz generalized linear models

In this section, we give an algorithm for minimizing the regularized Moreau envelopes of Lipschitz statistical optimization problems under the strong contamination model, following the exposition of Section 8.7.4. Concretely, we recall we wish to compute an approximation to

$$\begin{aligned}\theta^\star_{\text{env}} := \text{argmin}_{\theta\in\mathbb{R}^d}\left\{F^\star_\lambda(\theta) + \frac{\mu}{2}\|\theta\|_2^2\right\}, \text{ where } F^\star(\theta) = \mathbb{E}_{f\sim\mathcal{D}_f}[f(\theta)],\\ \text{and } F^\star_\lambda(\theta) := \inf_{\theta'}\left\{F^\star(\theta') + \frac{1}{2\lambda}\|\theta-\theta'\|_2^2\right\}.\end{aligned} \tag{8.66}$$

Recall that in Corollary 38, we developed a noisy gradient oracle for $F^\star$, as long as our function distribution is captured by Model 6. However, techniques of Section 8.9 do not immediately apply to this setting, as $F^\star$ is not smooth. On the other hand, we do not have direct access to $F^\star_\lambda$.

We ameliorate this by developing a noisy gradient oracle (Definition 33) for the Moreau envelope $F^\star_\lambda$ in Section 8.10.1, under only Assumption 9; this will allow us to apply the acceleration techniques of Section 8.9 to the problem (8.66), which we complete in Section 8.10.2. Interestingly, our noisy gradient oracle for $F^\star_\lambda$ will have noise and runtime guarantees *independent* of the envelope parameter λ, allowing for a range of statistical and runtime tradeoffs for applications.

8.10.1 Noisy gradient oracle for the Moreau envelope

In this section, we give an efficient reduction which enables the construction of a noisy gradient oracle for $F^\star_\lambda$, assuming a radiusless noisy gradient oracle for $F^\star$, and that $F^\star$ is Lipschitz. We note that both of these assumptions hold under Model 6: we showed in Lemma 133 that $F^\star$ is $\sqrt{L}$-Lipschitz, and constructed a radiusless noisy gradient oracle in Corollary 38.

To begin, we recall standard facts about the Moreau envelope $F^\star_\lambda$, which can be found in e.g. [435].

Fact 20. *$F^\star_\lambda$ is λ^{-1}-smooth, satisfies $0 \le F^\star(\theta) - F^\star_\lambda(\theta) \le L\lambda$ for all $\theta\in\mathbb{R}^d$, and has gradient*

$$\nabla F^\star_\lambda(\theta) = \frac{1}{\lambda}\left(\theta - \text{prox}_{\lambda,F^\star}(\theta)\right), \text{ where } \text{prox}_{\lambda,F^\star}(\theta) := \text{argmin}_{\theta'}\left\{F^\star(\theta') + \frac{1}{2\lambda}\|\theta-\theta'\|_2^2\right\}.$$

Fact 20 demonstrates that to construct a noisy gradient oracle for $F^\star_\lambda$, it suffices to approximate the minimizer of the subproblem defining the $\text{prox}_{\lambda,F^\star}$ operator. To this end, we give a simple algorithm which approximates this proximal minimizer, based on noisy projected gradient descent.

We now begin our analysis of ApproxMoreauMinimizer. In the following discussion, for notational simplicity define $\theta^\star_{\bar\theta} := \text{prox}_{\lambda,F^\star}(\bar\theta)$ to be the exact minimizer of the proximal subproblem. We

Algorithm 52: ApproxMoreauMinimizer$(\bar{\theta}, \mathcal{O}_{\mathrm{ng}}, \delta)$

Input: $\mathcal{O}_{\mathrm{ng}}$, a $(L, 1, \frac{\delta}{T})$-radiusless noisy gradient oracle for $\sqrt{L}$-Lipschitz $F^\star : \mathbb{R}^d \to \mathbb{R}$ for $T = O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$, $\bar{\theta} \in \mathbb{R}^d$, $\delta \in (0,1)$;

Output: With probability $\geq 1-\delta$, $\hat{\theta}$ satisfying for a universal constant C_{env},

$$\left\|\hat{\theta} - \mathrm{prox}_{\lambda, F^\star}(\bar{\theta})\right\|_2 \leq C_{\mathrm{env}}\sqrt{L\epsilon}\lambda. \tag{8.67}$$

;

$T \leftarrow O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ for a sufficiently large constant, $t \leftarrow 0$, $\theta_0 \leftarrow \bar{\theta}$, $\eta \leftarrow \frac{4C_{\mathrm{ng}}^2\epsilon\lambda}{5}$;

while $t < T$ **do**

  $g_t \leftarrow \mathcal{O}_{\mathrm{ng}}(\theta_t, \infty, \frac{\delta}{T}) + \frac{1}{\gamma}(\theta_t - \bar{\theta})$;

  $\theta_{t+1} \leftarrow \mathrm{Proj}_{\mathbb{B}}(\theta_t - \eta g_t)$, where Proj is the ℓ_2 projection and $\mathbb{B} := \left\{\theta \mid \left\|\theta - \bar{\theta}\right\|_2 \leq 2\sqrt{L}\lambda\right\}$;

  $t \leftarrow t + 1$;

Return: θ_t;

require a simple helper bound showing $\theta_{\bar{\theta}}^\star$ does not lie too far from $\bar{\theta}$.

Lemma 145. *For* $\mathbb{B} := \left\{\theta \mid \left\|\theta - \bar{\theta}\right\|_2 \leq 2\sqrt{L}\lambda\right\}$ *and* $\theta_{\bar{\theta}}^\star := \mathrm{prox}_{\lambda, F^\star}(\bar{\theta})$, $\theta_{\bar{\theta}}^\star \in \mathbb{B}$.

Proof. Let $R := \left\|\theta_{\bar{\theta}}^\star - \bar{\theta}\right\|_2$. Since $\theta_{\bar{\theta}}^\star$ minimizes the proximal subproblem and $F^\star$ is convex,

$$F^\star\left(\theta_{\bar{\theta}}^\star\right) + \frac{R^2}{2\lambda} \leq F^\star\left(\bar{\theta}\right) \leq F^\star\left(\theta_{\bar{\theta}}^\star\right) + \left\langle \nabla F^\star\left(\bar{\theta}\right), \theta_{\bar{\theta}}^\star - \bar{\theta}\right\rangle \implies \frac{R^2}{2\lambda} \leq \sqrt{L}R.$$

□

We now prove correctness of ApproxMoreauMinimizer.

Lemma 146. ApproxMoreauMinimizer *correctly computes* $\hat{\theta}$ *satisfying* (8.67) *in* $O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ *calls to* $\mathcal{O}_{\mathrm{ng}}$, *with probability* $\geq 1-\delta$.

Proof. We assume throughout correctness of all calls to $\mathcal{O}_{\mathrm{ng}}$, which follows from a union bound. Consider some iteration $0 \leq t < T$, and let $\hat{\theta}_{t+1} := \theta_t - \eta g_t$ be the unprojected iterate. Since Euclidean projections decrease distances to points within a set (see e.g. Lemma 3.1, [98]), letting

$g_t = e_t + \nabla F^\star(\theta_t) + \frac{1}{\lambda}(\theta_t - \bar{\theta})$, where $\|e_t\|_2 \le C_{\mathrm{ng}}\sqrt{L\epsilon}$,

$$\begin{aligned}\frac{1}{2}\left\|\theta_{t+1} - \theta^\star_{\bar{\theta}}\right\|_2^2 &\le \frac{1}{2}\left\|\hat{\theta}_{t+1} - \theta^\star_{\bar{\theta}}\right\|_2^2 = \frac{1}{2}\left\|\theta_t - \eta g_t - \theta^\star_{\bar{\theta}}\right\|_2^2 \\ &= \frac{1}{2}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 - \eta\left\langle e_t + \nabla F^\star(\theta_t) + \frac{1}{\lambda}(\theta_t - \bar{\theta}), \theta_t - \theta^\star_{\bar{\theta}}\right\rangle + \frac{\eta^2}{2}\left\|g_t\right\|_2^2 \\ &\le \frac{1}{2}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 - \frac{\eta}{\lambda}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 + \eta\left\|e_t\right\|_2\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 + \frac{\eta^2}{2}\left\|g_t\right\|_2^2 \\ &\le \frac{1}{2}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 - \frac{\eta}{\lambda}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 + \eta C_{\mathrm{ng}}\sqrt{L\epsilon}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 + 5\eta^2 L.\end{aligned} \tag{8.68}$$

In the third line, we lower bounded $\left\langle \nabla F^\star(\theta_t) + \frac{1}{\lambda}(\theta_t - \bar{\theta}), \bar{\theta} - \theta^\star_{\bar{\theta}}\right\rangle$ by using strong convexity of $F^\star + \frac{1}{\lambda}\left\|\cdot - \bar{\theta}\right\|_2^2$; in the last line, we used the assumed bound on e_t as well as

$$\left\|g_t\right\|_2 \le \left\|\nabla F^\star(\theta_t)\right\|_2 + \left\|e_t\right\|_2 + \frac{1}{\lambda}\left\|\theta_t - \bar{\theta}\right\|_2 \le 3\sqrt{L} + C_{\mathrm{ng}}\sqrt{L\epsilon} \le \sqrt{10L},$$

for sufficiently small ϵ. Next, consider some iteration t where iterate θ_t satisfies

$$\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 \ge 4C_{\mathrm{ng}}\sqrt{L\epsilon}\lambda. \tag{8.69}$$

On this iteration, we have from the definition of η that

$$\eta C_{\mathrm{ng}}\sqrt{L\epsilon}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 \le \frac{\eta}{4\lambda}\left\|\theta - \theta^\star_{\bar{\theta}}\right\|_2^2,\ 5\eta^2 L \le \frac{\eta}{4\lambda}\left\|\theta - \theta^\star_{\bar{\theta}}\right\|_2^2.$$

Plugging these bounds back into (8.68), on any iteration where (8.69) holds,

$$\left\|\theta_{t+1} - \theta^\star_{\bar{\theta}}\right\|_2^2 \le \left(1 - \frac{\eta}{\lambda}\right)\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2 = \left(1 - \frac{4}{5}C_{\mathrm{ng}}^2\epsilon\right)\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2.$$

Because the squared distance $\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2^2$ is bounded by $16L\lambda^2$ initially and decreases by a factor of $O(\epsilon)$ every iteration until (8.69) no longer holds, it will reach an iteration where (8.69) no longer holds within T iterations. Finally, by (8.68), in every iteration t after the first where (8.69) is violated, either the squared distance to $\theta^\star_{\bar{\theta}}$ goes down, or it can go up by at most

$$\eta C_{\mathrm{ng}}\sqrt{L\epsilon}\left\|\theta_t - \theta^\star_{\bar{\theta}}\right\|_2 + 5\eta^2 L = O\left(\epsilon^2\lambda^2 L\right).$$

Here we used our earlier claim that the distance can only go up when (8.69) is false. Thus, the squared distance will never be more than $16C_{\mathrm{ng}}^2 L(\epsilon + O(\epsilon^2))\lambda^2$ within T iterations, as desired. □

By using the gradient characterization in Fact 20 and the noisy gradient oracle implementation of Corollary 38, we conclude this section with our Moreau envelope noisy gradient oracle claim.

Corollary 39. *Consider a robust Lipschitz stochastic optimization instance where we have sample*

access to datasets $\mathbf{X} = \{X_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$ *and* $y = \{y_i\}_{i\in[n]} \in \mathbb{R}^n$ *under Models 1, 2, 6 with sample size* n *corresponding to Proposition 35. For*

$$F(\theta) = F_\lambda^\star(\theta) + \frac{\mu}{2}\left\|\theta\right\|_2^2,$$

we can construct a $(L, O(1), \delta)$*-radiusless noisy gradient oracle in* $O(\frac{nd}{\epsilon}\log^3(n)\log^2(\frac{n}{\delta\epsilon})\log(\frac{1}{\epsilon}))$ *time. The sample complexity of our noisy gradient oracle is*

$$O\left(\log\left(\frac{1}{\delta\epsilon}\right)\cdot\left(\frac{d\log(d/\epsilon)}{\epsilon}\right)\right).$$

8.10.2 Accelerated optimization of the regularized Moreau envelope

We conclude by combining Proposition 39, the smoothness bound from Fact 20, and the noisy gradient oracle implementation of Corollary 39 to give this section's main result, Theorem 59.

Theorem 59. *Under Models 1, 2, and 6, supposing* $\epsilon\kappa^2$ *is sufficiently small for* $\kappa := \max(1, \frac{1}{\lambda\mu})$*, given* $\theta_0 \in \mathbb{R}^d$ *and* $\left\|\theta_0 - \theta_{\mathrm{env}}^\star\right\|_2 \le R_0$*,* RobustAccel *using the noisy gradient oracle of Corollary 39 returns* θ *with* $\left\|\theta - \theta_{\mathrm{env}}^\star\right\|_2 = O\left(\sqrt{\frac{\kappa\epsilon}{\mu}}\right)$ *with probability at least* $1-\delta$*. The algorithm runs in*

$$O\left(\frac{nd\sqrt{\kappa}}{\epsilon}\log\left(\frac{R_0\sqrt{\lambda^{-1}+\mu}}{\epsilon}\right)\log^3(n)\log^2\left(\frac{n\log\left(R_0\sqrt{\lambda^{-1}+\mu}\right)}{\delta\epsilon}\right)\log\left(\frac{1}{\epsilon}\right)\right) \text{ time,}$$

where n *is the dataset size of Proposition 35. The sample complexity of the method is*

$$O\left(\log\left(\frac{\log\left(R_0\sqrt{\lambda^{-1}+\mu}\right)}{\delta\epsilon}\right)\cdot\left(\frac{d\log(d/\epsilon)}{\epsilon}\right)\right).$$

Combined with Fact 20, Theorem 59 offers a range of tradeoffs by tuning the parameter λ: the smaller λ is, the more the Moreau envelope resembles the original function, but the statistical and runtime guarantees offered by Theorem 59 become correspondingly weaker.

8.11 Robust sub-Gaussian principal component analysis

We consider the following statistical task, which we call *robust sub-Gaussian principal component analysis* (PCA). Given samples $X_1, \ldots, X_n$ from sub-Gaussian[13] distribution $\mathcal{D}$ with covariance $\mathbf{\Sigma}$, an ϵ fraction of which are arbitrarily corrupted, the task asks to output unit vector u with $u^\top\mathbf{\Sigma}u \ge (1-\gamma)\left\|\mathbf{\Sigma}\right\|_\infty$[14] for tolerance γ. Ergo, the goal is to robustly return a $(1-\gamma)$-approximate

[13] See Section 3.2 for a formal definition.

[14] Throughout we use $\left\|\mathbf{M}\right\|_p$ to denote the Schatten p-norm.

top eigenvector of the covariance of sub-Gaussian $\mathcal{D}$. This is the natural extension of PCA to the robust statistics setting.

There has been a flurry of recent work on efficient algorithms for robust statistical tasks, e.g. covariance estimation and PCA. From an information-theoretic perspective, sub-Gaussian concentration suffices for robust covariance estimation. Nonetheless, to date all polynomial-time algorithms achieving nontrivial guarantees on covariance estimation (including PCA specifically) in the presence of adversarial noise require additional algebraic structure. For instance, sum-of-squares certifiably bounded moments have been leveraged in polynomial time covariance estimation algorithms [275, 330]; however, this is a stronger assumption than sub-Gaussianity.

In many applications (see discussion in [177]), the end goal of covariance estimation is PCA. Thus, a natural question which relaxes robust covariance estimation is: can we robustly estimate the top eigenvector of the covariance $\mathbf{\Sigma}$, assuming only sub-Gaussian concentration? Our work answers this question affirmatively via two incomparable algorithms. The first achieves $\gamma = O(\epsilon \log \epsilon^{-1})$ in polynomial time; the second achieves $\gamma = O(\sqrt{\epsilon \log \epsilon^{-1} \log d})$, in nearly-linear time under a mild gap assumption on $\mathbf{\Sigma}$. Moreover, both methods have nearly-optimal sample complexity.

Previous work

The study of estimators robust to a small fraction of adversarial outliers dates back to foundational work, e.g. [278, 512]. Following more recent work [337, 175], there has been significant interest in efficient, robust algorithms for statistical tasks in high-dimensional settings. We focus on methods robustly estimating covariance properties here, and defer a thorough discussion of the (extensive) robust statistics literature to [492, 360, 182].

There has been quite a bit of work in understanding and giving guarantees for robust covariance estimation where the uncorrupted distribution is exactly Gaussian [177, 178, 175, 132]. These algorithms strongly use relationships between higher-order moments of Gaussian distributions via Isserlis' theorem. Departing from the Gaussian setting, work of [337] showed that if the distribution is an affine transformation of a 4-wise independent distribution, robust covariance estimation is possible. This was extended by [330], which also assumed nontrivial structure in the moments of the distribution, namely that sub-Gaussianity was certifiable via the sum-of-squares proof system. To the best of our knowledge it has remained open to give nontrivial guarantees for robust estimation of any covariance properties under minimal assumptions, i.e. sub-Gaussian concentration.

All aforementioned algorithms also yield guarantees for robust PCA, by applying a top eigenvector method to the learned covariance. However, performing robust PCA via the intermediate covariance estimation step is lossy, both statistically and computationally. From a statistical perspective, $\Omega(d^2)$ samples are necessary to learn the covariance of a d-dimensional Gaussian in Frobenius norm (and for known efficient algorithms for spectral norm error [185]); in contrast, $O(d)$ samples suffice for (non-robust) PCA. Computationally, even when the underlying distrubution is exactly Gaussian,

the best-known covariance estimation algorithms run in time $\Omega(d^{3.25})$; algorithms working in more general settings based on the sum-of-squares approach require much more time. In contrast, the power method for PCA in a $d \times d$ matrix takes time $\widetilde{O}(d^2)$[15]. Motivated by this, our work initiates the direct study of robust PCA, which is often independently interesting in applications.

We remark there is another problem termed "robust PCA" in the literature, e.g. [103], under a different generative model. We defer a detailed discussion to [177], which experimentally shows that algorithms from that line of work do not transfer well to our corruption model.

Our results

We give two algorithms for robust sub-Gaussian PCA. Both are sample optimal, polynomial-time, and assume only sub-Gaussianity. We follow the distribution and corruption model described in Assumption 10.

Assumption 10 (Corruption model, see [175])**.** *Let $\mathcal{D}$ be a mean-zero distribution on $\mathbb{R}^d$ with covariance $\mathbf{\Sigma}$ and sub-Gaussian proxy $\mathbf{\Gamma} \preceq c\mathbf{\Sigma}$ for a constant c. Denote by index set G' with $|G'| = n$ a set of (uncorrupted) samples $\{X_i\}_{i \in G'} \sim \mathcal{D}$. An adversary arbitrarily replaces ϵn points in G'; we denote the new index set by $[n] = B \cup G$, where B, is the (unknown) set of points added by an adversary, and $G \subseteq G'$ is the set of points from G' that were not changed.*

As we only estimate covariance properties, the assumption that $\mathcal{D}$ is mean-zero only loses constants in problem parameters, by pairing samples and subtracting them (cf. [175], Section 4.5.1). Our first method is via a simple filtering approach, as summarized in the following (and developed in Section 8.11.1).

Theorem 60. *Under Assumption 10, let $\delta \in [0, 1]$, and $n = \Omega\left(\frac{d + \log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$. Algorithm 94 runs in time $O(\frac{nd^2}{\epsilon} \log \frac{n}{\delta\epsilon} \log \frac{n}{\delta})$, and outputs u with $u^\top \mathbf{\Sigma} u > (1 - C^\star \epsilon \log \epsilon^{-1}) \|\mathbf{\Sigma}\|_\infty$, for $C^\star$ a fixed multiple of parameter c in Assumption 10, with probability at least $1 - \delta$.*

Our second algorithm is more efficient under mild conditions, but yields a worse approximation $1 - \gamma$ for $\gamma = O(\sqrt{\epsilon \log \epsilon^{-1} \log d})$. Specifically, if there are few eigenvalues of $\mathbf{\Sigma}$ larger than $1 - \gamma$, our algorithm runs in nearly-linear time. Note that if there are many eigenvalues above this threshold, then the PCA problem itself is not very well-posed; our algorithm is very efficient in the interesting setting where the approximate top eigenvector is identifiable. We state our main algorithmic guarantee here, and defer details to Section 8.11.2.

Theorem 61. *Under Assumption 10, let $\delta \in [0, 1]$, $n = \Omega\left(\frac{d + \log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$, $\gamma = C\sqrt{\epsilon \log \epsilon^{-1} \log d}$, for C a fixed multiple of parameter c from Assumption 10, and let $t \in [d]$ satisfy $\mathbf{\Sigma}_{t+1} < (1 - \gamma) \|\mathbf{\Sigma}\|_\infty$. Algorithm 53 outputs a unit vector $u \in \mathbb{R}^d$ with $u^\top \mathbf{\Sigma} u \geq (1 - \gamma) \|\mathbf{\Sigma}\|_\infty$ in time $\widetilde{O}(\frac{nd}{\epsilon^{4.5}} + \frac{ndt}{\epsilon^{1.5}})$.*

[15] We say $g = \widetilde{O}(f)$ if $g = O(f \log^c f)$ for some constant $c > 0$.

We remark that $\Omega(d\epsilon^{-2})$ samples are necessary for a $(1-\epsilon)$-approximation to the top eigenvector of $\mathbf{\Sigma}$ via uncorrupted samples from $\mathcal{N}(0, \mathbf{\Sigma})$, so our first method is sample-optimal, as is our second up to a $\widetilde{O}(\epsilon^{-1})$ factor.

Preliminaries

General notation. $[n]$ denotes the set $1 \leq i \leq n$. Applied to a vector, $\|\cdot\|_p$ is the ℓ_p norm; applied to a symmetric matrix, $\|\cdot\|_p$ is the Schatten-p norm, i.e. the ℓ_p norm of the spectrum. The *dual norm* of ℓ_p is ℓ_q for $q = \frac{p}{p-1}$; when $p = \infty$, $q = 1$. Δ^n is the n-dimensional simplex (subset of positive orthant with ℓ_1-norm 1) and we define $\mathfrak{S}_\varepsilon^n \subseteq \Delta^n$ to be the truncated simplex:

$$\mathfrak{S}_\varepsilon^n := \left\{ w \in \mathbb{R}^n_{\geq 0} \;\middle|\; \|w\|_1 = 1,\ w \leq \frac{1}{n(1-\varepsilon)} \text{ entrywise} \right\}. \tag{8.70}$$

Matrices. $\mathbb{S}^d$ is $d \times d$ symmetric matrices, and $\mathbb{S}^d_{\geq 0}$ is the positive semidefinite subset. $\mathbf{I}$ is the identity of appropriate dimension. $\lambda_{\max}$, $\lambda_{\min}$, and Tr are the largest and smallest eigenvalues and trace of a symmetric matrix. For $\mathbf{M}, \mathbf{N} \in \mathbb{S}^d$, $\langle \mathbf{M}, \mathbf{N} \rangle := \operatorname{Tr}(\mathbf{MN})$ and we use the Loewner order $\preceq$, ($\mathbf{M} \preceq \mathbf{N}$ iff $\mathbf{N} - \mathbf{M} \in \mathbb{S}^d_{\geq 0}$). The seminorm of $\mathbf{M} \succeq 0$ is $\|v\|_{\mathbf{M}} := \sqrt{v^\top \mathbf{M} v}$.

Fact 21. *For $\mathbf{A}$, $\mathbf{B}$ with compatible dimension, $\operatorname{Tr}(\mathbf{AB}) = \operatorname{Tr}(\mathbf{BA})$. For $\mathbf{M}, \mathbf{N} \in \mathbb{S}^d_{\geq 0}$, $\langle \mathbf{M}, \mathbf{N} \rangle \geq 0$.*

Fact 22. *We have the following characterization of the Schatten-p norm: for $\mathbf{M} \in \mathbb{S}^d$, and $q = \frac{p}{p-1}$,*

$$\|\mathbf{M}\|_p = \sup_{\mathbf{N} \in \mathbb{S}^d,\ \|\mathbf{N}\|_q = 1} \langle \mathbf{N}, \mathbf{M} \rangle .$$

For $\mathbf{M} = \sum_{j \in [d]} \lambda_i v_i v_i^\top$, the satisfying $\mathbf{N}$ is $\frac{\sum_{j \in [d]} \pm \lambda_i^{p-1} v_i v_i^\top}{\|\mathbf{M}\|_p^{p-1}}$, so $\mathbf{NM}$ has spectrum $\frac{|\lambda|^p}{\|\mathbf{M}\|_p^{p-1}}$.

Distributions. We denote drawing vector X from distribution $\mathcal{D}$ by $X \sim \mathcal{D}$, and the covariance $\mathbf{\Sigma}$ of $\mathcal{D}$ is $\mathbb{E}_{X \sim \mathcal{D}} \left[X X^\top \right]$. We say scalar distribution $\mathcal{D}$ is γ^2-sub-Gaussian if $\mathbb{E}_{X \sim \mathcal{D}}[X] = 0$ and

$$\mathbb{E}_{X \sim \mathcal{D}} \left[\exp\left(tX \right) \right] \leq \exp\left(\frac{t^2 \gamma^2}{2} \right) \ \forall t \in \mathbb{R}.$$

Multivariate $\mathcal{D}$ has sub-Gaussian proxy $\mathbf{\Gamma}$ if its restriction to any unit v is $\|v\|_{\mathbf{\Gamma}}^2$-sub-Gaussian, i.e.

$$\mathbb{E}_{X \sim \mathcal{D}} \left[\exp\left(t X^\top v \right) \right] \leq \exp\left(\frac{t^2 \|v\|_{\mathbf{\Gamma}}^2}{2} \right) \text{ for all } \|v\|_2 = 1,\ t \in \mathbb{R}. \tag{8.71}$$

8.11.1 Robust sub-Gaussian PCA via filtering

In this section, we sketch the proof of Theorem 60, which gives guarantees on our filtering algorithm for robust sub-Gaussian PCA. This algorithm obtains stronger statistical guarantees than Theorem 61, at the cost of super-linear runtime; the algorithm is given as Algorithm 94. Our analysis stems largely from concentration facts about sub-Gaussian distributions, as well as the following (folklore) fact regarding estimation of variance along any particular direction.

Lemma 147. *Under Assumption 10, let $\delta \in [0,1]$, $n = \Omega\left(\frac{\log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$, and $u \in \mathbb{R}^d$ be a fixed unit vector. Algorithm 93,* 1DRobustVariance*, takes input $\{X_i\}_{i\in[n]}$, u, and ϵ, and outputs σ_u^2 with $|u^\top \mathbf{\Sigma} u - \sigma_u^2| < C u^\top \mathbf{\Sigma} u \cdot \epsilon \log \epsilon^{-1}$ with probability at least $1-\delta$, and runs in time $O(nd + n \log n)$, for C a fixed multiple of the parameter c in Assumption 10.*

In other words, we show that using corrupted samples, we can efficiently estimate a $1+O(\epsilon \log \epsilon^{-1})$-multiplicative approximation of the variance of $\mathcal{D}$ in any unit direction[16]. This proof is deferred to Appendix G.8 for completeness. Algorithm 94 combines this key insight with a soft filtering approach, suggested by the following known structural fact found in previous work (e.g. Lemma A.1 of [194], see also [493, 492]).

Lemma 148. *Let $\{a_i\}_{i\in[m]}$, $\{w_i\}_{i\in[m]}$ be sets of nonnegative reals, and $a_{\max} = \max_{i\in[m]} a_i$. Define $w_i' = \left(1 - \frac{a_i}{a_{\max}}\right) w_i$, for all $i \in [m]$. Consider any disjoint partition I_B, I_G of $[m]$ with $\sum_{i\in I_B} w_i a_i > \sum_{i\in I_G} w_i a_i$. Then, $\sum_{i\in I_B} w_i - w_i' > \frac{1}{2a_{\max}} \sum_{i\in[m]} w_i a_i > \sum_{i\in I_G} w_i - w_i'$.*

Our Algorithm 94, PCAFilter, takes as input a set of corrupted samples $\{X_i\}_{i\in[n]}$ following Assumption 10 and the corruption parameter ϵ. At a high level, it initializes a uniform weight vector $w^{(0)}$, and iteratively operates as follows (we denote by $\mathbf{M}(w)$ the empirical covariance $\sum_{i\in[n]} w_i X_i X_i^\top$).

1. $u_t \leftarrow$ approximate top eigenvector of $\mathbf{M}(w^{(t-1)})$ via power iteration.
2. Compute $\sigma_t^2 \leftarrow$ 1DRobustVariance$(\{X_i\}_{i\in[n]}, u_t, \epsilon)$.
3. If $\sigma_t^2 > (1 - O(\epsilon \log \epsilon^{-1})) \cdot u_t^\top \mathbf{M}(w^{(t-1)}) u_t$, then terminate and return u_t.
4. Else:
   (a) Sort indices $i \in [n]$ by $a_i \leftarrow \langle u_t, X_i \rangle^2$, with a_1 smallest.
   (b) Let $\ell \le i \le n$ be the smallest set for which $\sum_{i=\ell}^n w_i \ge 2\epsilon$, and apply the downweighting procedure of Lemma 148 to this subset of indices.

The analysis of Algorithm 94 then proceeds in two stages.

[16] Corollary 71 gives a slightly stronger guarantee that reusing samples does not break dependencies of u.

Monotonicity of downweighting. We show the invariant criteria for Lemma 148 (namely, that for the set $\ell \le i \le n$ in every iteration, there is more spectral mass on bad points than good) holds inductively for our algorithm. Specifically, lack of termination implies $\mathbf{M}(w^{(t-1)})$ puts significant mass on bad directions, which combined with concentration of good directions yields the invariant. The details of this argument can be found as Lemma 320.

Roughly uniform weightings imply approximation quality. As Lemma 148 then applies, the procedure always removes more mass from bad points than good, and thus can only remove at most 2ϵ mass total by the corruption model. Thus, the weights $w^{(t)}$ are always roughly uniform (in $\mathfrak{S}^n_{O(\epsilon)}$), which by standard concentration facts (see Appendix G.7) imply the quality of the approximate top eigenvector is good. Moreover, the iteration count is bounded by roughly d because whenever the algorithm does not terminate, enough mass is removed from large spectral directions. Combining with the termination criteria imply that when a vector is returned, it is a close approximation to the top direction of $\mathbf{\Sigma}$. Details can be found as Lemma 322 and in the proof of Theorem 60.

8.11.2 Robust sub-Gaussian PCA in nearly-linear time

We give our nearly-linear time robust PCA method, leveraging developments of Section 7.5. Throughout, we will be operating under Assumption 10, for some corruption parameter ϵ with $\epsilon \log \epsilon^{-1} \log d = O(1)$; $\epsilon = O(\frac{1}{\log d \log\log d})$ suffices. We now develop tools to prove Theorem 61.

Algorithm 53 uses three subroutines: our earlier 1DRobustVariance method (Lemma 147), an application of our earlier Proposition 30 to approximate the solution to

$$\min_{w \in \mathfrak{S}^n_\epsilon} \left\| \sum_{i \in [n]} w_i X_i X_i^\top \right\|_p, \text{ for } p = \Theta\left(\sqrt{\frac{\log d}{\epsilon \log \epsilon^{-1}}} \right), \tag{8.72}$$

and a method for computing approximate eigenvectors by [404] (discussed in Appendix G.9).

Proposition 40. *There is an algorithm* Power *(Algorithm 1, [404]), parameterized by $t \in [d]$, tolerance $\tilde{\epsilon} > 0$, $p \ge 1$, and $\mathbf{A} \in \mathbb{S}^d_{\succeq 0}$, which outputs orthonormal $\{z_j\}_{j \in [t]}$ with the guarantee*

$$\begin{cases} \left| z_j^\top \mathbf{A}^p z_j - \lambda_j^p(\mathbf{A}) \right| & \le \tilde{\epsilon} \lambda_j^p(\mathbf{A}) \\ \left| z_j^\top \mathbf{A}^{p-1} z_j - \lambda_j^{p-1}(\mathbf{A}) \right| & \le \tilde{\epsilon} \lambda_j^{p-1}(\mathbf{A}) \end{cases} \text{ for all } j \in [t]. \tag{8.73}$$

Here, $\lambda_j(\mathbf{A})$ is the j^{th} largest eigenvalue of $\mathbf{A}$. The total time required by the method is $O(\mathrm{nnz}(\mathbf{A}) \frac{tp \log d}{\varepsilon})$.

Algorithm 53 is computationally bottlenecked by the application of Proposition 30 on Line 2 and the t calls to 1DRobustVariance on Line 5, from which the runtime guarantee of Theorem 61 follows straightforwardly. To demonstrate correctness, we first certify the quality of the solution to (8.72).

Algorithm 53: RobustPCA$(\{X_i\}_{i\in[n]}, \epsilon, t)$

1 **Input:** $\{X_i\}_{i\in[n]}$ $\epsilon = O(\frac{1}{\log d \log\log d})$, $t \in [d]$ with $\mathbf{\Sigma}_{t+1} \preceq (1-\gamma)\mathbf{\Sigma}$ for γ in Theorem 61;
2 $w \leftarrow$ BoxedSchattenPacking (Proposition 30) on $\{\mathbf{A}_i = X_i X_i^\top\}_{i\in[n]}$, $\alpha \leftarrow \epsilon$, p as in (8.72);
3 $\mathbf{M} = \sum_{i\in[n]} w_i X_i X_i^\top$;
4 $\{z_j\}_{j\in[t]} = \mathsf{Power}(t, \epsilon, p, \mathbf{M})$;
5 $\alpha_j \leftarrow \mathsf{1DRobustVariance}(\{X_i\}_{i\in[n]}, \mathbf{M}^{\frac{p-1}{2}} z_j / \|\mathbf{M}^{\frac{p-1}{2}} z_j\|_2, \epsilon)$ for all $j \in [t]$;
6 **Return:** z_{j^*} for $j^* = \mathrm{argmax}_{j\in[t]} \alpha_j$;

Lemma 149. *Let $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$. With probability $1-\frac{\delta}{2}$, the uniform distribution over G attains value $(1+\frac{\tilde\epsilon}{2})\|\mathbf{\Sigma}\|_p$ for objective (8.72), where $\tilde\epsilon = C'\epsilon\log\epsilon^{-1}$ for a universal constant $C' > 0$.*

The proof of this is similar to results in e.g. [175, 360], and combines concentration guarantees with a union bound over all possible corruption sets B. This implies the following immediately, upon applying the guarantees of Proposition 30.

Corollary 40. *Let w be the output of Line 2 of* RobustPCA*. Then, we have $\|w\|_\infty \le \frac{1}{(1-2\epsilon)n}$, and $\left\|\sum_{i\in[n]} w_i X_i X_i^\top\right\|_p \le (1+\tilde\epsilon)\|\mathbf{\Sigma}\|_p$ under the guarantee of Lemma 149.*

Let w be the output of the solver. Recall that $\mathbf{M} = \sum_{i=1}^n w_i X_i X_i^\top$. Additionally, define

$$\mathbf{M}_G := \sum_{i\in G} w_i X_i X_i^\top,\ w_G := \sum_{i\in G} w_i,\ \mathbf{M}_B := \sum_{i\in B} w_i X_i X_i^\top,\ w_B := \sum_{i\in G} w_i\ . \tag{8.74}$$

Notice in particular that $\mathbf{M} = \mathbf{M}_G + \mathbf{M}_B$, and that all these matrices are PSD. We next prove the second, crucial fact, which says that $\mathbf{M}_G$ is a good approximator to $\mathbf{\Sigma}$ in Loewner ordering:

Lemma 150. *Let $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$. With probability at least $1-\frac{\delta}{2}$, $(1+\tilde\epsilon)\mathbf{\Sigma} \succeq \mathbf{M}_G \succeq (1-\tilde\epsilon)\mathbf{\Sigma}$.*

The proof combines the strategy in Lemma 149 with the guarantee of the SDP solver. Perhaps surprisingly, Corollary 40 and Lemma 150 are the only two properties about $\mathbf{M}$ that our final analysis of Theorem 61 will need. In particular, we have the following key geometric proposition, which carefully combines trace inequalities to argue that the corrupted points $\mathbf{M}_B$ cannot create too many new large eigendirections.

Proposition 41. *Let $\mathbf{M} = \mathbf{M}_G + \mathbf{M}_B$ be so that $\|\mathbf{M}\|_p \le (1+\tilde\epsilon)\|\mathbf{\Sigma}\|_p$, $\mathbf{M}_G \succeq 0$ and $\mathbf{M}_B \succeq 0$, and so that $(1+\tilde\epsilon)\mathbf{\Sigma} \succeq \mathbf{M}_G \succeq (1-\tilde\epsilon)\mathbf{\Sigma}$. Following notation of Algorithm 53, let*

$$\mathbf{M} = \sum_{j\in[d]} \lambda_j v_j v_j^\top,\ \mathbf{\Sigma} = \sum_{j\in[d]} \sigma_j u_j u_j^\top \tag{8.75}$$

be sorted eigendecompositions of $\mathbf{M}$ and $\mathbf{\Sigma}$, so $\lambda_1 \ge \ldots \ge \lambda_d$, and $\sigma_1 \ge \ldots \ge \sigma_d$. Let γ be as in

Theorem 61, and assume $\sigma_{t+1} < (1-\gamma)\sigma_1$. Then,

$$\max_{j\in[t]} v_j^\top \mathbf{\Sigma} v_j \geq (1-\gamma)\left\|\mathbf{\Sigma}\right\|_\infty.$$

Proof. For concreteness, we will define the parameters

$$p = \frac{2}{7}\sqrt{\frac{\log(3d)}{\tilde{\epsilon}}},\ \gamma = 14\sqrt{\tilde{\epsilon}\log(3d)} = 49p\tilde{\epsilon}.$$

For these choices of p, γ, we will use the following (loose) approximations for sufficiently small $\tilde{\epsilon}$:

$$\left(1-\frac{\gamma}{4}\right)^p = \left(1-\frac{\gamma}{4}\right)^{\frac{4}{\gamma}\log(3d)} \leq \frac{1}{3d},\ (1+\tilde{\epsilon})^p - (1-\tilde{\epsilon})^p \leq \exp(p\tilde{\epsilon}) - (1-p\tilde{\epsilon}) \leq 3p\tilde{\epsilon}. \tag{8.76}$$

Suppose for contradiction that all $v_j^\top \mathbf{\Sigma} v_j < (1-\gamma)\sigma_1$ for $j \in [t]$. By applying the guarantee of Corollary 40 and Fact 22, it follows that

$$\left\langle \mathbf{M}, \mathbf{M}^{p-1}\right\rangle = \left\|\mathbf{M}\right\|_p^p \leq (1+\tilde{\epsilon})^p \left\|\mathbf{\Sigma}\right\|_p^p. \tag{8.77}$$

Let $s \in [d]$ be the largest index such that $\sigma_s > \left(1-\frac{\gamma}{4}\right)\sigma_1$, and note that $s \leq t$. We define

$$\mathbf{N} := \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \preceq \mathbf{M}^{p-1}.$$

That is, $\mathbf{N}$ is the restriction of $\mathbf{M}^{p-1}$ to its top s eigendirections. Then,

$$\begin{aligned}\left\langle \mathbf{M}, \mathbf{M}^{p-1}\right\rangle &= \left\langle \mathbf{M}_B, \mathbf{M}^{p-1}\right\rangle + \left\langle \mathbf{M}_G, \mathbf{M}^{p-1}\right\rangle \\ &\geq \left\langle \mathbf{M}_B, \mathbf{M}^{p-1}\right\rangle + \left\langle (1-\tilde{\epsilon})\mathbf{\Sigma}, \mathbf{M}^{p-1}\right\rangle \geq \left\langle \mathbf{M}_B, \mathbf{N}\right\rangle + (1-\tilde{\epsilon})^p \left\|\mathbf{\Sigma}\right\|_p^p.\end{aligned} \tag{8.78}$$

In the second line, we used Lemma 150 twice, as well as the trace inequality Lemma 151 with $\mathbf{A} = \mathbf{M}$ and $\mathbf{B} = (1-\tilde{\epsilon})\mathbf{\Sigma}$. Combining (8.77) with (8.78), and expanding the definition of $\mathbf{M}_B$, yields

$$\begin{aligned}((1+\tilde{\epsilon})^p - (1-\tilde{\epsilon})^p)\left\|\mathbf{\Sigma}\right\|_p^p \geq \left\langle \mathbf{M}_B, \mathbf{N}\right\rangle &= \left\langle \mathbf{M}_B, \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \right\rangle \\ &= \left\langle \mathbf{M}, \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \right\rangle - \left\langle \mathbf{M}_G, \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \right\rangle \\ &\geq \left\langle \mathbf{M}, \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \right\rangle - (1+\tilde{\epsilon})\left\langle \mathbf{\Sigma}, \sum_{j\in[s]} \lambda_j^{p-1} v_j v_j^\top \right\rangle \\ &= \sum_{j\in[s]} \left(\lambda_j^p - (1+\tilde{\epsilon})\lambda_j^{p-1} v_j^\top \mathbf{\Sigma} v_j\right) \geq \sum_{j\in[s]} \left(\lambda_j^p - \lambda_j^{p-1}(1+\tilde{\epsilon})(1-\gamma)\sigma_1\right).\end{aligned} \tag{8.79}$$

The third line followed from from the spectral bound $\mathbf{M}_G \preceq (1+\tilde{\epsilon})\mathbf{\Sigma}$ of Lemma 150, and the fourth followed from the fact that $\{\lambda_j\}_{j\in[d]}$, $\{v_j\}_{j\in[d]}$ eigendecompose $\mathbf{M}$, as well as the assumption $v_j^\top \mathbf{\Sigma} v_j \le (1-\gamma)\sigma_1$ for all $j \in [t]$. Letting $S := \sum_{j\in[s]} \sigma_j^p$, and using both approximations in (8.76),

$$\|\mathbf{\Sigma}\|_p^p \le \sum_{j\in[s]} \sigma_j^p + \left(1-\frac{\gamma}{4}\right)^p (d-s)\sigma_1^p \le \frac{4}{3}S \implies \left((1+\tilde{\epsilon})^p - (1-\tilde{\epsilon})^p\right)\|\mathbf{\Sigma}\|_p^p \le 4p\tilde{\epsilon}S. \tag{8.80}$$

Next, we bound the last term of (8.79). By using $(1+\tilde{\epsilon})(1-\gamma) \le 1-\frac{\gamma}{2}$,

$$\begin{aligned}\sum_{j\in[s]} \left(\lambda_j^p - \lambda_j^{p-1}(1+\tilde{\epsilon})(1-\gamma)\sigma_1\right) &\ge \sum_{j\in[s]} \lambda_j^{p-1}\left(\lambda_j - \left(1-\frac{\gamma}{2}\right)\sigma_1\right) \\ &\ge \frac{\gamma}{6}\sum_{j\in[s]} \lambda_j^{p-1}\sigma_1 \ge \frac{\gamma}{6}(1-\tilde{\epsilon})^{p-1}\sum_{j\in[s]}\sigma_j^p \ge \frac{\gamma}{12}S.\end{aligned} \tag{8.81}$$

The second line used $\lambda_j - (1-\frac{\gamma}{2})\sigma_1 \ge (1-\tilde{\epsilon})\sigma_j - (1-\frac{\gamma}{2})\sigma_1 \ge \frac{\gamma}{6}\sigma_1$ by definition of s, Lemma 152 (twice), and $(1-\tilde{\epsilon})^{p-1} \ge \frac{1}{2}$. Combining (8.81) and (8.80) and plugging into (8.79),

$$4p\tilde{\epsilon}S \ge \frac{\gamma}{12}S \implies 48p\tilde{\epsilon} \ge \gamma.$$

By the choice of γ and p (i.e. $\gamma = 49p\tilde{\epsilon}$), we attain a contradiction. □

In the proof of Proposition 41, we used the following facts.

Lemma 151. *Let $\mathbf{A} \succeq \mathbf{B} \succeq 0$ be symmetric matrices and p a positive integer. Then we have*

$$\mathrm{Tr}\left(\mathbf{A}^{p-1}\mathbf{B}\right) \ge \mathrm{Tr}\left(\mathbf{B}^p\right).$$

Proof. For any $1 \le k \le p-1$,

$$\mathrm{Tr}\left(\mathbf{A}^k\mathbf{B}^{p-k}\right) \ge \mathrm{Tr}\left(\mathbf{A}^{k-1}\mathbf{B}^{\frac{p-k}{2}}\mathbf{A}\mathbf{B}^{\frac{p-k}{2}}\right) \ge \mathrm{Tr}\left(\mathbf{A}^{k-1}\mathbf{B}^{\frac{p-k}{2}}\mathbf{B}\mathbf{B}^{\frac{p-k}{2}}\right) = \mathrm{Tr}\left(\mathbf{A}^{k-1}\mathbf{B}^{p-k+1}\right).$$

The first step used the Extended Lieb-Thirring trace inequality $\mathrm{Tr}(\mathbf{M}\mathbf{N}^2) \ge \mathrm{Tr}(\mathbf{M}^\alpha\mathbf{N}\mathbf{M}^{1-\alpha}\mathbf{N})$ for $\alpha \in [0,1]$, $\mathbf{M},\mathbf{N} \in \mathbb{S}^d_{\succeq 0}$ (see e.g. Lemma 2.1, [19]), and the second $\mathbf{A} \succeq \mathbf{B}$. Finally, induction on k yields the claim. □

Lemma 152. *For all $j \in [d]$, $\lambda_j \ge (1-\tilde{\epsilon})\sigma_j$.*

Proof. By the Courant-Fischer minimax characterization of eigenvalues,

$$\lambda_j \ge \min_{k\in[j]} u_k^\top \mathbf{M} u_k.$$

However, we also have $\mathbf{M} \succeq \mathbf{M}_G \succeq (1-\tilde{\epsilon})\mathbf{\Sigma}$ (Lemma 150), yielding the conclusion. □

The guarantees of Proposition 41 were geared towards exact eigenvectors of the matrix $\mathbf{M}$. We now modify the analysis to tolerate inexactness in the eigenvector computation, in line with the processing of Line 5 of our Algorithm 53. This yields our final claim in Theorem 61.

Corollary 41. *In the setting of Proposition 41, and letting $\{z_j\}_{j\in[t]}$ satisfy (8.73), set for all $j \in [t]$*

$$y_j := \frac{\mathbf{M}^{\frac{p-1}{2}} z_j}{\left\|\mathbf{M}^{\frac{p-1}{2}} z_j\right\|_2}.$$

Then with probability at least $1-\delta$,

$$\max_{j\in[t]} y_j^\top \mathbf{\Sigma} y_j \geq (1-\gamma)\left\|\mathbf{\Sigma}\right\|_\infty.$$

Proof. Assume all y_j have $y_j^\top \mathbf{\Sigma} y_j \leq (1-\gamma)\sigma_1$ for contradiction. We outline modifications to the proof of Proposition 41. Specifically, we redefine the matrix $\mathbf{N}$ by

$$\mathbf{N} := \mathbf{M}^{\frac{p-1}{2}} \left(\sum_{j\in[s]} z_j z_j^\top\right) \mathbf{M}^{\frac{p-1}{2}}.$$

Because $\sum_{j\in[s]} z_j z_j^\top$ is a projection matrix, it is clear $\mathbf{N} \preceq \mathbf{M}^{p-1}$. Therefore, by combining the derivations (8.77) and (8.78), it remains true that

$$((1+\tilde{\epsilon})^p - (1-\tilde{\epsilon})^p)\left\|\mathbf{\Sigma}\right\|_p^p \geq \langle \mathbf{M}_B, \mathbf{N}\rangle = \langle \mathbf{M}, \mathbf{N}\rangle - \langle \mathbf{M}_G, \mathbf{N}\rangle.$$

We now bound these two terms in an analogous way from Proposition 41, with negligible loss; combining these bounds will again yield a contradiction. First, we have the lower bound

$$\left\langle \mathbf{M}, \sum_{j\in[s]} \mathbf{M}^{\frac{p-1}{2}} z_j z_j^\top \mathbf{M}^{\frac{p-1}{2}} \right\rangle = \sum_{j\in[s]} z_j^\top \mathbf{M}^p z_j \geq (1-\tilde{\epsilon}) \sum_{j\in[s]} \lambda_j^p.$$

Here, the last inequality applied the assumption (8.73) with respect to $\mathbf{M}^p$. Next, we upper bound

$$\begin{aligned}
\left\langle \mathbf{M}_G, \sum_{j\in[s]} \mathbf{M}^{\frac{p-1}{2}} z_j z_j^\top \mathbf{M}^{\frac{p-1}{2}} \right\rangle &\leq (1+\tilde{\epsilon}) \left\langle \mathbf{\Sigma}, \sum_{j\in[s]} \mathbf{M}^{\frac{p-1}{2}} z_j z_j^\top \mathbf{M}^{\frac{p-1}{2}} \right\rangle \\
&= (1+\tilde{\epsilon}) \sum_{j\in[s]} \left\|\mathbf{M}^{\frac{p-1}{2}} z_j\right\|_2^2 y_j^\top \mathbf{\Sigma} y_j \\
&\leq (1+\tilde{\epsilon})(1-\gamma)\sigma_1 \sum_{j\in[s]} z_j^\top \mathbf{M}^{p-1} z_j \\
&\leq (1-\gamma)(1+\tilde{\epsilon})^2 \sigma_1 \sum_{j\in[s]} \lambda_j^{p-1},
\end{aligned}$$

The first line used $\mathbf{M}_G \preceq (1+\tilde{\epsilon})\mathbf{\Sigma}$, the second used the definition of y_j, the third used our assumption $y_j^\top \mathbf{\Sigma} y_j \leq (1-\gamma)\sigma_1$, and the last used (8.73) with respect to $\mathbf{M}^{p-1}$. Finally, the remaining derivation (8.81) is tolerant to additional factors of $1+\tilde{\epsilon}$, yielding the same conclusion up to constants. □

Finally, we prove Theorem 61 by combining the tools developed thus far.

Proof of Theorem 61. Correctness of the algorithm is immediate from Corollary 41 and the guarantees of 1DRobustVariance. Concretely, Corollary 41 guarantees that one of the vectors we produce will be a $(1-\gamma)$-approximate top eigenvector (say some index $j \in [t]$), and 1DRobustVariance will only lose a negligible fraction $O(\epsilon \log \epsilon^{-1})$ of this quality (see Lemma 147); the best returned eigenvector as measured by 1DRobustVariance can only improve the guarantee. Finally, the failure probability follows by combining the guarantees of Lemmas 147, 149, and 150.

We now discuss runtime. The complexity of lines 2, 4, and 5, as guaranteed by Proposition 30, Proposition 40, and Lemma 147 are respectively (recalling $p = \widetilde{O}(\epsilon^{-0.5})$)

$$\widetilde{O}\left(\frac{nd}{\epsilon^{4.5}}\right), \widetilde{O}\left(\frac{ndt}{\epsilon^{1.5}}\right), \widetilde{O}\left(ndt\right).$$

Throughout we use that we can compute matrix-vector products in an arbitrary linear combination of the $X_i X_i^\top$ in time $O(nd)$; it is easy to check that in all runtime guarantees, nnz can be replaced by this computational cost. Combining these bounds yields the final conclusion. □

Chapter 9

Semi-Random Linear Systems

This chapter is based on [287, 319], with Arun Jambulapati, Jonathan A. Kelner, Jerry Li, Allen Liu, Christopher Musco, and Aaron Sidford.

9.1 Organization

In this chapter, we present several related results on solving various linear systems or regression tasks under the presence of a semi-random adversary. We will consider both the overcomplete, full-rank case (when the number of linear measurements is at least the dimensionality of the problem, and the solution is fully determined) as well as the undercomplete case (when the measurement matrix is rank-deficient and therefore additional structural assumptions, e.g. sparsity of the solution vector, must be imposed). In both settings, we will investigate the curious phenomenon that the guarantees of nearly-linear time algorithms for regression (as typically presented) do not behave well under *data augmentation*. Namely, by giving additional consistent information which does not impede upon the *statistical tractability* of the problem, an adversary can arbitrarily break the *runtime guarantees* of first-order methods. We propose two new robust solvers for these respective cases, which are able to match in appropriate semi-random settings (up to logarithmic factors) the performance of nearly-linear time algorithms succeeding under fully random, "well-conditioned" data.

Because our solution for the overcomplete case is encapsulated by a natural self-contained problem in numerical linear algebra, that of designing an *optimal diagonal preconditioner* for an arbitrary full-rank matrix, we present this result first in two self-contained sections (Sections 9.2, 9.3); the algorithm follows fairly straightforwardly (with a few tweaks) from our earlier developments in Section 7.4. We then give exposition and our algorithm for the undercomplete case in the later portion of this chapter.

9.2 Diagonal preconditioning

We demonstrate applications of our SDP solvers (Theorems 42, 43) to a classic problem in numerical linear algebra: how to best reduce the condition number of a PSD matrix $\mathbf{K} \in \mathbb{R}^{d\times d}$ via a diagonal scaling. Specifically, the goal is to find a $d \times d$ positive diagonal matrix $\mathbf{W}$ that approximately minimizes $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})$[1]. Given such a $\mathbf{W}$, a solution to the linear system $\mathbf{K}x = b$ can then be obtained by solving the better-conditioned system $\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}y = \mathbf{W}^{\frac{1}{2}}b$ and returning $x = \mathbf{W}^{\frac{1}{2}}y$.

Diagonal preconditioning has been studied since at least the 1950s [225, 518, 445]. Since the high computational cost of a generic SDP solver outweighs any computational benefits from reducing the condition number of $\mathbf{K}$, until our work finding a near optimal diagonal $\mathbf{W}$ has has not been feasible [448]. Instead, prior work mostly studies heuristics like Jacobi preconditioning, which simply chooses $\mathbf{W}$ to be the inverse-diagonal of $\mathbf{K}$, so $\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}$ has a constant diagonal [369, 225][2]. Another approach is to choose $\mathbf{W}$ to minimize $\|\mathbf{I} - \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\|_{\mathrm{F}}$ [254, 71]. We also note that diagonal preconditioning bears superficial resemblance to the *matrix scaling problem*, which has received recent attention in theoretical computer science [552, 145, 234]. The goal in matrix scaling is to find a diagonal scaling that equalizes the row and column norms of $\mathbf{K}$. While sometimes effective as a heuristic [322], since they target a different objective, scaling algorithms do not yield provable guarantees on $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})$.

In contrast to prior work on efficient algorithms, our main result yields a provably near optimal result for any $\mathbf{K}$. Specifically, letting $\kappa_o^\star(\mathbf{K}) := \min_{\text{diagonal } \mathbf{W}\succeq\mathbf{0}} \kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})$, we show that:

Theorem 62. *Let $\epsilon > 0$ be a fixed constant.*[3] *There is an algorithm, which given full-rank $\mathbf{K} \in \mathbb{S}^d_{\succ\mathbf{0}}$ computes $w \in \mathbb{R}^d_{\geq 0}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) \leq (1+\epsilon)\kappa_o^\star(\mathbf{K})$ with probability $\geq 1-\delta$ in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot (\kappa_o^\star(\mathbf{K}))^{1.5} \cdot \log^7\left(\frac{d\kappa_o^\star(\mathbf{K})}{\delta}\right)\right).$$

Importantly the runtime in Theorem 62 does not depend on the *original condition number*, $\kappa(\mathbf{K})$. Instead, it scales with the optimal $\kappa_o^\star(\mathbf{K})$, meaning that there is only a small computational overhead in comparison to the cost of solving the linear system after preconditioning, which is is $\tilde{O}(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_o^\star(\mathbf{K})})$ using conjugate gradient or accelerate gradient descent. Roughly, whenever $\kappa(\mathbf{K}) \geq (\kappa_o^\star(\mathbf{K}))^3$, we obtain a faster linear system solver for $\mathbf{K}$. This has applications to e.g. solving least squares regression problems of the form $\min_x \|\mathbf{A}x - b\|_2$. Solving the least squares problem

[1] The symmetry of the rescaling $\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}$ is important, as it preserves the symmetry and positive definiteness of $\mathbf{K}$, which is required for iterative solvers like the conjugate gradient method.

[2] A well-known result of van der Sluis [518, 251] proves that the Jacobi preconditioner gives an m-factor approximation to the optimal preconditioning problem where $m \leq d$ is the maximum number of non-zeros in any row of $\mathbf{K}$. We review and slightly strengthen this result in Appendix H.1.3. We also prove a new *dimension-independent* baseline result that may be of independent interest: the Jacobi preconditioner always obtains condition number no worse than $(\min \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})^2$.

[3] We do not focus on the ϵ dependence and instead take it to be constant since, in applications involving solving linear systems, there is little advantage to obtaining better than a two factor approximation (i.e. setting $\epsilon = 1$).

requires solving a linear system in the PSD matrix $\mathbf{K} = \mathbf{A}^\top \mathbf{A}$. In this setting, $\mathcal{T}_{\text{mv}}(\mathbf{K}) = O(\text{nnz}(\mathbf{A}))$, where $\text{nnz}(\mathbf{A})$ denotes the number of non-zeros in $\mathbf{A}$.

A result analogous to Theorem 62 but depending quadratically on $\kappa_o^\star(\mathbf{K})$) can be obtained by directly applying Theorem 43 with $n = d$, $\mathbf{M}_i = e_i e_i^\top$, $\kappa = \kappa_o^\star(\mathbf{K})$, and $\mathbf{B} = \frac{1}{\kappa}\mathbf{K}$ (i.e. using the dictionary of 1-sparse diagonal matrices to approximate $\mathbf{K}$). We obtain an improved $(\kappa_o^\star(\mathbf{K}))^{1.5}$ dependence via another homotopy method (similar to the one used for our SDP solver), which allows us to efficiently compute matrix-vector products with a symmetric square-root of $\mathbf{K}$. Access to the square root allows us to reduce the iteration complexity of our SDP solver.

In addition to standard diagonal preconditioning, which is an "outer scaling" problem, we also introduce an analogous "inner scaling" problem. Given a full rank $n \times d$ matrix $\mathbf{A}$ with $n \geq d$, the inner scaling problem asks to find an $n \times n$ nonnegative diagonal matrix $\mathbf{W}$ that approximately minimizes $\kappa\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)$. Let $\kappa_i^\star(\mathbf{A}) := \min_{\text{diagonal } \mathbf{W} \succeq \mathbf{0}} \kappa\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)$ denote the minimum (inner)-rescaled condition number. In contrast to the outer scaling problem applied e.g. to $\mathbf{K} = \mathbf{A}^\top \mathbf{A}$, where the goal is to rescale columns of $\mathbf{A}$ by $\mathbf{W}^{\frac{1}{2}}$, the inner scaling problem asks to rescale $\mathbf{A}$'s rows. To the best of our knowledge, this problem has not been studied previously. Directly applying Theorem 42 with $\kappa = \kappa_i^\star(\mathbf{A})$ and $\mathbf{M}_i = a_i a_i^\top$ for all $i \in [n]$ yields the following.

Theorem 63. *Let $\epsilon > 0$ be a fixed constant. There is an algorithm, which given full-rank $\mathbf{A} \in \mathbb{R}^{n \times d}$ for $n \geq d$ computes $w \in \mathbb{R}^n_{\geq 0}$ such that $\kappa(\mathbf{A}^\top \mathbf{W} \mathbf{A}) \leq (1+\epsilon)\kappa_i^\star(\mathbf{A})$ with probability $\geq 1 - \delta$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{A}) \cdot (\kappa_i^\star(\mathbf{A}))^{1.5} \cdot \log^6\left(\frac{n\kappa_i^\star(\mathbf{A})}{\delta}\right)\right).$$

A natural open question is if the $\kappa_o^\star(\mathbf{K})$ dependence in Theorem 62 (and the corresponding dependence in Theorem 63) can be reduced even further, ideally to $\sqrt{\kappa_o^\star(\mathbf{K})}$. This would match the most efficient linear system solvers in $\mathbf{K}$ under diagonal rescaling, *if the best known rescaling was known in advance*. We prove in Appendix H.1.4 that if a sufficiently approximation-tolerant width-independent variant of Theorem 42 is developed, it can be used to achieve such a runtime for both the classical outer scaling problem, and the inner scaling problem we introduce. We also give generalizations to finding rescalings which minimize natural average notions of conditioning, under existence of such a conjectured solver.

9.2.1 Inner scaling

We prove Theorem 63 on computing constant-factor optimal inner scalings.

Proof of Theorem 63. We apply Theorem 42 with $\kappa = \kappa_i^\star(\mathbf{A})$ and $\mathbf{M}_i = a_i a_i^\top$ for all $i \in [n]$, where rows of $\mathbf{A}$ are denoted $\{a_i\}_{i \in [n]}$. The definition of $\kappa_i^\star(\mathbf{A})$ and $\mathbf{A}^\top \mathbf{W} \mathbf{A} = \sum_{i \in [n]} w_i a_i a_i^\top$ (where $\mathbf{W} = \mathbf{diag}\,(w)$) implies that (7.7) is feasible for these parameters: namely, there exists $w^\star \in \mathbb{R}^n_{\geq 0}$

such that

$$\mathbf{I} \preceq \sum_{i\in[n]} w_i \mathbf{M}_i \preceq \kappa \mathbf{I}.$$

Moreover, factorizations (7.9) hold with $\mathbf{V}_i = a_i$ and $m = 1$. Note that $\mathcal{T}_{\text{mv}}(\{\mathbf{V}_i\}_{i\in[n]}) = O(\text{nnz}(\mathbf{A}))$ since this is the cost of multiplication through all rows of $\mathbf{A}$. □

9.2.2 Outer scaling

In this section, we prove a result on computing constant-factor optimal outer scalings of a matrix $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$. We first remark that we can obtain a result analogous to Theorem 63, but which scales quadratically in the $\kappa^\star_o(\mathbf{K})$, straightforwardly by applying Theorem 43 with $n = d$, $\mathbf{M}_i = e_i e_i^\top$ for all $i \in [d]$, $\kappa = \kappa^\star_o(\mathbf{K})$, and $\mathbf{B} = \frac{1}{\kappa}\mathbf{K}$. This is because the definition of $\kappa^\star_o$ implies there exists $w^\star \in \mathbb{R}^d_{\geq 0}$ with $\mathbf{I} \preceq (\mathbf{W}^\star)^{-\frac{1}{2}}\mathbf{K}(\mathbf{W}^\star)^{-\frac{1}{2}} \preceq \kappa\mathbf{I}$, where $\mathbf{W}^\star = \mathbf{diag}\,(w^\star)$ and where this $w^\star$ is the entrywise inverse of the optimal outer scaling attaining $\kappa^\star_o(\mathbf{K})$. This then implies

$$\frac{1}{\kappa}\mathbf{K} \preceq \sum_{i\in[d]} w^\star_i \mathbf{M}_i \preceq \mathbf{K},$$

since $\sum_{i\in[d]} w^\star_i \mathbf{M}_i = \mathbf{W}^\star$ and hence Theorem 43 applies, obtaining a runtime for the outer scaling problem of roughly $\mathbf{V}(\mathbf{K}) \cdot \kappa^\star_o(\mathbf{K})^2$ (up to logarithmic factors).

We give an alternative approach in this section which obtains a runtime of roughly $\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \kappa^\star_o(\mathbf{K})^{1.5}$, matching Theorem 63 up to logarithmic factors. Our approach is to define

$$\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$$

to be the positive definite square root of $\mathbf{K}$; we use this notation throughout the section, and let $\{a_i\}_{i\in[d]}$ be the rows of $\mathbf{A}$. We cannot explicitly access $\mathbf{A}$, but if we could, directly applying Theorem 63 suffices because $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) = \kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{A}^2\mathbf{W}^{\frac{1}{2}}) = \kappa(\mathbf{AWA})$ for any nonnegative diagonal $\mathbf{W} \in \mathbb{R}^{d\times d}$. We show that by using a homotopy method similar to the one employed in Section 7.4.2, we can implement this strategy with only a polylogarithmic runtime overhead. At a high level, the improvement from Theorem 43 is because we have explicit access to $\mathbf{K} = \mathbf{A}^2$. By exploiting cancellations in polynomial approximations we can improve the cost of iterations of Algorithm 29 from roughly κ (where one factor of $\sqrt{\kappa}$ comes from the cost of rational approximations to square roots in Section 7.4.2, and the other comes from the degree of polynomials), to roughly $\sqrt{\kappa}$.

Finally, throughout this section we will assume $\kappa(\mathbf{A}) \leq \kappa^\star_o(\mathbf{K})$, which is without loss of generality by rescaling based on the diagonal (Jacobi preconditioning), as we show in Appendix H.1.3. Also, for notational convenience we will fix a matrix $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ and denote $\kappa^\star := \kappa^\star_o(\mathbf{K})$.

Preliminaries

We first state a number of preliminary results which will be used in buliding our outer scaling method. We begin with a polynomial approximation to the square root, proven in Appendix H.1. It yields a corollary regarding approximating matrix-vector products with a matrix square root.

Fact 23 (Polynomial approximation of $\sqrt{\cdot}$)**.** *Let $\mathbf{M} \in \mathbb{S}^d_{\succ \mathbf{0}}$ have $\mu\mathbf{I} \preceq \mathbf{M} \preceq \kappa\mu\mathbf{I}$ where μ is known. Then for any $\delta \in (0,1)$, there is an explicit polynomial p of degree $O(\sqrt{\kappa}\log\frac{\kappa}{\delta})$ with*

$$(1-\delta)\mathbf{M}^{\frac{1}{2}} \preceq p(\mathbf{M}) \preceq (1+\delta)\mathbf{M}^{\frac{1}{2}}.$$

Corollary 42. *For any vector $b \in \mathbb{R}^d$, $\delta, \epsilon \in (0,1)$, and $\mathbf{M} \in \mathbb{S}^d_{\succ \mathbf{0}}$ with $\kappa(\mathbf{M}) \leq \kappa$, with probability $\geq 1-\delta$ we can compute $u \in \mathbb{R}^d$ such that*

$$\left\|u - \mathbf{M}^{\frac{1}{2}}b\right\|_2 \leq \epsilon \left\|\mathbf{M}^{\frac{1}{2}}b\right\|_2 \text{ in time } O\left(\mathcal{T}_{\text{mv}}(\mathbf{M}) \cdot \left(\sqrt{\kappa}\log\frac{\kappa}{\epsilon} + \log\frac{d}{\delta}\right)\right).$$

Proof. First, we compute a 2-approximation to μ in Fact 23 within the runtime budget using the power method (Fact 40), since κ is given. This will only affect parameters in the remainder of the proof by constant factors. If $u = \mathbf{P}b$ for commuting $\mathbf{P}$ and $\mathbf{M}$, our requirement is equivalent to

$$-\epsilon^2\mathbf{M} \preceq \left(\mathbf{P} - \mathbf{M}^{\frac{1}{2}}\right)^2 \preceq \epsilon^2\mathbf{M}.$$

Since square roots are operator monotone (by the Löwner-Heinz inequality), this is true iff

$$-\epsilon\mathbf{M}^{\frac{1}{2}} \preceq \mathbf{P} - \mathbf{M}^{\frac{1}{2}} \preceq \epsilon\mathbf{M}^{\frac{1}{2}},$$

and such a $\mathbf{P}$ which is applicable within the runtime budget is given by Fact 23. □

We next demonstrate two applications of Corollary 42 in estimating applications of products involving $\mathbf{A} = \mathbf{K}^{\frac{1}{2}}$, where we can only explicitly access $\mathbf{K}$. We will use the following standard fact about operator norms, whose proof is deferred to Appendix H.1.

Lemma 153. *Let $\mathbf{B} \in \mathbb{R}^{d\times d}$ and let $\mathbf{A} \in \mathbb{S}^d_{\succ \mathbf{0}}$. Then* $\min\left(\|\mathbf{AB}\|_2, \|\mathbf{BA}\|_2\right) \geq \frac{1}{\kappa(\mathbf{A})}\|\mathbf{B}\|_2\|\mathbf{A}\|_2$.

First, we discuss the application of a bounded-degree polynomial in **AWA** to a uniformly random unit vector, where $\mathbf{W}$ is an explicit nonnegative diagonal matrix.

Lemma 154. *Let $u \in \mathbb{R}^d$ be a uniformly random unit vector, let $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa$, and let $\mathbf{P}$ be a degree-Δ polynomial in **AWA** for an explicit diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$. For $\delta, \epsilon \in (0,1)$, with probability $\geq 1-\delta$ we can compute $w \in \mathbb{R}^d$ so $\|w - \mathbf{P}u\|_2 \leq \epsilon\|\mathbf{P}u\|_2$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \left(\Delta + \sqrt{\kappa}\log\frac{d\kappa}{\delta\epsilon}\right)\right).$$

Proof. We can write $\mathbf{P} = \mathbf{ANA}$ for some matrix $\mathbf{N}$ which is a degree-$O(\Delta)$ polynomial in $\mathbf{K}$ and $\mathbf{W}$, which we have explicit access to. Standard concentration bounds show that with probability at least $1-\delta$, for some $N = \mathrm{poly}(d, \delta^{-1})$, $\|\mathbf{P}u\|_2 \geq \frac{1}{N} \|\mathbf{P}\|_2$. Condition on this event for the remainder of the proof, such that it suffices to obtain additive accuracy $\frac{\epsilon}{N} \|\mathbf{P}\|_2$. By two applications of Lemma 153, we have

$$\|\mathbf{ANA}\|_2 \geq \frac{1}{\kappa} \|\mathbf{A}\|_2^2 \|\mathbf{N}\|_2 . \tag{9.1}$$

Our algorithm is as follows: for $\epsilon' \leftarrow \frac{\epsilon}{3N\kappa}$, compute v such that $\|v - \mathbf{A}u\|_2 \leq \epsilon' \|\mathbf{A}\|_2 \|u\|_2$ using Corollary 42, explicitly apply $\mathbf{N}$, and then compute w such that $\|w - \mathbf{AN}v\|_2 \leq \epsilon' \|\mathbf{A}\|_2 \|\mathbf{N}v\|_2$; the runtime of this algorithm clearly fits in the runtime budget. The desired approximation is via

$$\begin{aligned}
\|w - \mathbf{ANA}u\|_2 &\leq \|w - \mathbf{AN}v\|_2 + \|\mathbf{AN}v - \mathbf{ANA}u\|_2 \\
&\leq \epsilon' \|\mathbf{A}\|_2 \|\mathbf{N}\|_2 \|v\|_2 + \|\mathbf{AN}v - \mathbf{ANA}u\|_2 \\
&\leq 2\epsilon' \|\mathbf{A}\|_2^2 \|\mathbf{N}\|_2 + \|\mathbf{A}\|_2 \|\mathbf{N}\|_2 \|v - \mathbf{A}u\|_2 \\
&\leq 2\epsilon' \|\mathbf{A}\|_2^2 \|\mathbf{N}\|_2 + \epsilon' \|\mathbf{A}\|_2^2 \|\mathbf{N}\|_2 \\
&\leq 3\epsilon'\kappa \|\mathbf{ANA}\|_2 = \frac{\epsilon}{N} \|\mathbf{P}\|_2 .
\end{aligned}$$

The third inequality used $\|v\|_2 \leq \|\mathbf{A}u\|_2 + \epsilon \|\mathbf{A}\|_2 \|u\|_2 \leq (1+\epsilon) \|\mathbf{A}\|_2 \leq 2 \|\mathbf{A}\|_2$. □

We give a similar guarantee for random bilinear forms through $\mathbf{A}$ involving an explicit vector.

Lemma 155. *Let $u \in \mathbb{R}^d$ be a uniformly random unit vector, let $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa$, and let $v \in \mathbb{R}^d$. For $\delta, \epsilon \in (0,1)$, with probability $\geq 1 - \delta$ we can compute $w \in \mathbb{R}^d$ so $\langle w, v \rangle$ is an ϵ-multiplicative approximation to $u^\top \mathbf{A} v$ in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot \sqrt{\kappa} \log \frac{d\kappa}{\delta\epsilon}\right).$$

Proof. As in Lemma 154, for some $N = \mathrm{poly}(d, \delta^{-1})$ it suffices to give a $\frac{\epsilon}{N} \|\mathbf{A}v\|_2$-additive approximation. For $\epsilon' \leftarrow \frac{\epsilon}{N\sqrt{\kappa}}$, we apply Corollary 42 to obtain w such that $\|w - \mathbf{A}u\|_2 \leq \epsilon' \|\mathbf{A}u\|_2$, which fits within the runtime budget. Correctness follows from

$$|\langle \mathbf{A}u - w, v \rangle| \leq \|\mathbf{A}u - w\|_2 \|v\|_2 \leq \epsilon' \|\mathbf{A}\|_2 \|v\|_2 \leq \epsilon'\sqrt{\kappa} \|\mathbf{A}v\|_2 \leq \frac{\epsilon}{N} \|\mathbf{A}v\|_2 .$$

□

Implementing Algorithm 29 implicitly

In this section, we bound the complexity of Algorithm 29 in the following setting. Throughout this section denote $\mathbf{M}_i = a_i a_i^\top$ for all $i \in [d]$, where $\mathbf{A} \in \mathbb{S}^d_{\succ \mathbf{0}}$ has rows $\{a_i\}_{i \in [d]}$, and $\mathbf{A}^2 = \mathbf{K}$. We

assume that

$$\kappa(\mathbf{K}) \leq \kappa_{\text{scale}} := 3\kappa^\star,$$

and we wish to compute a reweighting $w \in \mathbb{R}^d_{\geq 0}$ such that

$$\kappa\left(\sum_{i\in[d]} w_i a_i a_i^\top\right) = \kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) \leq (1+\epsilon)\kappa^\star,$$

assuming there exists a reweighting $w^\star \in \mathbb{R}^d_{\geq 0}$ such that above problem is feasible with conditioning $\kappa^\star$. In other words, we assume we start with a matrix whose conditioning is within a 3-factor of the optimum after rescaling, and wish to obtain a $1+\epsilon$-approximation to the optimum. We show in the next section how to use a homotopy method to reduce the outer scaling problem to this setting.

Our strategy is to apply the method of Theorem 63. To deal with the fact that we cannot explictly access the matrix $\mathbf{A}$, we give a custom analysis of the costs of Lines 6, 7, and 13 under implicit access in this section, and prove a variant of Theorem 63 for this specific setting.

Estimating the smallest eigenvalue implicitly. We begin by discussing implicit implementation of Line 13. Our strategy combines the approach of Lemma 101 (applying the power method to the negative exponential), with Lemma 154 since to handle products through random vectors.

Lemma 156. *Given $\delta \in (0,1)$, constant $\epsilon > 0$, $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa_{\text{scale}}$, and diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ such that $\mathbf{M} := \mathbf{AWA} \preceq O(\kappa_{\text{scale}} \log d)\mathbf{I}$, we can compute a $O(\log d)$-additive approximation to $\lambda_{\min}(\mathbf{M})$ with probability $\geq 1-\delta$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_{\text{scale}}} \cdot \log^2 \frac{d\kappa_{\text{scale}}}{\delta}\right).$$

Proof. The proof of Lemma 101 implies it suffices to compute a 0.2-multiplicative approximation to the largest eigenvalue of $\mathbf{P}$, a degree-$\Delta = O(\sqrt{\kappa_{\text{scale}}}\log d)$ polynomial in $\mathbf{M}$. Moreover, letting $\Delta' = O(\log \frac{d}{\delta})$ be the degree given by Fact 40 with $\delta \leftarrow \frac{\delta}{3}$, the statement of the algorithm in Fact 40 shows it suffices to compute for a uniformly random unit vector u,

$$\left\|\mathbf{P}^\Delta u\right\|_2 \text{ and } \left\|\mathbf{P}^{\Delta+1} u\right\|_2 \text{ to multiplicative accuracy } \frac{1}{30}.$$

We demonstrate how to compute $\left\|\mathbf{P}^\Delta u\right\|_2$ to this multiplicative accuracy with probability at least $1-\frac{\delta}{3}$; the computation of $\left\|\mathbf{P}^{\Delta+1} u\right\|_2$ is identical, and the failure probability follows from a union bound over these three random events. Since $\mathbf{P}^\Delta$ is a degree-$O(\Delta\Delta') = O(\sqrt{\kappa_{\text{scale}}}\log d \log \frac{d}{\delta})$ polynomial in $\mathbf{AWA}$, the conclusion follows from Lemma 154. □

Estimating inner products with a negative exponential implicitly. We next discuss implicit implementation of Line 6. In particular, we give variants of Lemmas 97 and 98 which are tolerant to implicit approximate access of matrix-vector products.

Lemma 157. *Given $\delta \in (0,1)$, constant $\epsilon > 0$, $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa_{\text{scale}}$, and diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ such that $\mathbf{M} := \mathbf{AWA} \preceq O(\kappa_{\text{scale}} \log d)\mathbf{I}$ and $\lambda_{\min}(\mathbf{M}) = O(\log d)$, we can compute an ϵ-multiplicative approximation to $\operatorname{Tr}\exp(-\mathbf{M})$ with probability $\geq 1 - \delta$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_{\text{scale}}} \cdot \log^2 \frac{d\kappa_{\text{scale}}}{\delta}\right).$$

Proof. The proof of Lemma 97 shows it suffices to compute $k = O(\log \frac{d}{\delta})$ times, an $\frac{\epsilon}{3}\exp(-R)$-additive approximation to $u^\top \mathbf{P} u$ where $R = O(\log d)$, for uniformly random unit u and $\mathbf{P}$, a degree-$\Delta = O(\sqrt{\kappa_{\text{scale}}} \log d)$-polynomial in $\mathbf{M}$ with $\|\mathbf{P}\|_2 \leq \|\exp(-\mathbf{M})\|_2 + \frac{\epsilon}{3}\exp(-R) \leq \frac{4}{3}\exp(-R)$. Applying Lemma 154 with $\epsilon \leftarrow \frac{\epsilon}{4}$ to compute w, an approximation to $\mathbf{P}u$, the approximation follows:

$$|\langle w, u\rangle - \langle \mathbf{P}u, u\rangle| \leq \|w - \mathbf{P}u\|_2 \leq \frac{\epsilon}{4}\|\mathbf{P}\|_2 \leq \frac{\epsilon}{3}\exp(-R).$$

The runtime follows from the cost of applying Lemma 154 to all k random unit vectors. □

Lemma 158. *Given $\delta \in (0,1)$, constant $\epsilon > 0$, $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa_{\text{scale}}$, and diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ such that $\mathbf{M} := \mathbf{AWA} \preceq O(\kappa_{\text{scale}} \log d)\mathbf{I}$, we can compute $(\epsilon, O(\frac{1}{\kappa_{\text{scale}} d}))$-approximations to all*

$$\left\{\left\langle a_i a_i^\top, \exp(-\mathbf{M})\right\rangle\right\}_{i \in [d]},$$

with probability $\geq 1 - \delta$ in time

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_{\text{scale}}} \cdot \log^2 \frac{d\kappa_{\text{scale}}}{\delta}\right).$$

Proof. The proof of Lemma 98 implies it suffices to compute $k = O(\log \frac{d}{\delta})$ times, for each $i \in [d]$, the quantity $\langle u, \mathbf{P}a_i\rangle$ to multiplicative error $\frac{\epsilon}{2}$, for uniformly random unit vector u and $\mathbf{P}$, a degree-$\Delta = O(\sqrt{\kappa_{\text{scale}}} \log(\kappa_{\text{scale}} d))$-polynomial in $\mathbf{M}$. Next, note that since $a_i = \mathbf{A}e_i$ and $\mathbf{P} = \mathbf{ANA}$ for $\mathbf{N}$ an explicit degree-$O(\Delta)$ polynomial in $\mathbf{K}$ and $\mathbf{W}$, we have $\langle u, \mathbf{P}a_i\rangle = u^\top \mathbf{A}(\mathbf{NK}e_i)$. We can approximate this by some $\langle w, \mathbf{NK}e_i\rangle$ via Lemma 155 to the desired accuracy. The runtime comes from applying Lemma 155 k times, multiplying each of the resulting vectors w by $\mathbf{KN}$ and stacking them to form a $k \times d$ matrix $\widetilde{\mathbf{Q}}$, and then computing all $\|\widetilde{\mathbf{Q}}e_i\|_2$ for $i \in [d]$. □

Implementing a packing oracle implicitly. Finally, we discuss implementation of Line 7 of Algorithm 29. The requirement of Line 7 is a multiplicative approximation (and a witnessing reweighting) to the optimization problem

$$\max_{\substack{\sum_{i \in [d]} w_i \mathbf{M}_i \preceq \mathbf{I} \\ w \in \mathbb{R}^d_{\geq 0}}} v^\top w.$$

Here, v is explicitly given by an implementation of Line 6 of the algorithm, but we do not have $\{\mathbf{M}_i\}_{i \in [d]}$ explicitly. To implement this step implicitly, we recall the approximation requirements of

the solver of Proposition 26, as stated in Section 7.5. We remark that the approximation tolerance is stated for the decision problem tester of Section 7.5 (Proposition 66); once the tester is implicitly implemented, the same reduction as described in Appendix F.2 yields an analog to Proposition 26.

Corollary 43 (Approximation tolerance of Proposition 26)**.** *Let $\epsilon > 0$ be a fixed constant. The runtime of Proposition 26 is due to $T = O(\log(\frac{d}{\delta}) \log d \cdot \log\log \frac{\mathrm{OPT}_+}{\mathrm{OPT}_-})$ iterations, each of which requires $O(1)$ vector operations and $O(\epsilon)$-multiplicative approximations to*

$$\mathrm{Tr}\left(\mathbf{M}^p\right), \ \left\{\left\langle \mathbf{A}_i, \mathbf{M}^{p-1}\right\rangle\right\}_{i\in[d]} \text{ for } \mathbf{M} := \sum_{i\in[d]} w_i \mathbf{M}_i \text{ for an explicitly given } w \in \mathbb{R}^d_{\geq 0}, \tag{9.2}$$

where $p = O(\log d) \in \mathbb{N}$ is odd, and $S\mathbf{I} \preceq \mathbf{M} \preceq R\mathbf{I}$, for $R = O(\log d)$ and $S = \mathrm{poly}(\frac{1}{nd}, \kappa((\sum_{i\in[n]} \mathbf{M}_i))^{-1})$.

We remark that the lower bound S comes from the fact that the initial matrix of the solver of Section 7.5 is a bounded scaling of $\sum_{i\in[n]} \mathbf{M}_i$, and the iterate matrices are monotone in Loewner order. We now demonstrate how to use Lemmas 154 and 155 to approximate all quantities in (9.2). Throughout the following discussion, we specialize to the case where each $\mathbf{M}_i = a_i a_i^\top$, so $\mathbf{M}$ in (9.2) will always have the form $\mathbf{M} = \mathbf{AWA}$ for diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$, and $S = \mathrm{poly}((d\kappa_{\mathrm{scale}})^{-1})$.

Lemma 159. *Given $\delta \in (0,1)$, constant $\epsilon > 0$, $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa_{\mathrm{scale}}$, and diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ such that $S\mathbf{I} \preceq \mathbf{M} := \mathbf{AWA} \preceq O(\log d)\mathbf{I}$ where $S = \mathrm{poly}((d\kappa_{\mathrm{scale}})^{-1})$, we can compute an ϵ-multiplicative approximation to $\mathrm{Tr}(\mathbf{M}^p)$ for integer p in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot \left(p + \sqrt{\kappa_{\mathrm{scale}}} \log \frac{d\kappa_{\mathrm{scale}}}{\delta}\right) \cdot \log \frac{d}{\delta}\right).$$

Proof. As in Lemma 157, it suffices to compute $k = O(\log \frac{d}{\delta})$ times, an $\frac{\epsilon}{N} S^p$-additive approximation to $u^\top \mathbf{M}^p u$, for uniformly random unit vector u and $N = \mathrm{poly}(d, \delta^{-1})$. By applying Lemma 154 with accuracy $\epsilon' \leftarrow \frac{\epsilon S^p}{N R^p}$ to obtain w, an approximation to $\mathbf{M}^p u$, we have the desired

$$|\langle u, \mathbf{M}^p u\rangle - \langle u, w\rangle| \leq \|\mathbf{M}^p u - w\|_2 \leq \epsilon' \|\mathbf{M}^p u\|_2 \leq \epsilon' R^p \leq \frac{\epsilon}{N} S^p.$$

The runtime follows from k applications of Lemma 154 to the specified accuracy level. □

Lemma 160. *Given $\delta \in (0,1)$, constant $\epsilon > 0$, $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$ and $\kappa(\mathbf{K}) \leq \kappa_{\mathrm{scale}}$, and diagonal $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ such that $\mathbf{M} := \mathbf{AWA} \preceq O(\log d)\mathbf{I}$, we can compute an ϵ-multiplicative approximation to all*

$$\left\{\left\langle a_i a_i^\top, \mathbf{M}^{p-1}\right\rangle\right\}_{i\in[d]} \text{ where } \{a_i\}_{i\in[d]} \text{ are rows of } \mathbf{A},$$

where p is an odd integer, with probability $\geq 1-\delta$ in time

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K})\cdot\left(p+\sqrt{\kappa_{\mathrm{scale}}}\log\frac{d\kappa_{\mathrm{scale}}}{\delta}\right)\cdot\log\frac{d}{\delta}\right).$$

Proof. First, observe that for all $i\in[d]$ it is the case that

$$\left\langle a_i a_i^\top, \mathbf{M}^{p-1}\right\rangle = (\mathbf{A}e_i)^\top \mathbf{M}^{p-1}(\mathbf{A}e_i) \geq S^{p-1}\left\|\mathbf{A}\right\|_2^2\kappa_{\mathrm{scale}}^{-2}.$$

Letting $r=\frac{1}{2}(p-1)$ and following Lemma 158 and the above calculation, it suffices to show how to compute $k=O(\log\frac{d}{\delta})$ times, for each $i\in[d]$, the quantity $\langle u,\mathbf{M}^r a_i\rangle$ to multiplicative error $\frac{\epsilon}{2}$, for uniformly random unit vector u and $N=\mathrm{poly}(d,\delta^{-1})$. As in Lemma 158, each such inner product is $u^\top\mathbf{A}(\mathbf{N}\mathbf{K}e_i)$ for $\mathbf{N}$ an explicit degree-$O(p)$ polynomial in $\mathbf{K}$ and $\mathbf{W}$. The runtime follows from applying Lemma 155 k times and following the runtime analysis of Lemma 158. □

Putting it all together. Finally, we state our main result of this section, regarding rescaling well-conditioned matrices, by combining the pieces we have developed.

Corollary 44. *Given $\delta\in(0,1)$, constant $\epsilon>0$, $\mathbf{K}\in\mathbb{S}^d_{\succ\mathbf{0}}$ such that $\kappa(\mathbf{K})\leq 3\kappa^\star$, and such that $\kappa_o^\star(\mathbf{K})=\kappa^\star$, we can compute diagonal $\mathbf{W}\in\mathbb{S}^d_{\geq 0}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})\leq(1+\epsilon)\kappa^\star$ with probability $\geq 1-\delta$ in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K})\cdot(\kappa^\star)^{1.5}\cdot\log^6\left(\frac{d\kappa^\star}{\delta}\right)\right).$$

Proof. The proof is essentially identical to the proof of Theorem 63 by way of Theorem 42. In particular, we parameterize Theorem 42 with $\mathbf{M}_i=a_i a_i^\top$ where $\mathbf{A}=\mathbf{K}^{\frac{1}{2}}$ is the positive definite square root of $\mathbf{K}$ with rows $\{a_i\}_{i\in[d]}$. Then, running Algorithm 29 with an incremental search for the optimal $\kappa^\star$ yields an overhead of $\widetilde{O}(\kappa^\star\log(d\kappa^\star))$. The cost of each iteration of Algorithm 29 follows by combining Lemmas 156, 157, 158, 159, 160, and Corollary 43. □

Homotopy method

In this section, we use Corollary 44, in conjunction with a homotopy method similar to that of Section 7.4.2, to obtain our overall algorithm for outer scaling. We state here three simple helper lemmas which follow almost identically from corresponding helper lemmas in Section 7.4.2; we include proofs of these statements in Appendix H.1 for completeness.

Lemma 161. *For any matrix $\mathbf{K}\in\mathbb{S}^d_{\succ\mathbf{0}}$ and $\lambda\geq 0$, $\kappa_o^\star(\mathbf{K}+\lambda\mathbf{I})\leq\kappa_o^\star(\mathbf{K})$.*

Lemma 162. *Let $\mathbf{K}\in\mathbb{S}^d_{\succ\mathbf{0}}$. Then, for $\lambda\geq\frac{1}{\epsilon}\lambda_{\max}(\mathbf{K})$, $\kappa(\mathbf{K}+\lambda\mathbf{I})\leq 1+\epsilon$. Moreover, given a diagonal $\mathbf{W}\in\mathbb{S}^d_{\geq 0}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}(\mathbf{K}+\lambda\mathbf{I})\mathbf{W}^{\frac{1}{2}})\leq\kappa_{\mathrm{scale}}$ for $0\leq\lambda\leq\frac{\epsilon\lambda_{\min}(\mathbf{K})}{1+\epsilon}$, $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})\leq(1+\epsilon)\kappa_{\mathrm{scale}}$.*

Lemma 163. *Let $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$, and let $\mathbf{W} \in \mathbb{S}^d_{\geq 0}$ be diagonal. Then for any $\lambda > 0$,*

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\left(\mathbf{K} + \lambda \mathbf{I}\right)\mathbf{W}^{\frac{1}{2}}\right) \leq 2\kappa\left(\mathbf{W}^{\frac{1}{2}}\left(\mathbf{K} + \frac{\lambda}{2}\mathbf{I}\right)\mathbf{W}^{\frac{1}{2}}\right).$$

Theorem 62. *Let $\epsilon > 0$ be a fixed constant.*[4] *There is an algorithm, which given full-rank $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ computes $w \in \mathbb{R}^d_{\geq 0}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) \leq (1+\epsilon)\kappa^\star_o(\mathbf{K})$ with probability $\geq 1 - \delta$ in time*

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot (\kappa^\star_o(\mathbf{K}))^{1.5} \cdot \log^7\left(\frac{d\kappa^\star_o(\mathbf{K})}{\delta}\right)\right).$$

Proof. We will assume we know the correct value of $\kappa^\star_o(\mathbf{K})$ up to a $1 + O(\epsilon)$ factor throughout this proof for simplicity, and call this estimate $\kappa^\star$. This will add an overall multiplicative overhead of $O(1)$ by using an incremental search as in Theorem 42. We will also assume that $\kappa(\mathbf{K}) = O((\kappa^\star)^2)$ by first applying the Jacobi preconditioner; see Appendix H.1.3 for a proof.

Our algorithm follows the framework of Section 7.4.2 and runs in phases indexed by k for $0 \leq k \leq K$ for some K, each computing a scaling of $\mathbf{K} + \lambda_k \mathbf{I}$ with condition number $(1+\epsilon)\kappa^\star$; note that a scaling with condition number $\kappa^\star$ is always feasible for any $\lambda_k \geq 0$ by Lemma 161. We will define $\lambda_0 = \frac{1}{\epsilon}V$ where V is a constant-factor overestimate of $\lambda_{\max}(\mathbf{K})$, which can be obtained by Fact 40 without dominating the runtime. We will then set

$$\lambda_k = \frac{\lambda_0}{2^k},\ K = O\left(\log \kappa^\star\right).$$

Lemma 162 shows that we have a trivial scaling attaing condition number $(1+\epsilon)\kappa^\star$ for $\mathbf{K} + \lambda_0 \mathbf{I}$, and that if we can compute rescalings for all λ_k where $1 \leq k \leq K$, then the last rescaling is also a $(1+\epsilon)\kappa^\star$-conditioned rescaling for $\mathbf{K}$ up to adjusting ϵ by a constant.

Finally, we show how to implement each phase of the algorithm, given access to the reweighting from the previous phase. Note that Lemma 163 shows that the reweighting $\mathbf{W}$ computed in phase k yields a rescaling $\mathbf{W}^{\frac{1}{2}}(\mathbf{K} + \lambda_{k+1}\mathbf{I})\mathbf{W}^{\frac{1}{2}}$ which is $3\kappa^\star$-conditioned. By running the algorithm of Corollary 44 on $\mathbf{K} \leftarrow \mathbf{W}^{\frac{1}{2}}(\mathbf{K} + \lambda_{k+1}\mathbf{I})\mathbf{W}^{\frac{1}{2}}$, we compute the desired reweighting for phase $k+1$. The final runtime loses one logarithmic factor over Corollary 44 due to running for K phases. □

9.3 Overcomplete semi-random linear systems

Unlike a diagonal preconditioner, a good inner scaling does not yield a faster solution to a given least squares regression problem $\min_x \|\mathbf{A}x - b\|_2$. Instead, it allows for a faster solution to the weighted least squares problem $\min_x \|\mathbf{W}^{\frac{1}{2}}\mathbf{A}x - \mathbf{W}^{\frac{1}{2}}b\|_2$. It turns out that this has a number of interesting applications. Specifically, we now explore an interesting connection to *semi-random*

[4] We do not focus on the ϵ dependence and instead take it to be constant since, in applications involving solving linear systems, there is little advantage to obtaining better than a two factor approximation (i.e. setting $\epsilon = 1$).

noise models for least-squares regression. As a motivating example, consider the case when there is a hidden parameter vector $x_{\text{true}} \in \mathbb{R}^d$ that we want to recover, and we have a "good" set of consistent observations $\mathbf{A}_g x_{\text{true}} = b_g$, in the sense that $\kappa(\mathbf{A}_g^\top \mathbf{A}_g)$ is small. Here, we can think of $\mathbf{A}_g$ as being drawn from a well-conditioned distribution. Now, suppose an adversary gives us a superset of these observations $(\mathbf{A}, b)$ such that $\mathbf{A}x_{\text{true}} = b$, and $\mathbf{A}_g$ are an (unknown) subset of rows of $\mathbf{A}$, but $\kappa(\mathbf{A}^\top \mathbf{A}) \gg \kappa(\mathbf{A}_g^\top \mathbf{A}_g)$. Perhaps counterintuitively, by giving additional consistent data, the adversary can arbitrarily hinder the cost of iterative methods for finding x_{true}. Our inner scaling methods can be viewed as a way of "robustifying" linear system solvers to such noise models (finding $\mathbf{W}$ which yields a rescaled condition number that is at least as good as the indicator of the rows of $\mathbf{A}_g$, which are not known a priori). We also demonstrate additional applications in reducing risk in more general statistical regression settings.

The semi-random noise model we introduce for linear system solving follows a line of work originating in [85] for graph coloring. A semi-random model consists of an (unknown) planted instance which a classical algorithm performs well against, augmented by additional information given by a "monotone" or "helpful" adversary masking the planted instance. Conceptually, when an algorithm fails given this "helpful" information, the algorithm may have overfit to its problem specification. This model has been studied in various statistical settings [294, 220, 219, 397, 380]. Of particular relevance to our work, which studies robustness to semi-random noise in the context of fast algorithms (as opposed to the distinction between polynomial-time algorithms and computational intractability) is [133], which developed an algorithm for matrix completion under semi-random noise extending work of [353].

9.3.1 Semi-random linear systems

Consider the following semi-random noise model for solving an overdetermined, consistent linear system $\mathbf{A}x_{\text{true}} = b$ where $\mathbf{A} \in \mathbb{R}^{n \times d}$ for $n \geq d$.

Definition 35 (Semi-random linear systems)**.** *In the* semi-random noise model *for linear systems, a matrix $\mathbf{A}_g \in \mathbb{R}^{m \times d}$ with $\kappa(\mathbf{A}_g^\top \mathbf{A}_g) = \kappa_g$, $m \geq d$ is "planted" as a subset of rows of a larger matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$. We observe the vector $b = \mathbf{A}x_{\text{true}}$ for some $x_{\text{true}} \in \mathbb{R}^d$ we wish to recover.*

We remark that we call the model in Definition 35 "semi-random" because of the following motivating example: the rows $\mathbf{A}_g$ are feature vectors drawn from some "nice" (e.g. well-conditioned) distribution, and the dataset is contaminated by an adversary supplying additional data (a priori indistinguishable from the "nice" data), aiming to hinder conditioning of the resulting system.

Interestingly, Definition 35 demonstrates in some sense a shortcoming of existing linear system solvers: their brittleness to *additional, consistent information*. In particular, $\kappa(\mathbf{A}^\top \mathbf{A})$ can be arbitrarily larger than κ_g. However, if we were given the indices of the subset of rows $\mathbf{A}_g$, we could

instead solve the linear system $b_g = \mathbf{A}_g x_{\text{true}}$ with iteration count dependent on the condition number of $\mathbf{A}_g$. Counterintuitively, by giving additional rows, the adversary can arbitrarily increase the condition number of the linear system, hindering the runtime of conditioning-dependent solvers.

The inner rescaling algorithms we develop in Section 9.2 are well-suited for robustifying linear system solvers to the type of adversary in Definition 35. In particular, note that

$$\kappa_i^\star(\mathbf{A}) \leq \kappa\left(\mathbf{A}^\top \mathbf{W}_g \mathbf{A}\right) = \kappa\left(\mathbf{A}_g^\top \mathbf{A}_g\right) = \kappa_g,$$

where $\mathbf{W}_g$ is the diagonal matrix which is the 0-1 indicator of rows of $\mathbf{A}_g$. Our solvers for reweightings approximating $\kappa_i^\star$ can thus be seen as trading off the *sparsity* of $\mathbf{A}_g$ for the potential of "mixing rows" to attain a runtime dependence on $\kappa_i^\star(\mathbf{A}) \leq \kappa_g$. In particular, our resulting runtimes scale with $\text{nnz}(\mathbf{A})$ instead of $\text{nnz}(\mathbf{A}_g)$, but also depend on $\kappa_i^\star(\mathbf{A})$ rather than κ_g.

We remark that the other solvers we develop are also useful in robustifying against variations on the adversary in Definition 35. For instance, the adversary could instead aim to increase $\tau(\mathbf{A}^\top \mathbf{A})$, or give additional irrelevant features (i.e. columns of $\mathbf{A}$) such that only some subset of coordinates x_g are important to recover. For brevity, we focus on the model in Definition 35 in this work.

9.3.2 Statistical linear regression

The second application we give is in solving noisy variants of the linear system setting of Definition 35. In particular, we consider statistical regression problems with various generative models.

Definition 36 (Statistical linear regression)**.** *Given full rank $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$ produced via*

$$b = \mathbf{A} x_{\text{true}} + \xi,\ \xi \sim \mathcal{N}(0, \mathbf{\Sigma}), \tag{9.3}$$

where we wish to recover unknown $x_{\text{true}} \in \mathbb{R}^d$, return x so that (where expectations are taken over the randomness of ξ) the risk *(mean-squared error) $\mathbb{E}[\|x - x_{\text{true}}\|_2^2]$ is small.*

In this section, we define a variety of generative models (i.e. specifying a covariance matrix $\mathbf{\Sigma}$ of the noise) for the problem in Definition 36. For each of the generative models, applying our rescaling procedures will yield computational gains, improved risk bounds, or both. We give statistical and computational results for statistical linear regression in both the *homoskedastic* and *heteroskedastic* settings. In particular, when $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$ (i.e. the noise for every data point has the same variance), this is the well-studied homoskedastic setting pervasive in stastical modeling. When $\mathbf{\Sigma}$ varies with the data $\mathbf{A}$, the model is called heteroskedastic (cf. [252]).

In most cases, we do not directly give guarantees on exact mean squared errors via our preprocessing, but rather certify (possibly loose) upper bound surrogates. We leave direct certification of conditioning and risk simultaneously without a surrogate bound as an interesting future direction.

Heteroskedastic statistical guarantees

We specify two types of heteroskedastic generative models (i.e. defining the covariance $\mathbf{\Sigma}$ in (9.3)), and analyze the effect of rescaling a regression data matrix on reducing risk.

Noisy features. Consider the setting where the covariance in (9.3) has the form $\mathbf{\Sigma} = \mathbf{A}\mathbf{\Sigma}'\mathbf{A}^\top$, for matrix $\mathbf{\Sigma}' \in \mathbb{S}^d_{\succeq 0}$. Under this assumption, we can rewrite (9.3) as $b = \mathbf{A}(x_{\text{true}} + \xi')$, where $\xi' \sim \mathcal{N}(0, \mathbf{\Sigma}')$. Intuitively, this corresponds to exact measurements through $\mathbf{A}$, under noisy features $x_{\text{true}} + \xi'$. As in this case $b \in \text{Im}(\mathbf{A})$ always, regression is equivalent to linear system solving, and thus directly solving any reweighted linear system $\mathbf{W}^{\frac{1}{2}}\mathbf{A}x^* = \mathbf{W}^{\frac{1}{2}}b$ will yield $x^* = x_{\text{true}} + \xi'$.

We thus directly obtain improved computational guarantees by computing a reweighting $\mathbf{W}^{\frac{1}{2}}$ with $\kappa(\mathbf{A}^\top\mathbf{W}\mathbf{A}) = O(\kappa_i^\star(\mathbf{A}))$. Moreover, we note that the risk (Definition 36) of the linear system solution x^* is independent of the reweighting:

$$\mathbb{E}\left[\|x^* - x_{\text{true}}\|_2^2\right] = \mathbb{E}\left[\|\xi'\|_2^2\right] = \text{Tr}\left(\mathbf{\Sigma}'\right).$$

Hence, computational gains from reweighting the system are without statistical loss in the risk.

Row norm noise. Consider the setting where the covariance in (9.3) has the form

$$\mathbf{\Sigma} = \sigma^2 \mathbf{diag}\left(\left\{\|a_i\|_2^2\right\}_{i\in[n]}\right). \tag{9.4}$$

Intuitively, this corresponds to the setting where noise is independent across examples and the size of the noise scales linearly with the squared row norm. We first recall a standard characterization of the regression minimizer.

Fact 24 (Regression minimizer)**.** *Let the regression problem $\|\mathbf{A}x - b\|_2^2$ have minimizer $x^\star$, and suppose that $\mathbf{A}^\top\mathbf{A}$ is invertible. Then,*

$$x^\star = \left(\mathbf{A}^\top\mathbf{A}\right)^{-1}\mathbf{A}^\top b.$$

Using Fact 24, we directly prove the following upper bound surrogate holds on the risk under the model (9.3), (9.4) for the solution to any reweighted regression problem.

Lemma 164. *Under the generative model* (9.3), (9.4)*, letting $\mathbf{W} \in \mathbb{S}^n_{\succeq 0}$ be a diagonal matrix and*

$$x_w^\star := \text{argmin}_x \left\{\left\|\mathbf{W}^{\frac{1}{2}}\left(\mathbf{A}x - b\right)\right\|_2^2\right\},$$

we have

$$\mathbb{E}\left[\|x_w^\star - x_{\text{true}}\|_2^2\right] \leq \sigma^2 \frac{\text{Tr}\left(\mathbf{A}^\top\mathbf{W}\mathbf{A}\right)}{\lambda_{\min}\left(\mathbf{A}^\top\mathbf{W}\mathbf{A}\right)}.$$

Proof. By applying Fact 24, we have that

$$x_w^\star = \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W} \left(\mathbf{A} x_{\text{true}} + \xi\right) = x_{\text{true}} + \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W} \xi.$$

Thus, we have the sequence of derivations

$$\begin{aligned} \mathbb{E}\left[\|x_w^\star - x_{\text{true}}\|_{\mathbf{A}^\top \mathbf{W} \mathbf{A}}^2\right] &= \mathbb{E}\left[\left\|\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W} \xi\right\|_{\mathbf{A}^\top \mathbf{W} \mathbf{A}}^2\right] \\ &= \mathbb{E}\left[\left\langle \mathbf{W}^{\frac{1}{2}} \xi \xi^\top \mathbf{W}^{\frac{1}{2}}, \mathbf{W}^{\frac{1}{2}} \mathbf{A} \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}^{\frac{1}{2}} \right\rangle\right] \\ &= \sigma^2 \left\langle \mathbf{diag}\left(\left\{w_i \|a_i\|_2^2\right\}\right), \mathbf{W}^{\frac{1}{2}} \mathbf{A} \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}^{\frac{1}{2}} \right\rangle \\ &\leq \sigma^2 \text{Tr}\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right). \end{aligned} \tag{9.5}$$

The last inequality used the ℓ_1-ℓ_∞ matrix Hölder inequality and that $\mathbf{W}^{\frac{1}{2}} \mathbf{A} \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}^{\frac{1}{2}}$ is a projection matrix, so $\|\mathbf{W}^{\frac{1}{2}} \mathbf{A} \left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}^{\frac{1}{2}}\|_\infty = 1$. Lower bounding the squared $\mathbf{A}^\top \mathbf{W} \mathbf{A}$ norm by a $\lambda_{\min}(\mathbf{A}^\top \mathbf{W} \mathbf{A})$ multiple of the squared Euclidean norm yields the conclusion. □

We remark that the analysis in Lemma 164 of the surrogate upper bound we provide was loose in two places: the application of Hölder and the norm conversion. Lemma 164 shows that the risk under the generative model (9.4) can be upper bounded by a quantity proportional to $\tau(\mathbf{A}^\top \mathbf{W} \mathbf{A})$, the average conditioning of the reweighted matrix.

Directly applying Lemma 164, our risk bounds improve with the conditioningof the reweighted system. Hence, our scaling procedures improve both the computational and statistical guarantees of regression under this generative model, albeit only helping the latter through an upper bound.

Homoskedastic statistical guarantees

In this section, we work under the homoskedastic generative model assumption. In particular, throughout the covariance matrix in (9.3) will be a multiple of the identity:

$$\mathbf{\Sigma} = \sigma^2 \mathbf{I}. \tag{9.6}$$

We begin by providing a risk upper bound under the model (9.3), (9.6).

Lemma 165. *Under the generative model* (9.3), (9.6), *let* $x^\star := \text{argmin}_x \{\|\mathbf{A}x - b\|_2^2\}$. *Then,*

$$\mathbb{E}\left[\|x^\star - x_{\text{true}}\|_{\mathbf{A}^\top \mathbf{A}}^2\right] = \sigma^2 d \implies \mathbb{E}\left[\|x^* - x_{\text{true}}\|_2^2\right] \leq \frac{\sigma^2 d}{\lambda_{\min}(\mathbf{A}^\top \mathbf{A})}. \tag{9.7}$$

Proof. Using Fact 24, we compute

$$\begin{aligned} x^\star - x_{\text{true}} &= \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top b - x_{\text{true}} \\ &= \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \left(\mathbf{A} x_{\text{true}} + \xi\right) - x_{\text{true}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \xi. \end{aligned}$$

Therefore via directly expanding, and using linearity of expectation,

$$\begin{aligned} \mathbb{E}\left[\|x^* - x_{\text{true}}\|_{\mathbf{A}^\top \mathbf{A}}^2\right] &= \mathbb{E}\left[\left\|\mathbf{A}\left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \xi\right\|_2^2\right] \\ &= \mathbb{E}\left[\left\langle \xi\xi^\top, \mathbf{A}\left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \right\rangle\right] = \sigma^2 \left(\mathbf{A}\left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top\right) = \sigma^2 d. \end{aligned}$$

The final implication follows from $\lambda_{\min}(\mathbf{A}^\top \mathbf{A}) \|x^* - x_{\text{true}}\|_2^2 \le \|x^* - x_{\text{true}}\|_{\mathbf{A}^\top \mathbf{A}}^2$. □

Lemma 165 shows that in regards to our upper bound (which is loose in the norm conversion at the end), the notion of adversarial semi-random noise is at odds in the computational and statistical senses. Namely, given additional rows of the matrix $\mathbf{A}$, the bound (9.7) can only improve, since $\lambda_{\min}$ is monotonically increasing as rows are added. To address this, we give guarantees about recovering reweightings which match the best possible upper bound anywhere along the "computational-statistical tradeoff curve." We begin by providing a weighted analog of Lemma 165.

Lemma 166. *Under the generative model* (9.3), (9.6)*, letting* $\mathbf{W} \in \mathbb{S}_{\ge 0}^n$ *be a diagonal matrix and*

$$x_w^\star := \text{argmin}_x \left\{ \left\| \mathbf{W}^{\frac{1}{2}} \left(\mathbf{A} x - b\right) \right\|_2^2 \right\},$$

we have

$$\mathbb{E}\left[\|x_w^\star - x_{\text{true}}\|_2^2\right] \le \sigma^2 d \cdot \frac{\|w\|_\infty}{\lambda_{\min}\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right)}. \tag{9.8}$$

Proof. By following the derivations (9.5) (and recalling the definition of $x_w^\star$),

$$\begin{aligned} \mathbb{E}\left[\|x_w^\star - x_{\text{true}}\|_{\mathbf{A}^\top \mathbf{W} \mathbf{A}}^2\right] &= \mathbb{E}\left[\left\langle \xi\xi^\top, \mathbf{W}\mathbf{A}\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W} \right\rangle\right] \\ &= \sigma^2 \text{Tr}\left(\mathbf{W}\mathbf{A}\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}\right). \end{aligned} \tag{9.9}$$

Furthermore, by $\mathbf{W} \preceq \|w\|_\infty \mathbf{I}$ we have $\mathbf{A}^\top \mathbf{W}^2 \mathbf{A} \preceq \|w\|_\infty \mathbf{A}^\top \mathbf{W} \mathbf{A}$. Thus,

$$\text{Tr}\left(\mathbf{W}\mathbf{A}\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{W}\right) = \left\langle \mathbf{A}^\top \mathbf{W}^2 \mathbf{A}, \left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right)^{-1} \right\rangle \le \|w\|_\infty \text{Tr}(\mathbf{I}) = d \|w\|_\infty.$$

Using this bound in (9.9) and converting to Euclidean norm risk yields the conclusion. □

Lemma 166 gives a quantitative version of a computational-statistical tradeoff curve. Specifically, we give guarantees which target the best possible condition number of a 0-1 reweighting, subject to

a given level of $\lambda_{\min}(\mathbf{A}^\top \mathbf{W}\mathbf{A})$. In the following discussion we assume there exists $\mathbf{A}_g \subseteq \mathbf{A}$, a subset of rows, satisfying (for known κ_g, ν_g, and sufficiently small constant $\epsilon \in (0,1)$)

$$\kappa_g \leq \kappa\left(\mathbf{A}_g^\top \mathbf{A}_g\right) \leq (1+\epsilon)\kappa_g, \ \frac{1}{\lambda_{\min}\left(\mathbf{A}_g^\top \mathbf{A}_g\right)} \leq \nu_g. \tag{9.10}$$

Our key observation is that we can use existence of a row subset satisfying (9.10), combined with a slight modification of Algorithm 29, to find a reweighting w such that

$$\kappa\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right) = O\left(\kappa_g\right), \ \frac{\|w\|_\infty}{\lambda_{\min}\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right)} = O(\nu_g). \tag{9.11}$$

Lemma 167. *Consider running Algorithm 29, with the modification that in Line 7, we set*

$$\begin{aligned} x_t \leftarrow \textit{an } \frac{\epsilon}{10}\textit{-multiplicative approximation of } \operatorname{argmax}_{\substack{\sum_{i\in[n]} w_i \widetilde{\mathbf{A}}_i \preceq \mathbf{I} \\ x \in \mathbb{R}^n_{\geq 0}}} \left\langle \kappa v_t, w \right\rangle, \\ \textit{where for all } i \in [n], \ \widetilde{\mathbf{A}}_i := \begin{pmatrix} \mathbf{A}_i & \mathbf{0}_{d\times n} \\ \mathbf{0}_{n\times d} & \mathbf{diag}\left(\frac{\kappa_g}{\nu_g} e_i\right) \end{pmatrix}. \end{aligned} \tag{9.12}$$

Then, if (9.10) *is satisfied for some* $\mathbf{A} \in \mathbb{R}^{n\times d}$ *and row subset* $\mathbf{A}_g \subseteq \mathbf{A}$*, Algorithm 29 run on* $\kappa \leftarrow \kappa_g$ *and* $\{\mathbf{A}_i = a_i a_i^\top\}_{i\in[n]}$ *where* $\{a_i\}_{i\in[n]}$ *are rows of* $\mathbf{A}$ *will produce* w *satisfying* (9.11)*.*

Proof. We note that each matrix $\widetilde{\mathbf{A}}_i$ is the same as the corresponding $\mathbf{A}_i$, with a single nonzero coordinate along the diagonal bottom-right block. The proof is almost identical to the proof of Lemma 96, so we highlight the main differences here. The main property that Lemma 96 used was that Line 9 did not pass, which lets us conclude (7.63). Hence, by the approximation guarantee on each x_t, it suffices to show that for any $\mathbf{Y}_t \in \mathbb{S}^d_{\geq 0}$ with $\operatorname{Tr}(\mathbf{Y}_t) = 1$, (analogously to (7.62)),

$$\max_{\substack{\sum_{i\in[n]} w_i \widetilde{\mathbf{A}}_i \preceq \mathbf{I} \\ x \in \mathbb{R}^n_{\geq 0}}} \kappa_g \left\langle \mathbf{Y}_t, \sum_{i\in[n]} w_i \mathbf{A}_i \right\rangle \geq 1 - O(\epsilon). \tag{9.13}$$

However, by taking w to be the 0-1 indicator of the rows of $\mathbf{A}_g$ scaled down by $\lambda_{\max}(\mathbf{A}_g^\top \mathbf{A}_g)$, we have by the promise (9.10) that

$$\sum_{i\in[n]} w_i \widetilde{\mathbf{A}}_i = \frac{1}{\lambda_{\max}(\mathbf{A}_g^\top \mathbf{A}_g)} \preceq \mathbf{I} \impliedby \frac{1}{\lambda_{\max}(\mathbf{A}_g^\top \mathbf{A}_g)} \mathbf{A}_g^\top \mathbf{A}_g \preceq \mathbf{I}, \ \frac{\kappa_g}{\nu_g} \cdot \frac{1}{\lambda_{\max}(\mathbf{A}_g^\top \mathbf{A}_g)} \leq 1. \tag{9.14}$$

Now, it suffices to observe that (9.14) implies our indicator w is feasible for (9.13), so

$$\max_{\substack{\sum_{i\in[n]} w_i \widetilde{\mathbf{A}}_i \preceq \mathbf{I} \\ x \in \mathbb{R}^n_{\geq 0}}} \kappa_g \left\langle \mathbf{Y}_t, \sum_{i\in[n]} w_i \mathbf{A}_i \right\rangle \geq \frac{\lambda_{\min}\left(\mathbf{A}_g^\top \mathbf{A}_g\right)}{\lambda_{\max}\left(\mathbf{A}_g^\top \mathbf{A}_g\right)} \cdot \kappa_g \geq 1 - O(\epsilon).$$

The remainder of the proof is identical to Lemma 96, where we note the output w satisfies

$$\sum_{i\in[n]} w_i \widetilde{\mathbf{A}}_i \preceq \mathbf{I}, \ \sum_{i\in[n]} w_i \mathbf{A}_i \succeq \frac{1-O(\epsilon)}{\kappa_g}\mathbf{I},$$

which upon rearrangement and adjusting ϵ by a constant yields (9.11). □

By running the modification of Algorithm 29 described for a given level of ν_g, it is straightforward to perform an incremental search on κ_g to find a value satisfying the bound (9.11) as described in Theorem 63. It is simple to verify that the modification in (9.12) is not the dominant runtime in any of Theorems 63 or 62 since the added constraint is diagonal and $\widetilde{\mathbf{A}}_i$ is separable. Hence, for every "level" of ν_g in (9.10) yielding an appropriate risk bound (9.8), we can match this risk bound up to a constant factor while obtaining computational speedups scaling with κ_g.

9.4 Semi-random sparse recovery

Sparse recovery is one of the most fundamental and well-studied inverse problems, with numerous applications in prevalent real-world settings [214]. In its most basic form, we are given an entrywise Gaussian measurement matrix $\mathbf{G} \in \mathbb{R}^{m\times d}$ and measurements $b = \mathbf{G}x^\star$ for an unknown s-sparse $x^\star \in \mathbb{R}^d$; the goal of the problem is to recover $x^\star$. Seminal works by Candès, Romberg, and Tao [104, 106, 105] showed that even when the linear system in $\mathbf{G}$ is extremely underconstrained, recovery is tractable so long as $m = \Omega(s \log d)$. Further they gave a polynomial-time algorithm known as *basis pursuit* based on linear programming recovering $x^\star$ in this regime.

Unfortunately, the runtime of linear programming solvers, while polynomial in the size of the input, can still be prohibitive in many high-dimensional real-world settings. Correspondingly, a number of alternative approaches which may broadly be considered first-order methods have been developed. These methods provably achieve similar recovery guarantees under standard generative models such as Gaussian measurements, with improved runtimes compared to the aforementioned convex programming methods. We refer to these first-order methods through as "fast" algorithms throughout and they may roughly be placed in the following (potentially non-disjoint) categories.

- **Greedy algorithms**, e.g. [382, 437, 411], seek to greedily find elements in the support of the true $x^\star$ using different combinatorial search criteria.
- **Non-convex iterative algorithms**, e.g. [410, 86, 87, 381, 226], directly optimize a (potentially non-convex) objective over a non-convex domain.
- **Convex first-order methods**, e.g. [223, 164, 148, 66, 68, 412, 9] quickly solve the convex objective underlying basis pursuit using first-order methods.

We also note that theoretically, when n is sufficiently large, recent advances by [517, 516], also obtain fast runtimes for the relevant linear programming objective. The fastest IPM for the noiseless sparse recovery objective runs in time[5] $\widetilde{O}(nd + n^{2.5})$ which is nearly-linear when $\mathbf{A}$ is dense and $n \ll d^{2/3}$. For a range of (superlogarithmic, but sublinear) n, these runtimes are no longer nearly-linear; furthermore, these IPMs are second-order and our focus is on designing first-order sparse recovery methods, which are potentially more practical.[6]

It has often been observed empirically that fast first-order methods can have large error, or fail to converge, in real-world settings [166, 280, 446] where convex programming-based algorithms (while potentially computationally cumbersome) perform well statistically [548, 15]. This may be surprising, given that in theory, fast algorithms essentially match the statistical performance of the convex programming-based algorithms under standard generative assumptions. While there have been many proposed explanations for this behavior, one compelling argument is that fast iterative methods used in practice are more brittle to changes in modeling assumptions. We adopt this viewpoint in this chapter, and develop fast sparse recovery algorithms which achieve optimal statistical rates under a *semi-random adversarial model* [85, 219], a popular framework for investigating the robustness of learning algorithms under changes to the data distribution.

Semi-random adversaries. Semi-random adversaries are a framework for reasoning about algorithmic robustness to distributional shift. They are defined in statistical settings, and one common type of semi-random adversary is one which corresponds to generative models where data has been corrupted in a "helpful" or "monotone" way. Such a monotone semi-random adversary takes a dataset from which learning is information-theoretically tractable, and augments it with additional information; this additional information may not break the problem more challenging from an information-theoretic perspective,[7] but may affect the performance of algorithms in other ways. In this chapter, we consider a semi-random adversary which makes the *computational problem* more difficult without affecting the problem information-theoretically, by returning a consistent superset of the unaugmented observations. This contrasts with other adversarial models such as gross corruption [31, 512, 278, 513], where corruptions may be arbitrary, and the corrupted measurements incorrect. It may be surprising that a "helpful" adversary has any implications whatsoever on a learning problem, from either an information-theoretic or computational standpoint.

Typically, convex programming methods for statistical recovery problems are robust to these sorts of perturbations — in brief, this is because constraints to a convex program that are met by an optimum point does not change the optimality of that point. However, greedy and non-convex methods — such as popular practical algorithms for sparse linear regression — can be susceptible to semi-random adversaries. Variants of this phenomenon have been reported in many common

[5] We use $\widetilde{O}$ to hide polylogarithmic factors in problem parameters for brevity of exposition throughout.

[6] We also note that these IPM results do not immediately apply to natural (nonlinear) convex programs for sparse recovery under noisy observations, see Appendix H.2.

[7] There are notable exceptions, e.g. the semi-random stochastic block model of [397].

statistical estimation problems, such as stochastic block models and broadcast tree models [397], PAC learning [84], matrix completion [396, 133], and principal component regression [77]. This can be quite troubling, as semi-random noise can be thought of as a relatively mild form of generative model misspecification: in practice, the true distribution is almost always different from the models considered in theory. Consequently, an algorithm's non-robustness to semi-random noise is suggestive that the algorithm may be more unreliable in real-world settings.

We consider a natural semi-random adversarial model for sparse recovery (see e.g. page 284 of [46]), which extends the standard restricted isometry property (RIP) assumption, which states that applying matrix $\mathbf{A}$ approximately preserves the ℓ_2 norm of sparse vectors. Concretely, throughout the chapter we say matrix $\mathbf{A}$ satisfies the (s, c)-*restricted isometry (RIP) property* if for all s-sparse vectors v,

$$\frac{1}{c}\|v\|_2^2 \le \|\mathbf{A}v\|_2^2 \le c\|v\|_2^2.$$

We state a basic version of our adversarial model here, and defer the statement of the fully general version to Definition 38.[8] We defer the introduction of notation used in the remainder of the chapter to Section 9.5.

Definition 37 (pRIP matrix)**.** *Let $m, n, d \in \mathbb{N}$ be known with $n \ge m$. We say $\mathbf{A} \in \mathbb{R}^{n\times d}$ is ρ-pRIP (planted RIP) if there is an (unknown) $\mathbf{G} \in \mathbb{R}^{m\times d}$ such that each row of $\mathbf{G}$ is also a row of $\mathbf{A}$ and $\frac{1}{\sqrt{m}}\mathbf{G}$ is $(\Theta(s), \Theta(1))$-RIP for appropriate constants, and $\|\mathbf{G}\|_{\max} \le \rho$. When $\rho = \widetilde{O}(1)$ for brevity we say $\mathbf{A}$ is pRIP.*

Under the problem parameterizations used in this chapter, standard RIP matrix constructions satisfy $\rho = \widetilde{O}(1)$ with high probability. For example, when $\mathbf{G}$ is entrywise Gaussian and $m = \Theta(s \log d)$, a tail bound shows that with high probability we may set $\rho = O(\sqrt{\log d})$ to be compatible with the assumptions in Definition 37.

pRIP matrices can naturally be thought of as arising from a semi-random adversarial model as follows. First, an RIP matrix $\mathbf{G} \in \mathbb{R}^{m\times d}$ is generated, for example from a standard ensemble (e.g. Gaussian or subsampled Hadamard). An adversary inspects $\mathbf{G}$, and forms $\mathbf{A} \in \mathbb{R}^{n\times d}$ by reshuffling and arbitrarily augmenting rows of $\mathbf{G}$. Whenever we refer to a "semi-random adversary" in the remainder of the introduction, we mean the adversary provides us a pRIP measurement matrix $\mathbf{A}$.

The key recovery problem we consider in the remainder of this chapter is recovering an unknown s-sparse vector $x^\star \in \mathbb{R}^d$ given measurements $b \in \mathbb{R}^n$ through $\mathbf{A}$. We consider both the *noiseless* or *exact* setting where $b = \mathbf{A}x^\star$ and the *noisy* setting where $b = \mathbf{A}x^\star + \xi$ for bounded ξ. In the noiseless setting in particular, the semi-random adversary hence only gives the algorithm additional *consistent* measurements of the unknown s-sparse vector $x^\star$. In this sense, the adversary is only "helpful," as it returns a superset of information which is sufficient for sparse recovery (formally, this

[8] When clear from context, as it will be throughout the main sections of the remainder of the chapter, s will always refer to the sparsity of a vector $x^\star \in \mathbb{R}^d$ in an exact or noisy recovery problem through $\mathbf{A} \in \mathbb{R}^{n\times d}$. For example, the parameter s in Definition 37 is the sparsity of the vector in an associated sparse recovery problem.

adversary cannot break the standard restricted nullspace condition which underlies the successful performance of convex programming methods). We note n may be much larger than m, i.e. we impose no constraint on how many measurements the adversary adds.

Semi-random sparse recovery in nearly-linear time. We devise algorithms which match the nearly-linear runtimes and optimal recovery guarantees of faster algorithms on fully random data, but which retain both their runtime and the robust statistical performances of convex programming methods against semi-random adversaries. In this sense, our algorithms obtain the "best of both worlds." We discuss and compare more extensively to existing sparse recovery algorithms under Definition 37 in the following section. We first state our result in the noiseless observation setting.

Theorem 64 (informal, see Theorem 66)**.** *Let $x^\star \in \mathbb{R}^d$ be an unknown s-sparse vector. Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be pRIP. There is an algorithm, which given $\mathbf{A}$ and $b = \mathbf{A}x^\star$, runs in time $\widetilde{O}(nd)$, and outputs $x^\star$ with high probability.*

Since our problem input is of size nd, our runtime in Theorem 64 is nearly-linear in the problem size. We also extend our algorithm to handle the noisy observation setting, where we are given perturbed linear measurements of $x^\star$ from a pRIP matrix.

Theorem 65 (informal, see Theorem 67)**.** *Let $x^\star \in \mathbb{R}^d$ be an unknown s-sparse vector, and let $\xi \in \mathbb{R}^n$ be arbitrary. Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be pRIP. There is an algorithm, which given $\mathbf{A}$ and $b = \mathbf{A}x^\star + \xi$, runs in time $\widetilde{O}(nd)$, and with high probability outputs x satisfying*

$$\|x - x^\star\|_2 \le O\left(\frac{1}{\sqrt{m}} \left\|\xi_{(m)}\right\|_2\right),$$

where $\xi_{(m)}$ denotes the largest m entries of ξ by absolute value, with other coordinates set to 0.

The error scaling of Theorem 65 is optimal in the semi-random setting. Indeed, when there is no semi-random noise, the guarantees of Theorem 65 exactly match the standard statistical guarantees in the fully-random setting for sparse recovery, up to constants; for example, when $\mathbf{A} = \sqrt{m}\mathbf{I}$ (which is clearly RIP, in fact an exact isometry, after rescaling), it is information-theoretically impossible to obtain a better ℓ_2 error.[9] The error bound of Theorem 65 is similarly optimal in the semi-random setting because in the worst case, the largest entries of ξ may correspond to the rows of the RIP matrix from which recovery is information-theoretically possible.

Performance of existing algorithms. To contextualize Theorems 64 and 65, we discuss the performance of existing algorithms for sparse recovery under the semi-random adversarial model of Definition 37. First, it can be easily verified that our semi-random adversary never changes the

[9] In the literature it is often standard to scale down the sensing matrix $\mathbf{A}$ by $\sqrt{m}$; this is why our error bound is similarly scaled. However, this scaling is more convenient for our analysis, especially when stating weighted results.

information-theoretic tractability of sparse recovery. In the noiseless setting for example, the performance of the minimizer to the classical convex program based on ℓ_1 minimization,

$$\min_{\mathbf{A}x=b} \|x\|_1 ,$$

is unchanged in the presence of pRIP matrices (as $x^\star$ is still consistent with the constraint set, and in particular a RIP constraint set), and hence the semi-random problem can be solved in polynomial time via convex programming. This suggests the main question we address: can we design a near-linear time algorithm obtaining optimal statistical guarantees under pRIP measurements?

As alluded to previously, standard greedy and non-convex methods we have discussed may fail to converge to the true solution against appropriate semi-random adversaries generating pRIP matrices. We give explicit counterexamples to several popular methods such as orthogonal matching pursuit and iterative hard thresholding in Appendix H.2. Further, it seems likely that similar counterexamples also break other, more complex methods commonly used in practice, such as matching pursuit [382] and CoSaMP [410].

Additionally, while fast "convex" iterative algorithms (e.g. first-order methods for solving objectives underlying polynomial-time convex programming approaches) will never fail to converge to the correct solution given pRIP measurements, the analyses which yield fast runtimes for these algorithms [412, 9] rely on properties such as restricted smoothness and strong convexity (a specialization of standard conditioning assumptions to numerically sparse vectors). These hold under standard generative models but again can be broken by pRIP measurements; consequently, standard convergence analyses of "convex" first-order methods may yield arbitrarily poor rates.

One intuitive explanation for why faster methods fail is that they depend on conditions such as incoherence [196] or the restricted isometry property [106], which can be destroyed by a semi-random adversary. For instance, RIP states that if S is any subset of $m = \Theta(s)$ columns of $\mathbf{A}$, and $\mathbf{A}_S$ is the submatrix formed by taking those columns of $\mathbf{A}$, then $\mathbf{A}_S^\top \mathbf{A}_S$ is an approximate isometry (i.e. it is well-conditioned). While it is well-known that RIP is satisfied with high probability when $\mathbf{A}$ consists of $\Theta(s \log d)$ Gaussian rows, it is not too hard to see that augmenting $\mathbf{A}$ with additional rows can easily ruin the condition number of submatrices of this form. In contrast, convex methods work under weaker assumptions such as the restricted nullspace condition, which cannot be destroyed by the augmentation used by pRIP matrices. Though these weaker conditions (e.g. the restricted nullspace condition) suffice for algorithms based on convex programming primitives, known analyses of near-linear time "fast" algorithms require additional instance structure, such as incoherence or RIP. Thus, it is plausible that fast algorithms for sparse recovery are less robust to the sorts of distributional changes that may occur in practice.

Beyond submatrices. Our methods naturally extend to a more general setting (see Definition 38, wherein we define "weighted RIP" (wRIP) matrices, a generalization of Definition 37). Rather than assuming there is a RIP submatrix $\mathbf{G}$, we only assume that there is a (nonnegative) reweighting of

the rows of $\mathbf{A}$ so that the reweighted matrix is "nice," i.e. it satisfies RIP. Definition 37 corresponds to the special case of this assumption where the weights are constrained to be either 0 or 1 (and hence must indicate a subset of rows). In our technical sections (Sections 9.6 and 9.7), our results are stated for this more general semi-random model, i.e. sparse recovery from wRIP measurements. For simplicity of exposition, throughout the introduction, we mainly focus on the simpler pRIP sparse recovery setting described following Definition 37.

Towards instance-optimal guarantees. While the performance of the algorithms in Theorems 64 and 65 is already nearly-optimal in the worst case semi-random setting, one can still hope to improve our runtime and error bounds in certain scenarios. Our formal results, Theorems 66 and 67, provide these types of fine-grained instance-optimal guarantees in several senses.

In the noiseless setting (Theorem 66), if it happens to be that the entire matrix $\mathbf{A}$ is RIP (and not just $\mathbf{G}$), then standard techniques based on subsampling the matrix can be used to solve the problem in time $\widetilde{O}(sd)$ with high probability. For example, if $\mathbf{A}$ is pRIP where, following the notation of Definition 37, $\mathbf{G}$ is entrywise Gaussian, and the adversary chose to give us additional Gaussian rows, one could hope for a runtime improvement (simply by ignoring the additional measurements given). Theorem 66 obtains a runtime which smoothly interpolates between the two regimes of a worst-case adversary and an adversary which gives us additional random measurements from an RIP ensemble. Roughly speaking, if there exists a (a priori unknown) submatrix of $\mathbf{A}$ of $m \gg \widetilde{\Theta}(s)$ rows which is RIP, then we show that our algorithm runs in *sublinear* time $\widetilde{O}(nd \cdot \frac{s}{m})$, which is $\widetilde{O}(sd)$ when $m \approx n$. We show this holds in our weighted semi-random model (under wRIP measurements, Definition 38) as well, where the runtime depends on the ratio of the ℓ_1 norm of the (best) weight vector to its ℓ_∞ norm, a continuous proxy for the number of RIP rows under pRIP.

We show a similar interpolation holds in the noisy measurement setting, both in the runtime sense discussed previously, and also in a statistical sense. In particular, Theorem 67 achieves (up to logarithmic factors) the same interpolating runtime guarantee of Theorem 66, but further attains a squared ℓ_2 error which is roughly the average of the m largest elements of the squared noise vector ξ (see the informal statement in Theorem 65). This bound thus improves as $m \gg \widetilde{\Theta}(s)$; we show it extends to weighted RIP matrices (Definition 38, our generalization of Definition 37) in a natural way depending on the ℓ_∞-ℓ_1 ratio of the weights.

9.4.1 Our techniques

Our overall approach for semi-random sparse recovery is fairly different from two recent works in the literature which designed fast iterative methods succeeding under a semi-random adversarial model: [133] and Section 9.3. In particular, these two algorithms were both based on the following natural framework, which separates the "planted learning" problem (e.g. identifying the planted benign matrix) from the "estimation" task (e.g. solving a linear system or regression problem).

1. Compute a set of weights for the data (in linear regression for example, these are weights on

each of the rows of a measurement matrix $\mathbf{A}$), such that after re-weighting, the data fits the input assumptions of a fast iterative method which performs well on a fully random instance.

2. Apply said fast iterative algorithm on the reweighted data in a black-box manner.

To give a concrete example, Section 9.3 studied the standard problem of *overdetermined* linear regression with a semi-random adversary, where a measurement matrix $\mathbf{A}$ is received with the promise that $\mathbf{A}$ contains a "well-conditioned core" $\mathbf{G}$. The algorithm of Section 9.3 first learned a re-weighting of the rows of $\mathbf{A}$ by a diagonal matrix $\mathbf{W}^{\frac{1}{2}}$, such that the resulting system in $\mathbf{A}^\top\mathbf{W}\mathbf{A}$ is well-conditioned and hence can be solved using standard first-order methods.

In the case of semi-random sparse recovery, there appear to be significant barriers to reweighting approaches (which we will shortly elaborate on). We take a novel direction that involves designing a new nearly-linear time iterative method for sparse recovery tailored to the geometry of the problem.

Why not (globally) reweight the rows? There are several difficulties immediately encountered when trying to use the aforementioned reweighting framework for sparse recovery. First of all, there is no effective analog of condition number for an underdetermined linear system. The standard assumption on the measurement matrix $\mathbf{A}$ to make sparse recovery tractable for fast iterative methods is that $\mathbf{A}$ satisfies RIP, i.e. $\mathbf{A}$ is roughly an isometry when restricted to $O(s)$-sparse vectors. However, RIP is NP-hard to verify [57] and this may suggest that it is computationally hard to try, say, learning a reweighting of the rows of $\mathbf{A}$ such that the resulting reweighted matrix is guaranteed to be RIP (though it would be very interesting if this were achievable). More broadly, almost all explicit conditions (e.g. RIP, incoherence etc.) which make sparse recovery tractable for fast algorithms are conditions about subsets of the *columns* of $\mathbf{A}$. Thus, any approach which reweights rows of $\mathbf{A}$ such that column subsets of the reweighted matrix satisfy an appropriate condition results in optimization problems that seems challenging to solve in nearly-linear time.

We circumvent these difficulties in two steps. First, we propose a new analysis of an iterative method based on (reweighted) projected gradient descent, which obtains a fast convergence rate whenever each step satisfies certain locally verifiable properties. Next, our algorithm computes a sequence of *local reweightings* (which can be different in each iteration) of the rows of our measurement matrix, such that each local reweighting satisfies our requisite progress conditions for that step. We use the existence of a global reweighting satisfying RIP to demonstrate that each local reweighting subproblem has a good solution, and design an efficient method for computing each local reweighting. Our framework of bypassing hardness of computing a global reweighting to recover planted statistical information, by instead designing an iterative method capable of exploiting local reweightings with (computationally tractable) certifiable progress conditions, is quite general, and we hope it will find uses in semi-random settings beyond our particular problem.

The geometry of sparse recovery. We now explain our new approach, and how we derive deterministic conditions on the steps of an iterative method which certify progress by exploiting the geometry of sparse recovery. We focus on the clean observation setting in this technical overview.

Suppose that we wish to solve a sparse regression problem $\mathbf{A}x^\star = b$ where $x^\star$ is s-sparse, and we are given $\mathbf{A}$ and b. To fix a scale, suppose for simplicity that we know $\|x^\star\|_1 = \sqrt{s}$ and $\|x^\star\|_2 = 1$. Also, assume for the purpose of conveying intuition that $\mathbf{A}$ is pRIP, and that the planted matrix $\mathbf{G}$ in Definition 37 is an entrywise random Gaussian matrix.

We next consider a natural family of iterative algorithms for solving the system $\mathbf{A}x^\star = b$. For simplicity, assume that our current iterate is $x_t = 0$ (such that b is the residual vector $b - \mathbf{A}x_t$; more generally, in the following description b will be replaced by the current residuals). Inspired by gradient methods for solving regression, we consider "first-order" algorithms of the following form:

$$\begin{aligned} y_t &\leftarrow x_t + \mathbf{A}^\top u \\ x_{t+1} &\leftarrow \mathbf{P}(y_t) \end{aligned} \tag{9.15}$$

where $u \in \mathbb{R}^n$ are coefficients to be computed (for example, a natural attempt could be to set u to be a multiple of $\mathbf{A}x_t - b$, resulting in the step being a scaled gradient of $\|\mathbf{A}x - b\|_2^2$ at x_t) and $\mathbf{P}$ denotes projection onto the (convex) ℓ_1 ball of radius $\sqrt{s}$. Our goal will be to make constant-factor progress in terms of $\|x_t - x^\star\|_2$ in each application of (9.15), to yield a $\widetilde{O}(1)$ iteration method. Note that x_{t+1} must at minimum make constant factor progress in the direction $x^\star - x_t$ (in terms of decreasing the projection onto this direction) if we hope to make constant factor progress in overall distance to $x^\star$. In other words, we must have

$$\langle x_{t+1} - x_t, x^\star - x_t \rangle = \Omega(\|x^\star - x_t\|_2^2) = \Omega(1)\,.$$

First, observe that by the definition of our step and shifting so that $x_t = 0$, the point y_t satisfies

$$\langle y_t - x_t, x^\star - x_t \rangle = \langle \mathbf{A}^\top u, x^\star - x_t \rangle = u^\top (b - \mathbf{A}x_t) = u^\top b,$$

so to obtain a corresponding progress lower bound for the move to y_t, we require

$$u^\top b = \Omega(1). \tag{9.16}$$

Of course, this condition alone is not enough, for two reasons: the step also moves in directions orthogonal to $x^\star - x_t$, and we have not accounted for the projection step $\mathbf{P}$. If all of the rows of $\mathbf{A}$ were random, then standard Gaussian concentration implies that we expect $\|b\|_2 = \sqrt{\frac{n}{d}}$, and thus to satisfy (9.16) we need $\|u\|_2 \geq \sqrt{\frac{d}{n}}$. This also implies

$$\|y_t - x_t\|_2 = \left\|\mathbf{A}^\top u\right\|_2 \approx \sqrt{\frac{d}{n}},$$

since for Gaussian $\mathbf{A}$, we expect that its rows are roughly orthogonal. Moreover, since $\sqrt{\frac{d}{n}} \gg 1$ in

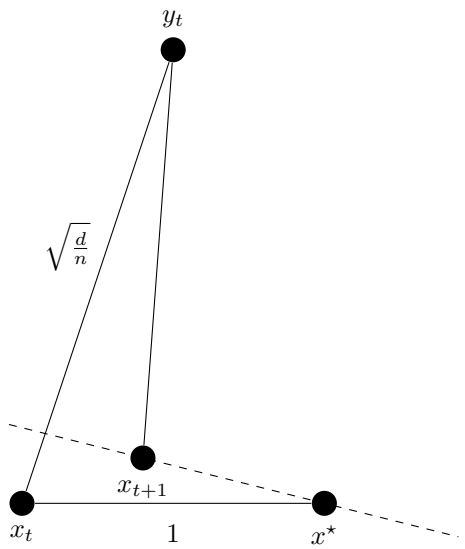


Figure 9.1: The effect of ℓ_1 projection on iterate progress. The dashed line represents a facet of the ℓ_1-ball around x_t of radius $\|x_t - x^\star\|_1$.

the typical underconstrained setting, almost all of the step from x_t to y_t is actually orthogonal to the desired "progress direction" parallel to $x^\star - x_t$. This appears to be a serious problem, because in order to argue that our algorithm makes progress, we need to argue that the ℓ_1 projection step $x_{t+1} = \mathbf{P}(y_t)$ "undoes" this huge ℓ_2 movement orthogonal to the progress direction (see Figure 1).

Our key geometric insight is that the ℓ_1 ball is very thin in most directions, but is thick in directions that correspond to "numerically sparse" vectors, namely vectors with bounded ℓ_1 to ℓ_2 ratios. Crucially, the movement of our step in the progress direction parallel to $x^\star - x_t$ is numerically sparse because $x^\star - x_t$ is itself $O(s)$-numerically sparse by assumption. However, for Gaussian $\mathbf{A}$, the motion in the subspace orthogonal to the progress direction ends up being essentially random (and thus is both not sparse, and is ℓ_∞-bounded). Formally, we leverage this decomposition to show that the ℓ_1 projection keeps most of the forward movement in the progress direction $x^\star - x_t$ but effectively filters out much of the orthogonal motion, as demonstrated by the figure below.

This geometric intuition is the basis for the deterministic conditions we require of the steps of our iterative method, to guarantee that it is making progress. More precisely, the main condition we require of our step in each iteration, parameterized by the coefficients u used to induce (9.15), is that $y_t - x_t$ has a "short-flat" decomposition into two vectors $p + e$ where

$$\|p\|_2 = O(1) \text{ is "short" and } \|e\|_\infty = O\left(\frac{1}{\sqrt{s}}\right) \text{ is "flat".}$$

Intuitively, the short component is a proxy for the progress component (in the direction towards $x^\star$), and the flat component is a proxy for the orthogonal component, which should be filtered out by the

projection step onto an ℓ_1 ball. The above bounds are rescaled appropriately in our actual method. We state these requirements formally (in the clean observation case, for example) in Definition 39, where we define a "step oracle" which is guaranteed to make constant-factor progress towards $x^\star$. By combining the above short-flat decomposition requirement with a progress requirement such as (9.16), we can show that as long as we can repeatedly implement a satisfactory step, our method is guaranteed to converge rapidly.

This framework for sparse recovery effectively reduces the learning problem to the problem of implementing an appropriate step oracle. Note that a valid step always exists by only looking at the Gaussian rows. To complete our algorithm, we give an implementation of this step oracle (i.e. actually solving for a valid step) which runs in nearly-linear time even when the data is augmented by a semi-random adversary (that is, our measurement matrix is pRIP or wRIP rather than RIP), and demonstrate that our framework readily extends to the noisy observation setting. Our step oracle is motivated by stochastic gradient methods. In particular, we track potentials corresponding to the progress made by our iterative method and the short-flat decomposition, and show that uniformly sampling rows of a pRIP (or wRIP) matrix $\mathbf{A}$ and taking steps in the direction of these rows which maximally improve our potentials rapidly implements a step oracle.

9.4.2 Related work

Sparse recovery. Sparse recovery, and variants thereof, are fundamental statistical and algorithmic problems which have been studied in many of settings, including signal processing [355, 469, 196, 86, 59], and compressed sensing [104, 106, 105, 195, 462]. A full review of the extensive literature on sparse recovery is out of the scope of the present chapter; we refer the reader to e.g. [214, 165, 335, 471] for more extensive surveys.

Within the literature on sparse recovery, arguably the closest line of work to ours is the line of work which attempts to design efficient algorithms which work when the restricted condition number of the sensing or measurement matrix is large. Indeed, it is known that many nonconvex methods fail when the restricted condition number of the sensing matrix is far from 1, which is often the case in applications [280]. To address this, several works [280, 471] have designed novel non-convex methods which still converge, when the restricted condition number of the matrix is much larger than 1. However, these methods still require that the restricted condition number is constant or bounded, whereas in our setting, the restricted condition number could be arbitrarily large due to the generality of the semi-random adversary assumption.

Another related line of work considers the setting where, instead of having a sensing matrix with rows which are drawn from an isotropic Gaussian, have rows drawn from $\mathcal{N}(0, \mathbf{\Sigma})$, for some potentially ill-conditioned $\mathbf{\Sigma}$ [79, 453, 514, 280, 324, 154, 546, 70, 317]. This setting is related to our semi-random adversarial model, in that the information-theoretic content of the problem does not change, but obtaining efficient algorithms which match the optimal statistical rates is very

challenging. However, there does not appear to be any further concrete connection between this "ill-conditioned covariance" setting and the semi-random model we consider in this work. Indeed, the ill-conditioned setting appears to be qualitatively much more difficult for algorithms: in particular, [317] shows evidence that there are in fact no efficient algorithms that achieve the optimal statistical rates, without additional assumptions on $\mathbf{\Sigma}$. In contrast in the semi-random setting, polynomial-time convex programming approaches, while having potentially undesirable superlinear runtimes, still obtain optimal statistical guarantees.

Finally as discussed earlier in the introduction, there is a large body of work on efficient algorithms for sparse recovery in an RIP matrix (or a matrix satisfying weaker or stronger analogous properties). These works e.g. [104, 106, 105, 382, 437, 411, 410, 86, 87, 381, 226, 223, 164, 148, 66, 68, 412, 9] are typically based on convex programming or different iterative first-order procedures.

Semi-random models. Context for semi-random models was previously discussed in Section 9.3. We also note that the well-studied *Massart noise* model in PAC learning [386] can be thought of as a semi-random variant of the random classification noise model. However, this setting appears to be quite different from ours: in particular, it was not until quite recently that polynomial-time algorithms were even known to be achievable for a number of fundamental learning problems under Massart noise [173, 189, 126, 183, 184, 187, 190, 174, 544].

9.5 Preliminaries: semi-random sparse recovery

General notation. We let $[n] := \{i \in \mathbb{N}, 1 \leq i \leq n\}$. The ℓ_p norm of a vector is denoted $\|\cdot\|_p$, and the sparsity (number of nonzero entries) of a vector is denoted $\|\cdot\|_0$. For a vector $v \in \mathbb{R}^d$ and $k \in [d]$, we let $v_{(k)}$ be the vector equalling v on the largest k entries of v in absolute value (with other coordinates zeroed out). The all-zeroes vector of dimension n is denoted 0_n. The nonnegative probability simplex in dimension n (i.e. $\|p\|_1 = 1$, $p \in \mathbb{R}^n_{\geq 0}$) is denoted Δ^n.

For mean $\mu \in \mathbb{R}^d$ and positive semidefinite covariance $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$, $\mathcal{N}(\mu, \mathbf{\Sigma})$ denotes the corresponding multivariate Gaussian. $i \sim_{\text{unif.}} S$ denotes a uniform random sample from set S. For $N \in \mathbb{N}$ and $p \in \Delta^n$ we we use $\text{Multinom}(N, p)$ to denote the probability distribution corresponding to N independent draws from $[n]$ as specified by p.

Sparsity. We say v is *s-sparse* if $\|v\|_0 \leq s$. We define the *numerical sparsity* of a vector by $\text{NS}(v) := \|v\|_1^2 / \|v\|_2^2$. Note that from the Cauchy-Schwarz inequality, if $\|v\|_0 \leq s$, then $\text{NS}(v) \leq s$.

Matrices. Matrices are in boldface throughout. The zero and identity matrix of appropriate dimension from context are $\mathbf{0}$ and $\mathbf{I}$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we let its rows be $\mathbf{A}_{i:}$, $i \in [n]$ and its columns be $\mathbf{A}_{:j}$, $j \in [d]$. The set of $d \times d$ symmetric matrices is $\mathbb{S}^d$, and its positive definite and positive semidefinite restrictions are $\mathbb{S}^d_{\succ \mathbf{0}}$ and $\mathbb{S}^d_{\geq 0}$. We use the Loewner partial order $\preceq$ on $\mathbb{S}^d$. The largest entry of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is denoted $\|\mathbf{A}\|_{\max} := \max_{i \in [n], j \in [d]} |\mathbf{A}_{ij}|$. When a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is clear from context, we refer to its rows as $\{a_i\}_{i \in [n]}$.

Short-flat decompositions. Throughout we frequently use the notion of "short-flat decompositions." We say $v \in \mathbb{R}^d$ has a (C_2, C_∞) *short-flat decomposition* if $v = p + e$ for some $e \in \mathbb{R}^d$ with $\|e\|_2 \le C_2$ and $p \in \mathbb{R}^d$ with $\|p\|_\infty \le C_\infty$. Further, we use $\text{trunc}(v, c) \in \mathbb{R}^d$ for $c \in \mathbb{R}_{\ge 0}$ to denote the vector which coordinatewise $[\text{trunc}(v, c)]_i = \text{sgn}(v_i) \max(|v_i| - c, 0)$ (i.e. the result of adding or subtracting at most c from each coordinate to decrease the coordinate's magnitude). Note that $v \in \mathbb{R}^d$ has a (C_2, C_∞) short-flat decomposition if and only if $\|\text{trunc}(v, C_\infty)\|_2 \le C_2$ (in which case $p = \text{trunc}(v, C_\infty)$ and $e = v - p$ is such a decomposition).

Restricted isometry property. We say that matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ satisfies the (s, c)*-restricted isometry property (RIP)* or (more concisely) $\mathbf{A}$ *is* (s, c)*-RIP*, if for all s-sparse vectors $v \in \mathbb{R}^d$,

$$\frac{1}{c} \|v\|_2^2 \le \|\mathbf{A}v\|_2^2 \le c \|v\|_2^2 .$$

9.6 Exact recovery

In this section, we give an algorithm for solving the underconstrained linear system $\mathbf{A}x^\star = b$ given the measurement matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ (for $n \le d$) and responses $b \in \mathbb{R}^n$ (i.e. noiseless or "exact" regression), and $x^\star$ is s-sparse. Our algorithm succeeds when $\mathbf{A}$ is weighted RIP (wRIP), i.e. it satisfies Definition 38, a weighted generalization of Definition 37.

Definition 38 (wRIP matrix)**.** *Let $w_\infty^\star \in [0, 1]$. We say $\mathbf{A} \in \mathbb{R}^{n \times d}$ is $(\rho, w_\infty^\star)$-wRIP if $\|\mathbf{A}\|_{\max} \le \rho$, and there exists a weight vector $w^\star \in \Delta^n$ satisfying $\|w^\star\|_\infty \le w_\infty^\star$, such that $\mathbf{diag}\,(w^\star)^{\frac{1}{2}} \mathbf{A}$ is $(\Theta(s), \Theta(1))$-RIP for appropriate constants. When $\rho = \widetilde{O}(1)$ for brevity we say $\mathbf{A}$ is $w_\infty^\star$-wRIP.*

As discussed after Definition 37, a wRIP matrix can be thought of as arising from a "semi-random model" because it strictly generalizes our previously-defined pRIP matrix notion in Definition 37 with $w_\infty^\star = \frac{1}{m}$, by setting $w^\star$ to be $\frac{1}{m}$ times the zero-one indicator vector of rows of $\mathbf{G}$. The main result of this section is the following theorem regarding sparse recovery with wRIP matrices.

Theorem 66. *Let $\delta \in (0, 1)$, $r > 0$, and suppose $R_0 \ge \|x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$. Then with probability at least $1 - \delta$, Algorithm 54 using Algorithm 55 as a step oracle takes as input a $(\rho, w_\infty^\star)$-wRIP matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $b = \mathbf{A}x^\star$, and computes $\hat{x}$ satisfying $\|\hat{x} - x^\star\|_2 \le r$ in time*

$$O\left(\left(nd \log^3(nd\rho) \log\left(\frac{1}{\delta} \cdot \log \frac{R_0}{r}\right) \log\left(\frac{R_0}{r}\right)\right) \cdot \left(w_\infty^\star s \rho^2 \log d\right)\right).$$

Under the wRIP assumption, Theorem 66 provides a natural interpolation between the fully random and semi-random generative models. To build intuition, if a pRIP matrix contains a planted RIP matrix with $\widetilde{O}(s)$ rows (the information-theoretically minimum size), then by setting $w_\infty^\star \approx \frac{1}{\widetilde{O}(s)}$, we obtain a near-linear runtime of $\widetilde{O}(nd)$. However, in the fully random regime where $w_\infty^\star \approx \frac{1}{n}$ (i.e. all of $\mathbf{A}$ is RIP), the runtime improves $\widetilde{O}(sd)$ which is sublinear for $n \gg s$.

The roadmap of our algorithm and its analysis are as follows.

1. In Section 9.6.1, we give an algorithm (Algorithm 54) which iteratively halves an upper bound on the radius to $x^\star$, assuming that either an appropriate step oracle (see Definition 39) based on short-flat decompositions can be implemented for each iteration, or we can certify that the input radius bound is now too loose. This algorithm is analyzed in Lemma 170.

2. We state in Assumption 11 a set of conditions on a matrix-vector pair $(\mathbf{A}, \Delta)$ centered around the notion of short-flat decompositions, which suffice to provide a sufficient step oracle implementation with high probability in nearly-linear time. In Section 9.6.2 we analyze this implementation (Algorithm 55) in the proof of Lemma 168 assuming the inputs satisfy Assumption 11.

3. In Section 9.6.3, we show Assumption 11, with appropriate parameters, follows from $\mathbf{A}$ being wRIP. This is a byproduct of a general equivalence we demonstrate between RIP, restricted conditioning measures used in prior work [9], and short-flat decompositions.

9.6.1 Radius contraction using step oracles

In this section, we provide and analyze the main loop of our overall algorithm for proving Theorem 66. This procedure, HalfRadiusSparse, takes as input an s-sparse vector x_{in} and a radius bound $R \geq \|x_{\text{in}} - x^\star\|_2$ and returns an s-sparse vector x_{out} with the guarantee $\|x_{\text{out}} - x^\star\|_2 \leq \frac{1}{2}R$. As a subroutine, it requires access to a "step oracle" $\mathcal{O}_{\text{step}}$, which we implement in Section 9.6.2 under certain assumptions on the matrix $\mathbf{A}$.

Definition 39 (Step oracle)**.** *We say that $\mathcal{O}_{\text{step}}$ is a $(C_{\text{prog}}, C_2, \delta)$-step oracle for $\Delta \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$, if the following holds. Whenever there is $v \in \mathbb{R}^d$ with $\frac{1}{4} \leq \|v\|_2 \leq 1$ and $\|v\|_1 \leq 2\sqrt{2s}$ such that $\Delta = \mathbf{A}v$, with probability $\geq 1 - \delta$, $\mathcal{O}_{\text{step}}$ returns $w \in \mathbb{R}^n_{\geq 0}$ such that the following two conditions hold. First,*

$$\sum_{i \in [n]} w_i \Delta_i^2 \geq C_{\text{prog}}. \tag{9.17}$$

Second, there exists a $(C_2, \frac{C_{\text{prog}}}{6\sqrt{s}})$ short-flat decomposition of $\mathbf{A}^\top \mathbf{diag}\,(w)\,\Delta$:

$$\left\| \text{trunc}\left(\mathbf{A}^\top \mathbf{diag}\,(w)\,\Delta, \frac{C_{\text{prog}}}{6\sqrt{s}} \right) \right\|_2 \leq C_2. \tag{9.18}$$

Intuitively, (9.18) guarantees that we can write $\gamma = p + e$ where p denotes a "progress" term which we require to be sufficiently short in the ℓ_2 norm, and e denotes an "error" term which we require to be small in ℓ_∞. We prove that under certain assumptions on the input $\mathbf{A}$ (stated in Assumption 11 below), we can always implement a step oracle with appropriate parameters.

Assumption 11. *The matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ satisfies the following. There is a weight vector $w^\star \in \Delta^n$ satisfying $\|w^\star\|_\infty \le w^\star_\infty$, a constant L, $\rho \ge 1$, and a constant K (which may depend on L) such that for all $v \in \mathbb{R}^d$ with $\frac{1}{4} \le \|v\|_2 \le 1$ and $\|v\|_1 \le 2\sqrt{2s}$ we have, defining $\Delta = \mathbf{A}v$:*

1. *$\mathbf{A}$ is entrywise bounded by $\pm\rho$, i.e. $\|\mathbf{A}\|_{\max} \le \rho$.*
2. $$\frac{1}{L} \le \sum_{i \in [n]} w_i^\star \Delta_i^2 \le L. \tag{9.19}$$
3. *For $\mathbf{W}^\star := \mathbf{diag}\,(w^\star)$, there is a $(L, \frac{1}{K\sqrt{s}})$ short-flat decomposition of $\mathbf{A}^\top \mathbf{W}^\star \Delta = \sum_{i \in [n]} w_i^\star \Delta_i a_i$:*
$$\left\| \mathrm{trunc}\left(\mathbf{A}^\top \mathbf{W}^\star \Delta, \frac{1}{K\sqrt{s}} \right) \right\|_2 \le L. \tag{9.20}$$

Our Assumption 11 may also be stated in a scale-invariant way (i.e. with (9.19), (9.20) scaling with $\|v\|_2$), but it is convenient in our analysis to impose a norm bound on v. Roughly, the second property in Assumption 11 is (up to constant factors) equivalent to the "restricted strong convexity" and "restricted smoothness" assumptions of [9], which were previously shown for specific measurement matrix constructions such as random Gaussian matrices. The use of the third property in Assumption 11 (the existence of short-flat decompositions for numerically sparse vectors) in designing an efficient algorithm is a key contribution of our work. Interestingly, we show in Section 9.6.3 that these assumptions are up to constant factors equivalent to RIP.

More specifically, we show that when $\mathbf{A}$ is wRIP, we can implement a step oracle for $\Delta = \mathbf{A}v$ where $v = \frac{1}{R}(x - x^\star)$ for some iterate x of Algorithm 54, which either makes enough progress to advance the algorithm or certifies that v is sufficiently short, by using numerical sparsity properties of v. We break this proof into two parts. In Lemma 168, we show that Assumption 11 suffices to implement an appropriate step oracle; this is proven in Section 9.6.2. In Lemma 169, we then demonstrate the wRIP assumption with appropriate parameters implies our measurement matrix satisfies Assumption 11, which we prove by way of a more general equivalence in Section 9.6.3.

Lemma 168. *Suppose $\mathbf{A}$ satisfies Assumption 11. Algorithm 55 is a $(C_{\mathrm{prog}}, C_2, \delta)$ step oracle* StepOracle *for $(\Delta, \mathbf{A})$ with $C_{\mathrm{prog}} = \Omega(1)$, $C_2 = O(1)$ running in time*
$$O\left(\left(nd \log^3(nd\rho) \log \frac{1}{\delta} \right) \cdot \left(w^\star_\infty s \rho^2 \log d \right) \right).$$

Lemma 169. *Suppose $\mathbf{A} \in \mathbb{R}^{n \times d}$ is $(\rho, w^\star_\infty)$-wRIP with a suitable choice of constants in the RIP parameters in Definition 38. Then, $\mathbf{A}$ also satisfies Assumption 11.*

We now give our main algorithm HalfRadiusSparse, assuming access to the step oracle $\mathcal{O}_{\mathrm{step}}$ from Section 9.6.2 with appropriate parameters, and that $\mathbf{A}$ obeys Assumption 11.

Algorithm 54: HalfRadiusSparse$(x_{\text{in}}, R, \mathcal{O}_{\text{step}}, \delta, \mathbf{A}, b)$

Input: s-sparse $x_{\text{in}} \in \mathbb{R}^d$, $R \geq \|x_{\text{in}} - x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$, $(C_{\text{prog}}, C_2, \delta)$-step oracle $\mathcal{O}_{\text{step}}$ for all $(\Delta, \mathbf{A})$ with $\Delta \in \mathbb{R}^n$, $\delta \in (0,1)$, $\mathbf{A} \in \mathbb{R}^{n \times d}$, $b = \mathbf{A}x^\star \in \mathbb{R}^n$;

Output: s-sparse vector x_{out} that satisfies $\|x_{\text{out}} - x^\star\|_2 \leq \frac{1}{2}R$ with probability $\geq 1 - T\delta$;

Set $x_0 \leftarrow x_{\text{in}}$, $\mathcal{X} \leftarrow \{x \in \mathbb{R}^d \mid \|x - x_{\text{in}}\|_1 \leq \sqrt{2s}R\}$ $T \leftarrow \left\lceil \frac{6C_2^2}{C_{\text{prog}}^2} \right\rceil$, $\eta \leftarrow \frac{C_{\text{prog}}}{2C_2^2}$;

for $0 \leq t \leq T-1$ **do**

  $w_t \leftarrow \mathcal{O}_{\text{step}}(\Delta_t, \mathbf{A})$ for $\Delta_t \leftarrow \frac{1}{R}(\mathbf{A}x_t - b)$, $\gamma_t \leftarrow \mathbf{A}^\top \mathbf{diag}\,(w_t)\,\Delta_t = \sum_{i \in [n]} [w_t]_i [\Delta_t]_i a_i$;

  if $\sum_{i \in [n]} [w_t]_i [\Delta_t]_i^2 < C_{\text{prog}}$ or $\left\| \text{trunc}(\gamma_t, \frac{C_{\text{prog}}}{6\sqrt{s}}) \right\|_2 > C_2$ **then**

    Return: $x_{\text{out}} \leftarrow [x_t]_{(s)}$

  else $x_{t+1} \leftarrow \text{argmin}_{x \in \mathcal{X}} \|x - x_t - \eta R \gamma_t\|_2$;

Return: $x_{\text{out}} \leftarrow [x_t]_{(s)}$

Lemma 170. *Assume* $\mathbf{A}$ *satisfies Assumption 11. With probability at least* $1 - T\delta$*, Algorithm 54 succeeds (i.e.* $\|x_{out} - x^\star\|_2 \leq \frac{1}{2}R$*).*

Proof. Throughout this proof, condition on the event that all step oracles succeed (which provides the failure probability via a union bound). We first observe that $x^\star \in \mathcal{X}$ because of Cauchy-Schwarz, the $2s$-sparsity of $x_{\text{in}} - x^\star$, and the assumption $\|x_{\text{in}} - x^\star\|_2 \leq R$.

Next, we show that in every iteration t of Algorithm 54,

$$\|x_{t+1} - x^\star\|_2^2 \leq \left(1 - \frac{C_2^2}{2C_{\text{prog}}^2}\right) \|x_t - x^\star\|_2^2 . \tag{9.21}$$

As $x^\star \in \mathcal{X}$, the optimality conditions of x_{t+1} as minimizing $\|x - (x_t - \eta R \gamma_t)\|_2^2$ over $\mathcal{X}$ imply

$$\begin{aligned} &2 \langle x_{t+1} - x_t + \eta R \gamma_t, x_{t+1} - x^\star \rangle \leq 0 \\ \implies &\|x_t - x^\star\|_2^2 - \|x_{t+1} - x^\star\|_2^2 \geq 2\eta R \langle \gamma_t, x_{t+1} - x^\star \rangle + \|x_t - x_{t+1}\|_2^2 . \end{aligned} \tag{9.22}$$

Hence, it suffices to lower bound the right-hand side of the above expression. Let $\gamma_t = p_t + e_t$ denote the $(C_2, \frac{C_{\text{prog}}}{6\sqrt{s}})$ short-flat decomposition of γ_t which exists by Definition 39 assuming the step oracle succeeded. We begin by observing

$$\begin{aligned} 2\eta R \langle \gamma_t, x_{t+1} - x_t \rangle + \|x_t - x_{t+1}\|_2^2 &= 2\eta R \langle e_t, x_{t+1} - x_t \rangle + 2\eta R \langle p_t, x_{t+1} - x_t \rangle + \|x_t - x_{t+1}\|_2^2 \\ &\geq -2\eta R \|e_t\|_\infty \|x_{t+1} - x_t\|_1 - \eta^2 R^2 \|p_t\|_2^2 \\ &\geq -\eta R^2 C_{\text{prog}} - \eta^2 R^2 C_2^2 . \end{aligned} \tag{9.23}$$

The first inequality followed from Hölder on the first term and Cauchy-Schwarz on the latter two terms in the preceding line. The second followed from the ℓ_1 radius of $\mathcal{X}$, and the bounds on e_t and

p_t from (9.18). Next, from Definition 39, for $\Delta = \Delta_t = \frac{1}{R}(\mathbf{A}x_t - b)$ and $v = \frac{1}{R}(x_t - x^\star)$,

$$2\eta R \langle \gamma_t, x_t - x^\star \rangle = 2\eta R \sum_{i\in[n]} w_i \Delta_i \langle a_i, v \rangle = 2\eta R^2 \sum_{i\in[n]} w_i \Delta_i^2 \geq 2\eta R^2 C_{\text{prog}}. \tag{9.24}$$

Finally, (9.21) follows from combining (9.22), (9.23), and (9.24), with our choice of η, and the fact that inducting on this lemma implies the ℓ_2 distance to $x^\star$ of the iterates is monotone decreasing.

Next, we claim that regardless of whether Algorithm 54 terminates on Line 7 or Line 11, we have $\|x_t - x^\star\|_2 \leq \frac{1}{4}R$. Note that the vector $v = \frac{1}{R}(x_t - x^\star)$ satisfies $\mathbf{A}v = \Delta := \frac{1}{R}(\mathbf{A}x_t - b)$. By assumption the condition $\|v\|_1 \leq 2\sqrt{2s}$ is met (since $x_t, x^\star \in \mathcal{X}$), and upon iterating (9.21) on our radius bound assumption, this implies that the condition $\|v\|_2 \leq 1$ is met. Hence, if the algorithm terminated on Line 7, we must have $\|v\|_2 \leq \frac{1}{4}R \implies \|x_t - x^\star\|_2 \leq \frac{1}{4}R$, as otherwise the termination condition would have been false. On the other hand, by (9.21), after T steps we have

$$\|x_T - x^\star\|_2^2 \leq \exp\left(-\frac{TC_2^2}{2C_{\text{prog}}^2}\right)\|x_0 - x^\star\|_2^2 \leq \frac{1}{16}R^2.$$

We conclude that at termination, $\|x_t - x^\star\|_2 \leq \frac{1}{4}R$. Now, s-sparsity of $x^\star$ and the definition of $x_{\text{out}} = \text{argmin}_{\|x\|_0 \leq s}\|x - x_t\|_2$ imply the desired

$$\|x_{\text{out}} - x^\star\|_2 \leq \|x_{\text{out}} - x_t\|_2 + \|x^\star - x_t\|_2 \leq 2\|x^\star - x_t\|_2 \leq \frac{1}{2}R. \tag{9.25}$$

□

9.6.2 Designing a step oracle

In this section, we design a step oracle $\mathcal{O}_{\text{step}}(\Delta, \mathbf{A})$ (see Definition 39) under Assumption 11 on the input matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$. Our step oracle iteratively builds a weight vector $\bar{w} \in \mathbb{R}^n_{\geq 0}$. It will be convenient to define

$$\gamma_{\bar{w}} := \sum_{i\in[n]} \bar{w}_i \Delta_i a_i. \tag{9.26}$$

Note that a valid step oracle always exists (although it is unclear how to implement the following solution): namely, setting $\bar{w} = w^\star$ satisfies the oracle assumptions by the second and third conditions in Assumption 11. In order to ensure Algorithm 58 is indeed a step oracle, we track two potentials for some μ, C we will define in Algorithm 55:

$$\begin{gathered}\Phi_2(\bar{w}) := \sum_{i\in[n]} \bar{w}_i \Delta_i^2 \text{ and } \Phi_{\text{sqmax}}(\bar{w}) := \left(\min_{\|p\|_2 \leq L\|\bar{w}\|_1} \text{sqmax}_\mu(\gamma_{\bar{w}} - p)\right) + \frac{\|\bar{w}\|_1}{4CLs}, \\ \text{where } \text{sqmax}_\mu(x) := \mu^2 \log\left(\sum_{j\in[d]} \exp\left(\frac{x_j^2}{\mu^2}\right)\right).\end{gathered} \tag{9.27}$$

Intuitively, $\Phi_2(\bar{w})$ corresponds to progress on (9.17), and $\Phi_{\text{sqmax}}(\bar{w})$ is intended to track the bounds (9.18). We note the following fact about the sqmax function which follows from direct calculation.

Fact 25. *For all $x \in \mathbb{R}^d$, $\|x\|_\infty^2 \le \text{sqmax}_\mu(x)$, and $\text{sqmax}_\mu(x) \ge \mu^2 \log(d)$.*

Also it will be important to note that $\Phi_{\text{sqmax}}(\bar{w})$ can be computed to high precision efficiently. We state this claim in the following and defer a full proof to Appendix H.3; we give a subroutine which performs a binary search on a Lagrange multiplier on the ℓ_2 constraint on p, and then solves for each optimal p_j using another binary search based on the Lagrange multiplier value.

Lemma 171. *Let $\delta > 0$ and $\theta \ge 0$. For any vector $\gamma \in \mathbb{R}^d$, we can solve the optimization problem*

$$\min_{\|p\|_2 \le \theta} \text{sqmax}_\mu(\gamma - p)$$

to additive accuracy δ in time

$$O\left(d \log^2\left(\frac{\|\gamma\|_2^2}{\mu\sqrt{\delta}}\right)\right).$$

We state the full implementation of our step oracle as Algorithm 55 below.

Algorithm 55: StepOracle$(\Delta, \mathbf{A}, \delta)$

Input: $\Delta \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{n \times d}$ satisfying Assumption 11, $\delta \in (0,1)$;
Output: w such that if there is $v \in \mathbb{R}^d$ with $\frac{1}{4} \le \|v\|_2 \le 1$ and $\|v\|_1 \le 2\sqrt{2s}$ such that $\Delta = \mathbf{A}v$, with probability $\ge 1 - \delta$, (9.17), (9.18) are satisfied with $Cp = 1$, $C_2 = O(1)$.
$C \leftarrow 200$, $\mu \leftarrow \frac{1}{\sqrt{Cs \log d}}$, $\eta \leftarrow \frac{1}{Kw_\infty^\star s\rho^2 \log d}$, $N' \leftarrow \lceil \log_2 \frac{1}{\delta} \rceil$;
for $0 \le k \le N'$ **do**
  $w_0 \leftarrow 0_n$, $N \leftarrow \lceil \frac{5Ln}{\eta} \rceil$;
  for $0 \le t \le N$ **do**
    if $\Phi_2(w_t) \ge 1$ **then Return:** $w \leftarrow w_t$;
    Sample $i \sim_{\text{unif.}} [n]$;
    Compute (using Lemma 171) $d_t \in [0, \eta w_\infty^\star]$ maximizing to additive $O(\frac{\eta}{n})$

$$\Gamma_t(d) := \Phi_2(w_t + de_i) - Cs\Phi_{\text{sqmax}}(w_t + de_i) \tag{9.28}$$

    $w_{t+1} \leftarrow w_t + d_t e_i$;
Return: $w \leftarrow 0_n$;

Our main helper lemma bounds the expected increase in Φ_{sqmax} from choosing a row of $\mathbf{A}$ uniformly at random, and choosing a step size according to $w^\star$. We do not know $w^\star$, but we argue that our algorithm makes at least this much expected progress. Define the decomposition promised by (9.20):

$$p^\star := \text{trunc}\left(\mathbf{A}^\top \mathbf{W}^\star \Delta, \frac{1}{K\sqrt{s}}\right), \; e^\star := \mathbf{A}^\top \mathbf{W}^\star \Delta - p^\star.$$

Furthermore, define for all $i \in [n]$,

$$z^{(i)} := \eta w_i^\star (\Delta_i a_i - p^\star), \tag{9.29}$$

where $p^\star$ is given by (9.20). We use $\{z^{(i)}\}_{i\in[n]}$ as certificates of Φ_{sqmax}'s growth in the following.

Lemma 172. *Assume that the constant K in Assumption 11 is sufficiently large, and that $\Delta = \mathbf{A}v$ where v satisfies the norm conditions in Assumption 11. Then for any $\bar{w} \in \mathbb{R}^n_{\geq 0}$ such that $\Phi_{\text{sqmax}}(\bar{w}) \leq C^2\mu^2 \log d$, and $\eta \leq \frac{1}{Kw^\star_\infty s\rho^2 \log d}$, we have*

$$\mathbb{E}_{i\sim_{\text{unif.}}[n]} \left[\Phi_{\text{sqmax}}(\bar{w} + \eta w_i^\star)\right] \leq \Phi_{\text{sqmax}}(\bar{w}) + \frac{1}{2CLs} \cdot \frac{\eta}{n}.$$

Proof. We assume for simplicity $L \geq 2\sqrt{2}$ as otherwise we may set $L \leftarrow \max(2\sqrt{2}, L)$ and (9.19) remains true. Let $p_{\bar{w}}$ be the minimizing argument in the definition of $\Phi_{\text{sqmax}}(\bar{w})$ in (9.27). For any $i \in [n]$, it is clear that $p_{\bar{w}} + (\eta w_i^\star)p^\star$ is a valid argument for the optimization problem defining $\Phi_{\text{sqmax}}(\bar{w} + \eta w_i^\star)$ by $\|p^\star\|_2 \leq L$, and since $\|w\|_1$ grows by $\eta w_i^\star$. Next, define

$$F(x) := \sum_{j\in[d]} \exp\left(\frac{x_j^2}{\mu^2}\right) \tag{9.30}$$

such that $\Phi_{\text{sqmax}}(\bar{w}) = \mu^2 \log F(x) + \frac{\|\bar{w}\|_1}{4CLs}$ for $x = \gamma_{\bar{w}} - p_{\bar{w}}$. As discussed earlier, since $\|p_{\bar{w}} + (\eta w_i^\star)p^\star\|_2 \leq \|p_{\bar{w}}\|_2 + \eta w_i^\star L$, we conclude

$$\Phi_{\text{sqmax}}(\bar{w} + \eta w_i^\star) \leq \mu^2 \log F(x + z^{(i)}) + \frac{\|\bar{w} + \eta w_i^\star\|_1}{4CLs}. \tag{9.31}$$

We next compute

$$\begin{aligned} \frac{1}{n}\sum_{i\in[n]} F(x + z^{(i)}) &= \frac{1}{n}\sum_{j\in[d]} \exp\left(\frac{x_j^2}{\mu^2}\right)\left(\sum_{i\in[n]} \exp\left(\frac{2x_j z_j^{(i)} + (z_j^{(i)})^2}{\mu^2}\right)\right) \\ &\leq \frac{1}{n}F(x)\max_{j\in[d]}\left(\sum_{i\in[n]} \exp\left(\frac{2x_j z_j^{(i)} + (z_j^{(i)})^2}{\mu^2}\right)\right). \end{aligned} \tag{9.32}$$

We now bound the right-hand side of this expression. For any $i \in [n]$ and $j \in [d]$, recalling (9.29),

$$\left|z_j^{(i)}\right| \leq \eta w_i^\star (|\Delta_i| \|a_i\|_\infty + \|p^\star\|_2) \leq \eta w_\infty^\star L(\sqrt{s}\rho^2 + 1). \tag{9.33}$$

The second inequality used our bounds from Assumption 11; note that for $\Delta = \mathbf{A}v$ where v satisfies the norm conditions in Assumption 11, $|\Delta_i| \leq \rho \|v\|_1 \leq 2\sqrt{2s}\rho$. Hence, if we choose a sufficiently

large constant K in Assumption 11, we have

$$\frac{1}{\mu}\left|z_j^{(i)}\right| \le \frac{\sqrt{C}}{K\sqrt{s\log d}\rho^2}\cdot\left(L(\sqrt{s}\rho^2+1)\right) \le \frac{1}{4C\sqrt{\log d}}.$$

Also by the assumption that $\Phi_{\mathrm{sqmax}}(\bar{w}) \le C^2\mu^2\log d$ we must have that for all $j\in[d]$,

$$\frac{|x_j|}{\mu} \le C\sqrt{\log d}.$$

Now, using $\exp(c) \le 1+c+c^2$ for $|c|\le 1$, we get

$$\begin{aligned}\sum_{i\in[n]}\exp\left(\frac{2x_jz_j^{(i)}+(z_j^{(i)})^2}{\mu^2}\right) &\le \sum_{i\in[n]}\left(1+\frac{2x_jz_j^{(i)}}{\mu^2}+\frac{(z_j^{(i)})^2}{\mu^2}+\left(\frac{2x_jz_j^{(i)}+(z_j^{(i)})^2}{\mu^2}\right)^2\right)\\ &\le \sum_{i\in[n]}\left(1+\frac{2x_jz_j^{(i)}}{\mu^2}+10C^2\log d\cdot\frac{(z_j^{(i)})^2}{\mu^2}\right).\end{aligned} \tag{9.34}$$

We control the first-order term via the observation that $\sum_{i\in[n]} z^{(i)} = \eta e^\star$ which is ℓ_∞-bounded from (9.20), so taking the constant K in Assumption 11 sufficiently large, we have

$$\begin{aligned}&\left|\sum_{i\in[n]}\frac{z_j^{(i)}}{\mu}\right| \le \frac{\eta}{\mu}\left\|e^\star\right\|_\infty \le \frac{\eta\sqrt{C\log d}}{K}\\ \implies &\left|\sum_{i\in[n]}\frac{2x_jz_j^{(i)}}{\mu^2}\right| \le 2C\sqrt{\log d}\cdot\frac{\eta\sqrt{C\log d}}{K} \le \frac{\eta\log d}{8L}.\end{aligned} \tag{9.35}$$

In the last inequality we assumed $K \ge 16C^{1.5}L$. We control the second-order term by using $(a+b)^2 \le 2a^2+2b^2$, $\left\|p^\star\right\|_\infty \le \left\|p^\star\right\|_2 \le L$, and (9.19):

$$\sum_{i\in[n]}\left(z_j^{(i)}\right)^2 \le 2\eta^2 w_\infty^\star\left(\sum_{i\in[n]} w_i^\star[p^\star]_j^2+\sum_{i\in[n]} w_i^\star\Delta_i^2\rho^2\right) \le 2\eta^2 w_\infty^\star(L\rho^2+L^2). \tag{9.36}$$

Putting together (9.34), (9.35), and (9.36), with the definition of μ, we conclude for sufficiently large K,

$$\begin{aligned}\frac{1}{n}\sum_{i\in[n]}\exp\left(\frac{2x_jz_j^{(i)}+(z_j^{(i)})^2}{\mu^2}\right) &\le 1+\frac{\eta\log d}{8Ln}+\frac{2\eta^2w_\infty^\star(L\rho^2+L^2)}{n\mu^2}\\ &\le 1+\frac{\eta\log d}{4Ln}.\end{aligned}$$

Hence, combining the above with (9.32), and using $\log(1+c) \le c$ for all c,

$$\mu^2 \log\left(\frac{1}{n}\sum_{i\in[n]} F(x+z^{(i)})\right) \le \mu^2 \log F(x) + \frac{\mu^2 \eta \log d}{4Ln} = \mu^2 \log F(x) + \frac{1}{Cs}\cdot\frac{\eta}{4Ln}. \tag{9.37}$$

Finally, we compute via (9.31) and concavity of log,

$$\begin{aligned}
\mathbb{E}_{i\sim_{\text{unif.}}[n]}[\Phi_{\text{sqmax}}(\bar{w}+\eta w_i^\star)] &\le \frac{\mu^2}{n}\sum_{i\in[n]} \log F(x+z^{(i)}) + \frac{\|\bar{w}\|_1}{4CLs} + \frac{1}{4CLs}\left(\frac{1}{n}\sum_{i\in[n]}\eta w_i^\star\right) \\
&\le \mu^2 \log\left(\frac{1}{n}\sum_{i\in[n]} F(x+z^{(i)})\right) + \frac{\|\bar{w}\|_1}{4CLs} + \frac{1}{4CLs}\cdot\frac{\eta}{n} \\
&\le \mu^2 \log F(x) + \frac{\|\bar{w}\|_1}{4CLs} + \frac{1}{Cs}\cdot\frac{\eta}{2Ln} = B(\bar{w}) + \frac{1}{Cs}\cdot\frac{\eta}{2Ln}.
\end{aligned}$$

In the last line, we used the bound (9.37). □

Finally, we can complete the analysis of Algorithm 55.

Lemma 168. *Suppose* $\mathbf{A}$ *satisfies Assumption 11. Algorithm 55 is a* $(C_{\text{prog}}, C_2, \delta)$ *step oracle* StepOracle *for* $(\Delta, \mathbf{A})$ *with* $C_{\text{prog}} = \Omega(1)$, $C_2 = O(1)$ *running in time*

$$O\left(\left(nd\log^3(nd\rho)\log\frac{1}{\delta}\right)\cdot\left(w_\infty^\star s\rho^2\log d\right)\right).$$

Proof. It suffices to prove Algorithm 55 meets its output guarantees in this time. Throughout this proof, we consider one run of Lines 5-10 of the algorithm, and prove that it successfully terminates on Line 7 with probability $\ge \frac{1}{2}$ assuming $\mathbf{A}$ satisfies Assumption 11 and that $\Delta = \mathbf{A}v$ for v satisfying the norm bounds in Assumption 11. This yields the failure probability upon repeating N' times.

For the first part of this proof, we assume we can exactly compute Δ_t, and carry out the proof accordingly. We discuss issues of approximation tolerance at the end, when bounding the runtime.

Correctness. We use the notation $A_t := \Phi_2(w_t)$, $B_t := \Phi_{\text{sqmax}}(w_t)$, and $\Phi_t := A_t - CsB_t$. We first observe that A_t is 1-Lipschitz, meaning it can only increase by 1 in any given iteration; this follows from $\eta w_\infty^\star \Delta_i^2 \le \frac{1}{8s\rho^2}\Delta_i^2 \le 1$, since $\Delta_i^2 = \langle a_i, v\rangle^2 \le 8s\rho^2$ by ℓ_∞-ℓ_1 Hölder.

Suppose some run of Lines 5-13 terminates by returning on Line 8 in iteration T, for $0 \le T \le N$. The termination condition implies that $A_T \ge 1 = C_{\text{prog}}$, so to show that the algorithm satisfies Definition 39, it suffices to show existence of a short-flat decomposition in the sense of (9.18). Clearly, Φ_t is monotone non-decreasing in t, since we may always force $\Gamma_t = 0$ by choosing $d_t = 0$. Moreover, $\Phi_0 = -CsB_0 = -Cs\mu^2 \log d = -1$. The above Lipschitz bound implies that $A_T \le 2$,

since $A_{T-1} \le 1$ by the termination condition; hence,

$$A_T - CsB_T = \Phi_T \ge \Phi_0 = -1 \implies B_T \le \frac{A_T + 1}{Cs} \le \frac{3}{Cs} \le C^2\mu^2 \log d.$$

Note that the above inequality and nonnegativity of sqmax_μ imply that $\frac{\|w_T\|_1}{4LCs} \le \frac{3}{Cs}$, so $\|w_T\|_1 \le 12L$. For the given value of $C = 200$, and the first inequality in Fact 25, the definition of the first summand in B implies there is a short-flat decomposition meeting (9.18) with $C_2 = L\|w_T\|_1 = O(1)$.

Hence, we have shown that Definition 39 is satisfied whenever the algorithm returns on Line 7. We make one additional observation: whenever $\Phi_t \ge 0$, the algorithm will terminate. This follows since on such an iteration,

$$A_t \ge CsB_t \ge CsB_0 = Cs\mu^2 \log d = 1,$$

since clearly the function B is minimized by the all-zeroes weight vector, attaining value $\mu^2 \log d$.

Success probability. We next show that with probability at least $\frac{1}{2}$, the loop in Lines 5-10 will terminate. Fix an iteration t. When sampling $i \in [n]$, the maximum gain in Φ_t for $d_t \in [0, \eta w_\infty^\star]$ is at least that attained by setting $d_t = \eta w_i^\star$, and hence

$$\mathbb{E}[\Phi_{t+1} - \Phi_t \mid A_t \le 1] \ge \frac{\eta}{Ln} - \frac{\eta}{2Ln} = \frac{\eta}{2Ln}. \tag{9.38}$$

Here, we used that the expected gain in A_t by choosing $d_t = \eta w_i^\star$ over a uniformly sampled $i \in [n]$ is lower bounded by $\frac{\eta}{Ln}$ via (9.19), and the expected gain in CsB_t is upper bounded by Lemma 172.

Let Z_t be the random variable equal to $\Phi_t - \Phi_0$, where we freeze the value of $w_{t'}$ for all $t' \ge t$ if the algorithm ever returns on Line 8 in an iteration t. Notice that $Z_t \le 2$ always: whenever $Z_t \ge 1$, we have $\Phi_t \ge 0$ so the algorithm will terminate, and Z_t is 1-Lipschitz because A_t is. Moreover, whenever we are in an iteration t where $\Pr[A_t \ge 1] \le \frac{1}{2}$, applying (9.38) implies

$$\mathbb{E}[Z_{t+1} - Z_t] = \mathbb{E}[Z_{t+1} - Z_t \mid A_t \le 1] \Pr[A_t \le 1] \ge \frac{\eta}{4Ln}.$$

Clearly, $\Pr[A_t \ge 1]$ is a monotone non-decreasing function of t, since A_t is monotone. After $N \ge \frac{5Ln}{\eta}$ iterations, if we still have $\Pr[A_t \ge 1] \le \frac{1}{2}$, we would obtain a contradiction since recursing the above display yields $\mathbb{E}[Z_N] > 2$. This yields the desired success probability.

Runtime. The cost of each iteration is dominated by the following computation in Line 9: we wish to find $d \in [0, \eta w_\infty^\star]$ maximizing to additive $O(\frac{\eta}{n})$ the following objective:

$$\Phi_2(w + de_i) - Cs\Phi_{\text{sqmax}}(w + de_i).$$

We claim the above function is a concave function of d. First, we show Φ_{sqmax} is convex (and the result will then follow from linearity of Φ_2). To see this, for two values w_i and w_i', let the

corresponding maximizing arguments in the definition of $\Phi_{\text{sqmax}}(\bar{w} + w_i)$ and $\Phi_{\text{sqmax}}(\bar{w} + w_i')$ be denoted p and p'. Then, $\frac{1}{2}(p+p')$ is a valid argument for $\bar{w}+\frac{1}{2}(w_i+w_i')$, and by convexity of sqmax_μ and linearity of the ℓ_1 portion, we have the conclusion.

Next, note that all $|\Delta_i|$ are bounded by $2\sqrt{2s}\rho$ (proven after (9.33)) and all a_{ij} are bounded by ρ by assumption. It follows that the restriction of Φ_2 to a coordinate is $8s\rho^2$-Lipschitz. Moreover the linear portion of Φ_{sqmax} is clearly $\frac{1}{4CLs}$-Lipschitz in any coordinate. Finally we bound the Lipschitz constant of the sqmax part of Φ_{sqmax}. It suffices to bound Lipschitzness for any fixed p of

$$\text{sqmax}_\mu \left(\gamma_{\bar{w}} - p + d_i\Delta_i a_i\right)$$

because performing the minimization over p involved in two $\text{sqmax}(\gamma_{\bar{w}} - p + d\Delta_i a_i)$ and $\text{sqmax}(\gamma_{\bar{w}} - p + d'\Delta_i a_i)$ can only bring the function values closer together. By direct computation the derivative of the above quantity with respect to d_i is

$$\sum_{j\in[d]} \Delta_i a_{ij} \left(2\left[\gamma_{\bar{w}} - p + d_i\Delta_i a_i\right]_j\right) q_j$$

for some probability density vector $q \in \Delta^d$. Further we have

$$\left|\gamma_{\bar{w}} - p + d_i\Delta_i a_i\right|_j \le O\left(\sqrt{s}\rho^2\right) + O(1) + 2\sqrt{2s}\rho^2 \cdot (\eta w_\infty^\star).$$

Here we used our earlier proof that we must only consider values of $\|p\|_2 = O(1)$ throughout the algorithm (since $\|w_t\|_1 = O(1)$ throughout) and this also implies no coordinate of $\gamma_{\bar{w}}$ can be larger than $(\max_{i\in[n]}|\Delta_i|)(\max_{i\in[n],j\in[d]}|a_{ij}|)\|\bar{w}\|_1$ by definition of $\gamma_{\bar{w}}$. Combined with our bounds on linear portions this shows Φ_2 and Φ_{sqmax} are $\text{poly}(nd\rho)$-Lipschitz.

Hence, we may evaluate to the desired $O(\frac{\eta}{n})$ accuracy by approximate minimization of a Lipschitz convex function over an interval (Lemma 33, [143]) with a total cost of $O(d\log^3(nd\rho))$. Here we use the subroutine of Lemma 171 in Lemma 33 of [143], with evaluation time $O(d\log^2(nd\rho))$.

The algorithm then runs in NN' iterations, each bottlenecked by the cost of approximating Γ_t; combining these multiplicative factors yields the runtime. We note that we do not precompute $\Delta = \mathbf{A}v$; we can compute coordinates of Δ in time $O(d)$ as they are required by Algorithm 55. □

9.6.3 Equivalence between Assumption 11 and RIP

The main result of this section is an equivalence between Assumption 11 and the weighted restricted isometry property, which requires two helper tools to prove. The first is a "shelling decomposition."

Lemma 173. *Let $v \in \mathbb{R}^d$ have $\text{NS}(v) \le \sigma$. Then if we write $v = \sum_{l\in[k]} v^{(l)}$ where $v^{(1)}$ is obtained by taking the s largest coordinates of v, $v^{(2)}$ is obtained by taking the next s largest coordinates and*

so on (breaking ties arbitrarily so that the supports are disjoint), we have

$$\sum_{2\le l\le k}\left\|v^{(l)}\right\|_2 \le \sqrt{\frac{\sigma}{s}}\left\|v\right\|_2.$$

Proof. Note that the decomposition greedily sets $v^{(l)}$ to be the s largest coordinates (by absolute value) of $v-\sum_{l'\in[l-1]}v^{(l')}$, zeroing all other coordinates and breaking ties arbitrarily. This satisfies

$$\left\|v^{(l+1)}\right\|_2 \le \sqrt{s}\left\|v^{(l+1)}\right\|_\infty \le \frac{1}{\sqrt{s}}\left\|v^{(l)}\right\|_1.$$

The last inequality follows since every entry of $v^{(l)}$ is larger than the largest of $v^{(l+1)}$ in absolute value. Finally, summing the above equation and using disjointness of supports yields

$$\sum_{2\le l\le k}\left\|v^{(l)}\right\|_2 \le \frac{1}{\sqrt{s}}\left\|v\right\|_1 \le \sqrt{\frac{\sigma}{s}}\left\|v\right\|_2.$$

□

The second bounds the largest entries of image vectors from the transpose of an RIP matrix.

Lemma 174. *Let $\mathbf{A}\in\mathbb{R}^{n\times d}$ be (s,c)-RIP, and let $u\in\mathbb{R}^n$. Then,*

$$\left\|[\mathbf{A}^\top u]_{(s)}\right\|_2 \le c\left\|u\right\|_2.$$

Proof. Let $v=[\mathbf{A}^\top u]_{(s)}$. The lemma is equivalent to showing $\|v\|_2\le c\|u\|_2$. Note that

$$\|v\|_2^2 = \left\langle v,\mathbf{A}^\top u\right\rangle \le \|\mathbf{A}v\|_2\|u\|_2 \le c\|v\|_2\|u\|_2.$$

The first inequality used Cauchy-Schwarz, and the second applied the RIP property of $\mathbf{A}$ to v, which is s-sparse by construction. The conclusion follows via dividing by $\|v\|_2$. □

Using these helper tools, we now prove the main result of this section.

Lemma 175. *The following statements are true.*

1. *If $\mathbf{A}$ satisfies Assumption 11 with weight vector $w^\star$, then $(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}$ is (s,L)-RIP.*
2. *If the matrix $(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}$ is RIP with parameters*

$$\left(12800L^3K^2s,\frac{\sqrt{L}}{2}\right)$$

for $L\ge 1$, and $\|\mathbf{A}\|_{\max}\le\rho$, then $\mathbf{A}$ satisfies Assumption 11.

Proof. We prove each equivalence in turn.

Assumption 11 implies RIP. The statement of RIP is scale-invariant, so we will prove it for all s-sparse unit vectors v without loss of generality. Note that such v satisfies the condition in Assumption 11, since $\|v\|_2 = 1$ and $\|v\|_1 \le \sqrt{s}$ by Cauchy-Schwarz. Then, the second condition of Assumption 11 implies that for $\Delta = \mathbf{A}v$, we have the desired norm preservation:

$$\frac{1}{L} \le \left\|(\mathbf{W}^*)^{\frac{1}{2}}\mathbf{A}v\right\|_2^2 = \sum_{i\in[n]} w_i^\star \Delta_i^2 \le L.$$

Boundedness and RIP imply Assumption 11. Let $v \in \mathbb{R}^d$ satisfy $\frac{1}{4} \le \|v\|_2 \le 1$ and $\|v\|_1 \le 2\sqrt{2s}$, and define $\Delta := \mathbf{A}v$. The first condition in Assumption 11 is immediate from our assumed entrywise boundedness on $\mathbf{A}$, so we begin by demonstrating the lower bound in (9.19). Let

$$s' = 12800L^3K^2s$$

and let $v^{(1)}, \dots, v^{(k)}$ be the shelling decomposition into s'-sparse vectors given by Lemma 173, where $\sigma = 128s$ from the ℓ_1 and ℓ_2 norm bounds on v. By Lemma 173, we have

$$\left\|v^{(2)}\right\|_2 + \cdots + \left\|v^{(k)}\right\|_2 \le \frac{0.1}{L}\|v\|_2.$$

In particular, the triangle inequality then implies $0.9\|v\|_2 \le \left\|v^{(1)}\right\|_2 \le \|v\|_2$. Next, recall that $\sum_{i\in[n]} w_i^\star \Delta_i^2 = \left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v\right\|_2^2$. By applying the triangle inequality and since $(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}$ is RIP,

$$\begin{aligned}\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v\right\|_2^2 &\ge \left(\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v^{(1)}\right\|_2 - \sum_{l=2}^{k}\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v^{(l)}\right\|_2\right)^2 \\ &\ge \left(\frac{5}{\sqrt{L}}\cdot 0.9\|v\|_2 - \frac{\sqrt{L}}{2}\cdot\frac{1}{L}\|v\|_2\right)^2 \ge \frac{16}{L}\|v\|_2^2 \ge \frac{1}{L}.\end{aligned}$$

In the second inequality, we applied the RIP assumption to each individual term, since all the vectors are s'-sparse. Similarly, to show the upper bound in (9.19), we have

$$\begin{aligned}\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v\right\|_2^2 &\le \left(\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v^{(1)}\right\|_2 + \sum_{l=2}^{k}\left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v^{(l)}\right\|_2\right)^2 \\ &\le \left(\frac{\sqrt{L}}{2}\cdot\|v\|_2 + \frac{\sqrt{L}}{2}\cdot\frac{1}{L}\|v\|_2\right)^2 \le L.\end{aligned}$$

It remains to verify the final condition of Assumption 11. First, for $u := \mathbf{W}^{\frac{1}{2}}\mathbf{A}v$, by applying

the shelling decomposition to v into s'-sparse vectors $\{v^{(l)}\}_{l\in[k]}$,

$$\|u\|_2 \le \sum_{l\in[k]} \left\|(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}v^{(l)}\right\|_2 \le \sqrt{L}\,\|v\|_2\,. \tag{9.39}$$

Here, we used our earlier proof to bound the contribution of all terms but $v^{(1)}$. Applying Lemma 174 to the matrix $(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}$ and vector u, we have for $\Delta = \mathbf{A}v$,

$$\left\|\left[\mathbf{A}^\top(\mathbf{W}^\star)^{\frac{1}{2}}u\right]_{(s')}\right\|_2 = \left\|\left[\mathbf{A}^\top\mathbf{W}^\star\Delta\right]_{(s')}\right\|_2 \le L.$$

By setting the ℓ_2 bounded component in the short-flat decomposition of $\mathbf{A}^\top\mathbf{W}^\star\Delta$ to be the top s' entries by magnitude, it remains to show the remaining coordinates are ℓ_∞ bounded by $\frac{1}{K\sqrt{s}}$. This follows from the definition of s' and (9.39), which imply that the $s'+1^{\text{th}}$ largest coordinate (in magnitude) cannot have squared value larger than $\frac{L^2}{s'} \le \frac{1}{K^2 s}$ without contradicting (9.39). □

Finally, it is immediate that Lemma 169 follows from Lemma 175.

9.6.4 Putting it all together

At this point, we have assembled the tools to prove our main result on exact recovery.

Theorem 66. *Let $\delta \in (0,1)$, $r > 0$, and suppose $R_0 \ge \|x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$. Then with probability at least $1-\delta$, Algorithm 54 using Algorithm 55 as a step oracle takes as input a $(\rho, w^\star_\infty)$-wRIP matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $b = \mathbf{A}x^\star$, and computes $\hat{x}$ satisfying $\|\hat{x} - x^\star\|_2 \le r$ in time*

$$O\left(\left(nd\log^3(nd\rho)\log\left(\frac{1}{\delta}\cdot\log\frac{R_0}{r}\right)\log\left(\frac{R_0}{r}\right)\right)\cdot\left(w^\star_\infty s\rho^2\log d\right)\right).$$

Proof. With probability at least $1-\delta$, combining Lemma 168 and Lemma 169 implies that Assumption 11 holds for all $v \in \mathbb{R}^d$ where $\frac{1}{4} \le \|v\|_2 \le 1$ and $\|v\|_1 \le 2\sqrt{2s}$, and that for $N = O(\log\frac{R_0}{r})$, we can implement a step oracle for N runs of Algorithm 54 in the allotted time, each with failure probability $1 - \frac{\delta}{N}$. Moreover, Algorithm 54 returns in $O(1)$ iterations, and allows us to halve our radius upper bound. By taking a union bound on failure probabilities and repeatedly running Algorithm 54 N times, we obtain a radius upper bound of r with probability $\ge 1-\delta$. □

9.7 Noisy recovery

In this section, we give an algorithm for solving a noisy sparse recovery problem in a wRIP matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ (where we recall Definition 38). In particular, we assume that we receive

$$b = \mathbf{A}x^\star + \xi, \tag{9.40}$$

for an arbitrary unknown $\xi \in \mathbb{R}^n$, and $x^\star \in \mathbb{R}^d$ is s-sparse. Throughout this section, we will define

$$m := \frac{1}{w_\infty^\star}, \tag{9.41}$$

where $w_\infty^\star$ is an entrywise bound on w in Definition 38. We define the (unknown) "noise floor"

$$R_\xi := \frac{1}{\sqrt{m}} \left\| \xi_{(m)} \right\|_2,$$

where we defined $\cdot_{(m)}$ in Section 9.5. Our goal will be to return x such that $\|x - x^\star\|_2 = O(R_\xi)$. We now formally state the main result of this section here.

Theorem 67. *Let $\delta \in (0,1)$, and suppose $R_0 \geq \|x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$. Further, suppose $\mathbf{A} \in \mathbb{R}^{n \times d}$ is $(\rho, w_\infty^\star)$-wRIP and $b = \mathbf{A}x^\star + \xi$, and $R_1 \geq R_\xi := \left\| \xi_{(m)} \right\|_2$. Then with probability at least $1 - \delta$, Algorithm 56 using Algorithm 56 as a noisy step oracle computes $\hat{x}$ satisfying*

$$\|\hat{x} - x^\star\|_2 \leq R_{\text{final}} = \Theta(R_\xi),$$

in time

$$O\left(\left(ndw_\infty^\star s \log^4(nd\rho) \log^2\left(\frac{d}{\delta} \cdot \log\left(\frac{R_0}{R_{\text{final}}}\right) \log\left(\frac{R_1}{R_{\text{final}}}\right)\right)\right) \cdot \rho^2 \log\left(\frac{R_0}{R_{\text{final}}}\right) \log\left(\frac{R_1}{R_{\text{final}}}\right)\right).$$

Similarly to Theorem 66, Theorem 67 provides a runtime guarantee which interpolates between the fully random and semi-random settings, and runs in sublinear time when e.g. the entire measurement matrix $\mathbf{A}$ satisfies RIP. Theorem 67 further provides a refined error guarantee as a function of the noise vector ξ, which again interpolates based on the "quality" of the weights w. This is captured through the parameter $m = \frac{1}{w_\infty^\star}$: when $m \approx n$, the squared error bound R_ξ^2 scales as the average squared entry of ξ, and more generally it scales as the average of the largest m entries.

We solve the noisy variant by essentially following the same steps as Section 9.6 and making minor modifications to the analysis; we give an outline of the section here. In Section 9.7.1, we generalize the framework of Section 9.6.1 to the setting where we only receive noisy observations (9.40), while our current radius is substantially above the noise floor. We then implement an appropriate step oracle for this outer loop in Section 9.7.2, and prove that the relevant Assumption 12 used in our step oracle implementation holds when $\mathbf{A}$ is wRIP in Section 9.7.3.

9.7.1 Radius contraction above the noise floor using step oracles

In this section, we give the main loop of our overall noise-tolerant algorithm, HalfRadiusSparseNoisy, which takes as input s-sparse x_{in} and a radius bound $R \geq \|x_{\text{in}} - x^\star\|_2$. It then returns an s-sparse vector x_{out} with the guarantee $\|x_{\text{out}} - x^\star\|_2 \leq \frac{1}{2}R$, *as long as* R is larger than an appropriate multiple of R_ξ. We give the analog of Definition 39 in this setting, termed a "noisy step oracle."

Definition 40 (Noisy step oracle)**.** *We say that $\mathcal{O}_{\text{nstep}}$ is a $(C_{\text{prog}}, C_2, C_\xi, \delta)$-noisy step oracle for $\widetilde{\Delta} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$ if the following holds. Whenever there is $v \in \mathbb{R}^d$ with $\frac{1}{12} \le \|v\|_2 \le 1$ such that $\widetilde{\Delta} = \mathbf{A}v + \xi$ where $\left\|\xi_{(m)}\right\|_2 \le \frac{\sqrt{m}}{C_\xi}$, with probability $\ge 1 - \delta$, $\mathcal{O}_{\text{nstep}}$ returns $w \in \mathbb{R}^n_{\ge 0}$ such that the following two conditions hold. First,*

$$\sum_{i \in [n]} w_i \widetilde{\Delta}_i \Delta_i \ge C_{\text{prog}}. \tag{9.42}$$

Second, there exists a $(C_2, \frac{C_{\text{prog}}}{6\sqrt{s}})$ short-flat decomposition of $\mathbf{A}^\top \mathbf{diag}(w)\, \Delta$:

$$\left\| \text{trunc}\left(\mathbf{A}^\top \mathbf{diag}(w)\, \Delta, \frac{C_{\text{prog}}}{6\sqrt{s}} \right) \right\|_2 \le C_2.$$

We next characterize how a strengthened step oracle with appropriate parameters also is a noisy step oracle. First, we will need a definition.

Definition 41. *For distributions A, B on $\mathbb{R}^n$, we say A stochastically dominates B if there is a random variable C on $\mathbb{R}^n$ whose coordinates are always nonnegative such that the distribution of A is the same as the distribution of $B + C$ (where C may depend on the realization of B).*

We now formalize the properties of the strengthened step oracle that we will construct.

Definition 42 (Strong step oracle)**.** *We say that $\mathcal{O}_{\text{step}}$ is a $(C_{\text{prog}}, C_2, C_\xi, \delta)$-strong step oracle for $\widetilde{\Delta} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$ if it satisfies all the properties of a standard step oracle (Definition 39), as well as the following additional guarantees.*

1. *For the output weights w, we have*

$$\|w\|_1 \le \frac{C_{\text{prog}} C_\xi^2}{4} \cdot \delta. \tag{9.43}$$

2. *The distribution of w output by the oracle is stochastically dominated by the distribution*

$$\frac{\delta}{4s\rho^2 \log \frac{d}{\delta}} \text{Multinom}\left(\left\lceil \frac{C_{\text{prog}} C_\xi^2 n s \rho^2 \log \frac{d}{\delta}}{m} \right\rceil, \left(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n} \right) \right)$$

   for some $\rho \ge 1$.

3. *Compared to Definition 39 (the step oracle definition), we have the stronger guarantees that $\mathbf{A}^\top \mathbf{diag}(w)\, \Delta$ admits a $(C_2, \frac{C_{\text{prog}}}{24\sqrt{s}})$ short-flat decomposition in* (9.18)*, and obtains its guarantees using the bounds $\frac{1}{12} \le \|v\|_2 \le 1$ (instead of a lower bound of $\frac{1}{4}$).*

We next demonstrate that a strong step oracle is a noisy step oracle.

Lemma 176. *Suppose $\mathcal{O}_{\text{step}}$ is a $(C_{\text{prog}}, C_2, C_\xi, \delta)$-strong step oracle for $\widetilde{\Delta} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$. Then, $\mathcal{O}_{\text{step}}$ is also a $(\frac{1}{4}C_{\text{prog}}, C_2, C_\xi, 2\delta)$-noisy step oracle for $(\widetilde{\Delta}, \mathbf{A})$.*

Proof. In the definition of a noisy step oracle, we only need to check the condition that $\sum_{i\in[n]} w_i \widetilde{\Delta}_i \Delta_i \geq \frac{1}{4}C_{\text{prog}}$ for an arbitrary $\Delta = \widetilde{\Delta} - \xi$ where $\left\|\xi_{(m)}\right\|_2 \leq \sqrt{m}C_\xi^{-1}$, as all other conditions are immediate from Definition 42. Note that

$$\begin{aligned}\sum_{i\in[n]} w_i \widetilde{\Delta}_i \Delta_i &= \sum_{i\in[n]} w_i \widetilde{\Delta}_i (\widetilde{\Delta}_i - \xi_i) \\ &\geq \frac{1}{2}\sum_{i\in[n]} w_i \widetilde{\Delta}_i^2 - \frac{1}{2}\sum_{i\in[n]} w_i \xi_i^2 \,.\end{aligned}$$

where we used $a^2 - ab \geq \frac{1}{2}a^2 - \frac{1}{2}b^2$. The first sum above is at least $\frac{1}{2}C_{\text{prog}}$ by assumption. To upper bound the second sum, we will use the second property in the definition of a strong step oracle. Let $S \subset [n]$ be the set consisting of the m largest coordinates of ξ (with ties broken lexicographically). Let α be drawn from the distribution

$$\frac{\delta}{4s\rho^2 \log\frac{d}{\delta}} \text{Multinom}\left(\left\lceil \frac{C_{\text{prog}}C_\xi^2 ns\rho^2 \log\frac{d}{\delta}}{m}\right\rceil, \left(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n}\right)\right).$$

Note that with $1 - 0.1\delta$ probability, by a Chernoff bound, we have that $\sum_{i\in S}\alpha_i \geq \frac{1}{5}\delta C_{\text{prog}}C_\xi^2$. If this happens, then since S consists of the largest coordinates of ξ, any vector β such that $\beta \leq \alpha$ entrywise and $\|\beta\|_1 \leq \frac{1}{4}\delta C_{\text{prog}}C_\xi^2$ must have

$$\sum_{i\in[n]} \beta_i \xi_i^2 \leq \frac{5}{4}\sum_{i\in S} \alpha_i \xi_i^2.$$

Now note that for any S with $|S| = m$,

$$\mathbb{E}\left[\sum_{i\in S} \alpha_i \xi_i^2\right] \leq \frac{\delta C_{\text{prog}}C_\xi^2}{4m} \cdot \left\|\xi_{(m)}\right\|_2^2 \leq \frac{\delta C_{\text{prog}}}{4}.$$

Combining the above two inequalities and Markov's inequality and the fact that the distribution of α stochastically dominates the distribution of w, we deduce that with at least $1 - \delta$ probability,

$$\sum_{i\in[n]} w_i \xi_i^2 \leq \frac{1}{0.9\delta} \cdot \mathbb{E}\left[\max_{\substack{\beta\leq\alpha \\ \|\beta\|_1 \leq \frac{1}{4}\delta C_{\text{prog}}C_\xi^2}} \sum_{i\in[n]} \beta_i \xi_i^2\right] \leq \frac{C_{\text{prog}}}{2}.$$

Putting everything together, we conclude that we have

$$\sum_{i\in[n]} w_i \widetilde{\Delta}_i \Delta_i \geq \frac{C_{\text{prog}}}{4}$$

with failure probability at most 2δ, completing the proof. □

In Section 9.7.2, we prove that if $\mathbf{A}$ satisfies Assumption 12 (a slightly different assumption than Assumption 11) then with high probability we can implement a strong step oracle with appropriate parameters. This is stated more formally in the following; recall m is defined in (9.41).

Assumption 12. *The matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ satisfies the following. There is a weight vector $w^\star \in \Delta^n$ with $\|w^\star\|_\infty \leq w^\star_\infty = \frac{1}{m}$, a constants L, $\rho \geq 1$, and constants K, C_ξ (which may depend on L) such that for all $v \in \mathbb{R}^d$, $\xi \in \mathbb{R}^n$ with*

$$\frac{1}{4} \leq \|v\|_2 \leq 1,\ \|v\|_1 \leq 2\sqrt{2s},\ \left\|\xi_{(m)}\right\|_2 \leq \frac{\sqrt{m}}{C_\xi}$$

we have, defining $\widetilde{\Delta} = \mathbf{A}v + \xi$:

1. *$\mathbf{A}$ is entrywise bounded by $\pm\rho$, i.e. $\|\mathbf{A}\|_{\max} \leq \rho$.*
2. $$\frac{1}{L} \leq \sum_{i\in[n]} w_i^\star \widetilde{\Delta}_i^2 \leq L. \tag{9.44}$$
3. *There is a $(L, \frac{1}{K\sqrt{s}})$ short-flat decomposition of $\mathbf{A}^\top \mathbf{W}^\star \widetilde{\Delta}$:*
$$\left\|\text{trunc}\left(\mathbf{A}^\top \mathbf{W}^\star \widetilde{\Delta}, \frac{1}{K\sqrt{s}}\right)\right\|_2 \leq L. \tag{9.45}$$

Lemma 177. *Suppose $\mathbf{A}$ satisfies Assumption 12. Algorithm 58 is a $(C_{\text{prog}}, C_2, C_\xi, \delta)$ strong step oracle* StrongStepOracle *for $(\widetilde{\Delta}, \mathbf{A})$ with*

$$C_{\text{prog}} = \Omega(1),\ C_2 = O\left(1\right),\ C_\xi = O(1),\ \delta = \frac{1}{2}\left(\frac{C_2}{10^5 C_{\text{prog}}}\right)^2,$$

running in time

$$O\left(\left(nd\log^3(nd\rho)\log\frac{1}{\delta}\right)\cdot\left(w^\star_\infty s\rho^2\log^2\frac{d}{\delta}\right)\right).$$

Here, in contrast to the noiseless setting, we can only guarantee that the strong step oracle (and thus also the noisy step oracle) succeeds with constant probability. In our full algorithm, we boost the success probability of the oracle by running a logarithmic number of independent trials and

aggregating the outputs. We also show that for an appropriate choice of constants in Definition 38, Assumption 12 is also satisfied, stated in Lemma 178 and proven in Section 9.7.3.

Lemma 178. *Suppose* $\mathbf{A} \in \mathbb{R}^{n\times d}$ *is* $(\rho, w^\star_\infty)$*-wRIP with a suitable choice of constants in the RIP parameters in Definition 38. Then,* $\mathbf{A}$ *also satisfies Assumption 12.*

Algorithm 56: HalfRadiusSparseNoisy$(x_{\text{in}}, R, R_\xi, \mathcal{O}_{\text{nstep}}, \delta, \mathbf{A}, b)$

Input: s-sparse $x_{\text{in}} \in \mathbb{R}^d$, $R \geq \|x_{\text{in}} - x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$, $(C_{\text{prog}}, C_2, C_\xi, \delta')$-noisy step oracle $\mathcal{O}_{\text{nstep}}$ for all $(\Delta, \mathbf{A})$ with $\Delta \in \mathbb{R}^n$, $\delta' \leq (10^{-4}\frac{C_{\text{prog}}}{C_2})^2$, $\mathbf{A} \in \mathbb{R}^{n\times d}$, $b = \mathbf{A}x^\star + \xi \in \mathbb{R}^n$ for $\left\|\xi_{(m)}\right\|_2 \leq R_\xi\sqrt{m}$, with $R \geq C_\xi R_\xi$;
Output: s-sparse vector x_{out} that satisfies $\|x_{\text{out}} - x^\star\|_2 \leq \frac{1}{2}R$ with probability $\geq 1 - \delta$
$x_0 \leftarrow x_{\text{in}}$, $\mathcal{X} \leftarrow \{x \in \mathbb{R}^d \mid \|x - x_{\text{in}}\|_1 \leq \sqrt{2s}R\}$;
$T \leftarrow \left\lceil \frac{100C_2^2}{C_{\text{prog}}^2} \right\rceil$, $\eta \leftarrow \frac{C_{\text{prog}}}{2C_2^2}$;
$N_{\text{trials}} \leftarrow 10 \log \frac{d}{\delta}$;
for $1 \leq j \leq N_{\text{trials}}$ **do**
  $x_0^j \leftarrow x_0$;
  for $0 \leq t \leq T-1$ **do**
    $w_t^j \leftarrow \mathcal{O}_{\text{nstep}}(\Delta_t^j, \mathbf{A})$ for $\Delta_t^j \leftarrow \frac{1}{R}(\mathbf{A}x_t^j - b)$, $\gamma_t^j \leftarrow \mathbf{A}^\top \mathbf{diag}\left([w_t^j]\right)\Delta_t^j = \sum_{i\in[n]}[w_t^j]_i[\Delta_t^j]_i a_i$;
    if (w_t^j, γ_t^j) *do not meet all of* (9.17), (9.18) *and the additional criteria in Definition 42* **then**
      $x_T^j \leftarrow [x_t^j]_{(s)}$;
      Break: ;
    $x_{t+1}^j \leftarrow \text{argmin}_{x\in\mathcal{X}} \left\|x - x_t^j - \eta R\gamma_t\right\|_2$;
$x_T \leftarrow$ Aggregate$(\{x_T^1, \ldots, x_T^{N_{\text{trials}}}\}, \frac{R}{2})$;
Return: $x_{\text{out}} \leftarrow [x_T]_{(s)}$;

Algorithm 57: Aggregate$(\mathcal{S}, R)$

Input: $\mathcal{S} = \{y_i\}_{i\in[k]} \subset \mathbb{R}^d$, $R \geq 0$ such that for some unknown $z \in \mathbb{R}^d$, at least $0.51k$ points $y_i \in \mathcal{S}$ have $\|y_i - z\|_2 \leq \frac{R}{3}$;
Output: $\widetilde{z}$ with $\|\widetilde{z} - z\|_2 \leq R$;
for $1 \leq i \leq k$ **do**
  if *at least* $0.51k$ *points* $y_j \in \mathcal{S}$ *satisfy* $\|y_i - y_j\|_2 \leq \frac{2R}{3}$ **then Return:** $\widetilde{z} \leftarrow y_i$;

Next, we give a guarantee regarding our geometric post-processing step, Algorithm 57.

Claim 1. Aggregate$(\mathcal{S}, R)$ *runs in* $O(k^2 d)$ *time and meets its output guarantees.*

Proof. Let T be the subset of indices $i \in [k]$ such that $\|y_i - z\| \leq \frac{R}{3}$. Whenever the algorithm tests y_i for some $i \in T$, it will be returned and satisfies the desired properties. Now consider any y_i

returned by the algorithm. The ball of radius $\frac{2R}{3}$ around y_i intersects the ball of radius $\frac{R}{3}$ around z, since otherwise it can only contain at most $0.49k$ points. Thus, $\|y_i - z\|_2 \le R$. The runtime is dominated by the time it takes to do k^2 distance comparisons of points in $\mathbb{R}^d$. □

We remark that is possible that for $k = \Omega(\log \frac{1}{\delta})$ as is the case in our applications, the runtime of Claim 1 can be improved to have a better dependence on k by subsampling the points and using low-rank projections for distance comparisons.

Lemma 179. *Assume* $\mathbf{A}$ *satisfies Assumption 12. Then, Algorithm 56 meets its output guarantees in time*

$$O\left(\left(nd\log^3(nd\rho)\right)\cdot\left(w_\infty^\star s\rho^2\log d\right)\cdot\log^2\frac{d}{\delta}\right).$$

Proof. We claim that for each independent trial $j \in [N_{\text{trials}}]$, except with probability $1 - T\delta'$, the output x_T^j satisfies $\|x_T^j - x^\star\|_2 \le \frac{R}{6}$. Once we prove this, by Chernoff at least $0.51N_{\text{trials}}$ of the trials satisfy $\|x_T^j - x^\star\|_2 \le \frac{R}{6}$ except with probability at most δ, and then we are done by Claim 1.

It remains to prove the above claim. Fix a trial j, and drop the superscript j for notational convenience. In every iteration t, $\widetilde{\Delta} := \frac{1}{R}(\mathbf{A}x_t - b)$ is given to $\mathcal{O}_{\text{nstep}}$. Since $b = \mathbf{A}x^\star + \xi$, we have

$$\widetilde{\Delta} = \frac{1}{R}(\mathbf{A}(x - x^\star) + \xi) = \mathbf{A}v + \xi,$$

for $\|v\|_2 \le 1$, $\|v\|_1 \le 2\sqrt{2s}$, and $\|\xi\|_{2,(m)} \le \frac{\sqrt{m}}{C_\xi}$, where the last inequality used the assumed bounds

$$\left\|\xi_{(m)}\right\|_2 \le R_\xi\sqrt{m},\ \frac{R_\xi}{R} \le \frac{1}{C_\xi}.$$

Hence, by the assumptions on $\mathcal{O}_{\text{nstep}}$, it will not fail for such inputs unless $\|v\|_2 \ge \frac{1}{12}$ is violated, except with probability $\le \delta'$. If the check in Line 10 fails, then except with probability $\le \delta'$, the conclusion $\|x_T - x^\star\|_2 \le \frac{R}{6}$ follows analogously to Lemma 170, since $v = \frac{1}{R}(x - x^\star)$.

The other case's correctness follows identically to the proof of Lemma 170, except for one difference: to lower bound the progress term (9.24), we use the assumption (9.42) which shows

$$2\eta R\left\langle\gamma_t, x_t - x^\star\right\rangle = 2\eta R\sum_{i\in[n]} w_i\widetilde{\Delta}_i\left\langle a_i, v\right\rangle = 2\eta R^2\sum_{i\in[n]} w_i\widetilde{\Delta}_i\Delta_i \ge 2\eta R^2 C_{\text{prog}}.$$

Hence, following the proof of Lemma 170 (and adjusting for constants), whenever the algorithm does not terminate we make at least a $\frac{50}{T}$ fraction of the progress towards $x^\star$, so in T iterations (assuming no step oracle failed) we will have $\|x_T - x^\star\|_2 \le \frac{R}{6}$.

Finally, the runtime follows from combining Lemma 177 (with constant failure probability) with a multiplicative overhead of $T \cdot N_{\text{trials}}$ due to the number of calls to the step oracle, contributing one additional logarithmic factor. We adjusted one of the $\log d$ terms to become a $\log \frac{d}{\delta}$ term to account for the runtime of Aggregate (see Claim 1). □

9.7.2 Designing a strong step oracle

In this section, we design a strong step oracle $\mathcal{O}_{\text{step}}(\widetilde{\Delta}, \mathbf{A})$ under Assumption 12. As in Section 9.6.2, our oracle iteratively builds a weight vector $\bar{w}$, and sets

$$\gamma_{\bar{w}} := \sum_{i \in [n]} \bar{w}_i \widetilde{\Delta}_i a_i.$$

We will use essentially the same potentials as in (9.27), defined in the following:

$$\widetilde{\Phi}_2(\bar{w}) := \sum_{i \in [n]} \bar{w}_i \widetilde{\Delta}_i^2, \ \widetilde{\Phi}_{\text{sqmax}}(\bar{w}) := \left(\min_{\|p\|_2 \le L \|\bar{w}\|_1} \text{sqmax}_\mu(\gamma_{\bar{w}} - p) \right) + \frac{\|\bar{w}\|_1}{4CLs}. \tag{9.46}$$

Algorithm 58: StrongStepOracle($\widetilde{\Delta}, \mathbf{A}, \delta$)

Input: $\widetilde{\Delta} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{n \times d}$ satisfying Assumption 11, $\delta \in (0,1)$;

Output: (w, γ) such that $\gamma = \sum_{i \in [n]} w_i \widetilde{\Delta}_i a_i$, and if there is $v \in \mathbb{R}^d$ with $\frac{1}{12} \le \|v\|_2 \le 1$ such that $\widetilde{\Delta} = \mathbf{A}v + \xi$ where $\left\|\xi_{(m)}\right\|_2 \le \frac{\sqrt{m}}{C_\xi}$, with probability $\ge 1 - \delta$, (9.17), (9.18) are satisfied with

$$C_{\text{prog}} = 1, \ C_2 = O(1).$$

Furthermore, the second condition in (9.18) is satisfied with the constant 24 rather than 6, and there is $C_\xi = O(1)$ such that (9.43) is also satisfied.

$C \leftarrow 3200$, $\mu \leftarrow \frac{1}{\sqrt{Cs \log d}}$, $\eta \leftarrow \frac{1}{K w_\infty^\star s \rho^2 \log d}$, $N' \leftarrow \lceil \log_2 \frac{2}{\delta} \rceil$;

for $0 \le k \le N'$ **do**

  $w_0 \leftarrow 0_n$, $N \leftarrow \lceil \frac{5Ln}{\eta} \rceil$;

  for $0 \le t \le N$ **do**

    if $\widetilde{\Phi}_2(w_t) \ge 1$ **then Return:** $\gamma \leftarrow \sum_{i \in [n]} [w_t]_i \widetilde{\Delta}_i a_i$, $w \leftarrow w_t$;

    Sample $i \sim_{\text{unif.}} [n]$;

    Compute (using Lemma 171) $d_t \in [0, \eta w_\infty^\star]$ maximizing to additive $O(\frac{\eta}{n})$

$$\Gamma_t(d) := \widetilde{\Phi}_2(w_t + d e_i) - Cs \widetilde{\Phi}_{\text{sqmax}}(w_t + d e_i)$$

    $w_{t+1} \leftarrow w_t + d_t e_i$;

Return: $\gamma \leftarrow 0_d$, $w \leftarrow 0_n$

Algorithm 58 is essentially identical to Algorithm 55 except for changes in constants. We further have the following which verifies the second property in the definition of strong step oracle.

Fact 26. *The distribution of w returned by Algorithm 58 is stochastically dominated by the distribution*

$$\eta w_\infty^* \text{Multinom}\left(\frac{5Ln}{\eta}, \left(\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}_{n}\right)\right)$$

Proof. Every time we inspect a row, we change the corresponding entry of w by at most ηw_∞^*. The result follows from the number of iterations in the algorithm and uniformity of sampling rows. □

To analyze Algorithm 58, we provide appropriate analogs of Lemmas 172 and 168. Because Algorithm 58 is very similar to Algorithm 55, we will largely omit the proof of the following statement, which follows essentially identically to the proof of Lemma 172 up to adjusting constants.

Lemma 180. *Assume that the constant K in Assumption 12 is sufficiently large, and that $\widetilde{\Delta} = \mathbf{A}v + \xi$ where v, ξ satisfy the norm conditions in Assumption 12. Then for any $\bar{w} \in \mathbb{R}^n_{\geq 0}$ such that $B(\bar{w}) \leq C^2\mu^2 \log d$, we have*

$$\mathbb{E}_{i \sim_{\text{unif.}[n]}}[B(\bar{w} + \eta w_i^\star)] \leq B(\bar{w}) + \frac{1}{2CLs} \cdot \frac{\eta}{n}.$$

Proof. The analysis is essentially identical to that of Lemma 172; we discuss only the main difference. To apply the Taylor expansion of the exponential, Lemma 172 required a bound that

$$\frac{1}{\mu}\left|z_j^{(i)}\right| = O\left(\frac{1}{\sqrt{\log d}}\right) \impliedby \left|z_j^{(i)}\right| = O\left(\frac{1}{\sqrt{s}\log d}\right),$$

for all $i \in [n]$ and $j \in [d]$. Note that in the setting of Lemma 172, we took $z_j^{(i)} = \eta w_i^\star(\Delta_i a_{ij} - p_j^\star)$. Here, we will take $z_j^{(i)} = \eta w_i^\star(\widetilde{\Delta}_i a_{ij} - p_j^\star)$; bounds on all of these terms follow identically to in Lemma 172, except that $\widetilde{\Delta}_i = \langle a_i, v\rangle + \xi_i$, so we need to show

$$\eta w_i^\star \rho |\xi_i| = O\left(\frac{1}{\sqrt{s}\log d}\right).$$

This follows since $\eta \leq \frac{1}{\log d}$ and $|\xi_i| = O(\sqrt{m})$ by assumption. Hence, as $w_i^\star \leq \frac{1}{m}$ by definition of m, this is equivalent to $m = \Omega(s\rho^2)$, an explicit assumption we make. □

We now give a full analysis of Algorithm 58, patterned off of Lemma 168.

Lemma 177. *Suppose $\mathbf{A}$ satisfies Assumption 12. Algorithm 58 is a $(C_{\text{prog}}, C_2, C_\xi, \delta)$ strong step oracle* StrongStepOracle *for $(\widetilde{\Delta}, \mathbf{A})$ with*

$$C_{\text{prog}} = \Omega(1),\ C_2 = O(1),\ C_\xi = O(1),\ \delta = \frac{1}{2}\left(\frac{C_2}{10^5 C_{\text{prog}}}\right)^2,$$

running in time

$$O\left(\left(nd\log^3(nd\rho)\log\frac{1}{\delta}\right)\cdot\left(w_\infty^\star s\rho^2\log^2\frac{d}{\delta}\right)\right).$$

Proof. The analysis is essentially identical to that of Algorithm 55 in Lemma 168; we discuss differences here. First, the stochastic domination condition follows from Fact 26 for sufficiently large C_ξ, K.

For the remaining properties, since the algorithm runs $N' \geq \log_2 \frac{2}{\delta}$ times independently, it suffices to show each run meets Definition 42 with probability $\geq \frac{1}{2}$ under the events of Assumption 12, assuming there exists the desired decomposition $\widetilde{\Delta} = \mathbf{A}v + \xi$ in the sense of Assumption 12. Union bounding with the failure probability in Fact 26 yields the overall failure probability.

Correctness. As in Lemma 168, it is straightforward to see that $\widetilde{\Phi}_2$ is 1-Lipschitz, since the value of η is smaller than that used in Algorithm 55. The termination condition in iteration T then again implies $\widetilde{\Phi}_2(w_T) \geq 1$, and $\widetilde{\Phi}_{\mathrm{sqmax}}(w_T) \leq \frac{3}{Cs}$. For $C = 3200$, this implies the short-flat decomposition with stronger parameters required by Definition 42, as well as the $\|w_T\|_1$ bound.

Success probability. As in Lemma 168, the expected growth in Φ_t in any iteration where $\Pr[\widetilde{\Phi}_2(w_t) \geq 1] \leq \frac{1}{2}$ is $\geq \frac{\eta}{4Ln}$. Hence, running for $\geq \frac{5Ln}{\eta}$ iterations and using $\Phi_t - \Phi_0 \leq 2$ yields the claim.

Runtime. This follows identically to the analysis in Lemma 168.

□

9.7.3 Equivalence between Assumption 12 and RIP

In this section, we prove Lemma 178, restated here for completeness. The proof will build heavily on our previous developments in the noiseless case, as shown in Section 9.6.3.

Lemma 178. *Suppose $\mathbf{A} \in \mathbb{R}^{n\times d}$ is $(\rho, w_\infty^\star)$-wRIP with a suitable choice of constants in the RIP parameters in Definition 38. Then, $\mathbf{A}$ also satisfies Assumption 12.*

Proof. The analysis is largely similar to the analysis of Lemma 175; we will now discuss the differences here, which are introduced by the presence of the noise term ξ. There are three components to discuss: the upper and lower bounds in (9.44), and the decomposition (9.45).

Regarding the bounds in (9.19), by changing constants appropriately in Definition 38, we can assume that $\mathbf{A}$ satisfies the second property in Assumption 11 with the parameters $\frac{4}{L}$ and $\frac{L}{4}$. In particular, for $\Delta = \mathbf{A}v$, we then have

$$\frac{4}{L} \leq \sum_{i\in[n]} w_i^\star \Delta_i^2 \leq \frac{L}{4}.$$

Recall that $\widetilde{\Delta} = \Delta + \xi$ for some $\left\|\xi_{(m)}\right\|_2 \le \frac{\sqrt{m}}{C_\xi}$. Hence,

$$\begin{aligned}\sum_{i\in[n]} w_i^\star \widetilde{\Delta}_i^2 &\le 2\sum_{i\in[n]} w_i^\star \Delta_i^2 + 2\sum_{i\in[n]} w_i^\star \xi_i^2 \\ &\le \frac{L}{2} + 2\left(\frac{1}{m}\left\|\xi_{(m)}\right\|_2^2\right) \le L,\end{aligned}$$

for an appropriately large $C_\xi^2 \ge \frac{4}{L}$. Here the first inequality used $(a+b)^2 \le 2a^2 + 2b^2$, and the second inequality used that the largest $\sum_{i\in[n]} w_i^\star \xi_i^2$ can be subject to $\|w^\star\|_1 = 1$ and $\|w^\star\|_\infty \le \frac{1}{m}$ is attained by greedily choosing the m largest coordinates of ξ by their magnitude, and setting $w_i^\star = \frac{1}{m}$ for those coordinates. This gives the upper bound in Assumption 12, and the lower bound follows similarly: for appropriately large $C_\xi^2 \ge \frac{L}{2}$,

$$\begin{aligned}\sum_{i\in[n]} w_i^\star \widetilde{\Delta}_i^2 &\ge \frac{1}{2}\sum_{i\in[n]} w_i^\star \Delta_i^2 - \frac{1}{2}\sum_{i\in[n]} w_i^\star \xi_i^2 \\ &\ge \frac{2}{L} - \frac{1}{2}\left(\frac{1}{m}\left\|\xi_{(m)}\right\|_2^2\right) \ge \frac{1}{L}.\end{aligned}$$

Lastly, for the decomposition required by (9.45), we will use the decomposition of Lemma 169 for the component due to $\sum_{i\in[n]} w_i^\star \Delta_i a_i$; in particular, assume by adjusting constants that this component has a $(\frac{L}{2}, \frac{1}{2K\sqrt{s}})$ short-flat decomposition. It remains to show that

$$\sum_{i\in[n]} w_i^\star \xi_i a_i = \mathbf{A}^\top \mathbf{W}^\star \xi.$$

also admits a $(\frac{L}{2}, \frac{1}{2K\sqrt{s}})$ short-flat decomposition, at which point we may conclude by the triangle inequality. Let $u = (\mathbf{W}^\star)^{\frac{1}{2}}\xi$; from earlier, we bounded

$$\|u\|_2^2 \le \frac{1}{m}\left\|\xi_{(m)}\right\|_2^2 \implies \|u\|_2 \le \frac{1}{C_\xi}.$$

Hence, applying Lemma 174 using the RIP matrix $(\mathbf{W}^\star)^{\frac{1}{2}}\mathbf{A}$ with appropriate parameters yields the conclusion, for large enough C_ξ. In particular, the ℓ_2-bounded part of the decomposition follows from Lemma 174, and the proof of the ℓ_∞-bounded part is identical to the proof in Lemma 175. □

9.7.4 Putting it all together

We now prove our main result on noisy recovery.

Theorem 67. *Let $\delta \in (0,1)$, and suppose $R_0 \ge \|x^\star\|_2$ for s-sparse $x^\star \in \mathbb{R}^d$. Further, suppose $\mathbf{A} \in \mathbb{R}^{n\times d}$ is $(\rho, w_\infty^\star)$-wRIP and $b = \mathbf{A}x^\star + \xi$, and $R_1 \ge R_\xi := \left\|\xi_{(m)}\right\|_2$. Then with probability at*

least $1-\delta$*, Algorithm 56 using Algorithm 56 as a noisy step oracle computes* $\hat{x}$ *satisfying*

$$\|\hat{x}-x^\star\|_2 \le R_{\text{final}} = \Theta(R_\xi),$$

in time

$$O\left(\left(ndw_\infty^\star s\log^4(nd\rho)\log^2\left(\frac{d}{\delta}\cdot\log\left(\frac{R_0}{R_{\text{final}}}\right)\log\left(\frac{R_1}{R_{\text{final}}}\right)\right)\right)\cdot\rho^2\log\left(\frac{R_0}{R_{\text{final}}}\right)\log\left(\frac{R_1}{R_{\text{final}}}\right)\right).$$

Proof. Our algorithm will iteratively maintain a guess R_{guess} on the value of $\frac{1}{\sqrt{m}}\left\|\xi_{(m)}\right\|_2$, initialized at $R_{\text{guess}} \leftarrow R_1$. For each value of $R_{\text{guess}} \ge R_\xi$, the hypothesis of Algorithm 56 is satisfied, and hence using a strategy similar to the proof of Theorem 66 (but terminating at accuracy $R = O(R_{\text{guess}})$ where the constant is large enough to satisfy the assumption $R \ge C_\xi R_{\text{guess}}$) results in an estimate at distance R with probability at least $1-\delta$, with runtime

$$O\left(\left(ndw_\infty^\star s\rho^2\log^4(nd\rho)\log^2\left(\frac{d}{\delta}\cdot\log\left(\frac{R_0}{R_{\text{final}}}\right)\right)\right)\cdot\rho^2\log\left(\frac{R_0}{R_{\text{final}}}\right)\right).$$

The runtime above follows from Lemma 179.

Our overall algorithm repeatedly halves R_{guess}, and outputs the last point returned by a run of the algorithm where it can certify a distance bound to $x^\star$ of $R = C_\xi R_{\text{guess}}$. We use R_{final} to denote $C_\xi R_{\text{guess}}$ on the last run. Clearly for any $R_{\text{guess}} \ge R_\xi$ this certification will succeed, so we at most lose a factor of 2 in the error guarantee as we will have $R_{\text{final}} \le 2C_\xi R_\xi$. The final runtime follows from adjusting δ by a factor of $O(\log \frac{R_1}{R_{\text{final}}})$ to account for the multiple runs of the algorithm. □

Chapter 10

Towards an Oracle Complexity of Sampling

This chapter is based on [344, 346, 345], with Yin Tat Lee and Ruoqi Shen.

10.1 Introduction

Since its study was pioneered by the celebrated randomized convex body volume approximation algorithm of Dyer, Frieze, and Kannan [212], designing samplers for logconcave distributions has been a very active area of research in theoretical computer science and statistics with many connections to other fields. In a generic form, the problem can be stated as: sample from a distribution whose negative log-density is convex, under various access models to the distribution.

Developing efficient algorithms for sampling from *structured* logconcave densities is a topic that has received significant recent interest due to its widespread practical applications. There are many types of structure which densities commonplace in applications may possess that are exploitable for improved runtimes. Examples of such structure include derivative bounds ("well-conditioned densities") and various types of separability (e.g. "composite densities" corresponding to possibly non-smooth regularization or restrictions to a set, and "logconcave finite sums" corresponding to averages over multiple data points).[1] Building an algorithmic theory for sampling these latter two families, which are not well-understood in the literature, is a primary motivation of this chapter.

There are strong parallels between the types of structured logconcave families garnering recent attention and the classes of convex functions known to admit efficient first-order optimization algorithms. Notably, gradient descent and its accelerated counterpart [417] are well-known to quickly

[1] We make this terminology more precise in Section 10.2.1, which contains various definitions used in this paper.

optimize a well-conditioned function, and have become ubiquitous in both practice and theory. Similarly, various methods have been developed for efficiently optimizing non-smooth but structured composite objectives [66] and well-conditioned finite sums [17].

Logconcave sampling and convex optimization are intimately related primitives (cf. e.g. [75, 2]), so it is perhaps unsurprising that there are analogies between the types of structure algorithm designers may exploit. Nonetheless, our understanding of the complexity landscape for sampling is quite a bit weaker in comparison to counterparts in the field of optimization; few lower bounds are known for the complexity of sampling tasks, and obtaining stronger upper bounds is an extremely active research area (contrary to optimization, where matching bounds exist in many cases). Moreover (and perhaps relatedly), the toolkit for designing logconcave samplers is comparatively lacking; for many important primitives in optimization, it is unclear if there are analogs in sampling, possibly impeding improved bounds.

Broadly speaking, this chapter presents improved upper and lower bounds aimed towards advancing the program of understanding the following question for various natural, practically-motivated types of structure and oracle access.

What is the oracle complexity of sampling from various structured density families?

The upper bounds presented in this chapter broadly fall under the themes of (1) understanding which types of structured logconcave distributions admit efficient samplers, and (2) leveraging connections between optimization and sampling for algorithm design. The lower bounds presented in this chapter aim to characterize the complexity of sampling from logconcave densities exhibiting a basic form of structure (well-conditioning), by demonstrating hardness for various standard sampling frameworks. Altogether, we address the oracle complexity of sampling on each of the following fronts, which constitute the primary technical contributions of this chapter.

1. We give a general reduction framework for bootstrapping samplers with mixing times with polynomial dependence on a conditioning measure κ to mixing times with linear dependence on κ. The framework is heavily motivated by a perspective on *proximal point methods* in structured convex optimization as instances of optimizing composite objectives, and leverages this connection via a surprisingly simple analysis (cf. Theorem 68).

2. We develop novel "base samplers" for composite logconcave distributions and logconcave finite sums (cf. Theorems 69, 70). The former is the first composite sampler with stronger guarantees than those known in the general logconcave setting. The latter constitutes the first high-accuracy finite sum sampler whose gradient query complexity improves upon the naïve strategy of querying full gradients of the negative log-density in each iteration. Using our novel base samplers within our reduction framework, we obtain state-of-the-art samplers for all of the aforementioned structured families, i.e. well-conditioned, composite, and finite sum,

as Corollaries 45, 46, and 47.

3. We give lower bounds on the performance of two of the most popular sampling methods in practice, the Metropolis-adjusted Langevin algorithm (MALA) and multi-step Hamiltonian Monte Carlo (HMC) with a leapfrog integrator, when applied to well-conditioned distributions. Our main lower bound result, Theorem 73, is a nearly-tight lower bound of $\widetilde{\Omega}(\kappa d)$ on the mixing time of MALA from an exponentially warm start, matching upper bounds obtained in this chapter up to logarithmic factors and answering an open question of [137]. We also show that a polynomial dependence on dimension is necessary for the relaxation time of HMC under any number of leapfrog steps, and bound the gains achievable by changing the step count (see Theorem 74 for a representative result).

We formally state our upper bound results in Section 10.1.1, our lower bound results in Section 10.1.2, and situate each in the literature in Section 10.1.3. Section 10.1.4 is a technical overview of our approaches for developing our base samplers for composite and finite sum-structured densities, as well as our reduction framework (Section 10.1.4). Section 10.1.5 is a technical overview for our various lower bound constructions. Finally, Section 10.1.6 gives a roadmap for the rest of the chapter.

10.1.1 Our results: upper bounds

Before stating our results, we first require the notion of a restricted Gaussian oracle, whose definition is a key ingredient in giving our reduction framework as well as our later composite samplers.

Definition 43 (Restricted Gaussian oracle)**.** *$\mathcal{O}(\lambda, v)$ is a* restricted Gaussian oracle *(RGO) for convex $g : \mathbb{R}^d \to \mathbb{R}$ if it returns*

$$\mathcal{O}(\lambda, v) \leftarrow \text{sample from the distribution with density} \propto \exp\left(-\frac{1}{2\lambda}\|x - v\|_2^2 - g(x)\right).$$

In other words, an RGO asks to sample from a multivariate Gaussian (with covariance a multiple of the identity), "restricted" by some convex function g. Intuitively, if we can reduce a sampling problem for the density $\propto \exp(-g)$ to calling an RGO a small number of times with a small value of λ, each RGO subproblem could be much easier to solve than the original problem. This can happen for a variety of reasons, e.g. if the regularized density is extremely well-conditioned, or because it inherits concentration properties of a Gaussian. This idea of reducing a sampling problem to multiple subproblems, each implementing an RGO, underlies the framework of Theorem 68. Because the idea of regularization by a large Gaussian component repeatedly appears in this paper, we make the following more specific definition for convenience, which lower bounds the size of the Gaussian.

Definition 44 (η-RGO)**.** *We say $\mathcal{O}(\lambda, v)$ is an η-restricted Gaussian oracle (η-RGO) if it satisfies Definition 43 with the restriction that parameter λ is required to be always at most η in calls to $\mathcal{O}$.*

Variants of our notion of an RGO have implicitly appeared previously [149, 401], and efficient RGO implementation was a key subroutine in the fastest sampling algorithm for general logconcave distributions [149]. It also extends a similar oracle used in composite optimization, which we will discuss shortly. However, the explicit use of RGOs in a framework such as Theorem 68 is a novel technical innovation of our work, and we believe this abstraction will find further uses.

Proximal reduction framework. In Section 10.3, we prove correctness of our proximal reduction framework, whose guarantees are stated in the following Theorem 68.

Theorem 68. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f_{\text{oracle}}(x))$ such that f_{oracle} is μ-strongly convex, and let $\epsilon \in (0,1)$. Let $\eta \le \frac{1}{\mu}$, $T = \Theta(\frac{1}{\eta\mu}\log\frac{d}{\eta\mu\epsilon})$ for some $\beta \ge 1$, and $\mathcal{O}$ be a η-RGO for f_{oracle}. Algorithm 59, initialized at the minimizer of f_{oracle}, runs in T iterations, each querying $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

In other words, if we can implement an η-RGO for a μ-strongly convex function f_{oracle} in time $\mathcal{T}_{\text{RGO}}$, we can sample from $\exp(-f_{\text{oracle}})$ in time $\widetilde{O}(\frac{1}{\eta\mu}\cdot\mathcal{T}_{\text{RGO}})$. To highlight the power of this reduction framework, suppose there was an existing sampler $\mathcal{A}$ for densities $\propto \exp(-f)$ with mixing time $\widetilde{O}(\kappa^{10}\sqrt{d})$, where $f:\mathbb{R}^d\to\mathbb{R}$ is L-smooth, μ-strongly convex, and has condition number $\kappa = \frac{L}{\mu}$ (cf. Section 10.2.1 for definitions).[2] Choosing $\eta = \frac{1}{L}$ and $f_{\text{oracle}} \leftarrow f$ in Theorem 68 yields a sampler whose mixing time is $\widetilde{O}(\kappa\cdot\mathcal{T}_{\text{RGO}})$, where $\mathcal{T}_{\text{RGO}}$ is the cost of sampling from a density proportional to

$$\exp\left(-\frac{L}{2}\left\|x-v\right\|_2^2 - f(x)\right),$$

for some $v\in\mathbb{R}^d$. However, observe that this distribution has a negative log-density with constant condition number $\frac{L+L}{L+\mu} \le 2$! By using $\mathcal{A}$ as our RGO, we have $\mathcal{T}_{\text{RGO}} = \widetilde{O}(\sqrt{d})$, and the overall mixing time is $\widetilde{O}(\kappa\sqrt{d})$. Leveraging Theorem 68 in applications, we obtain the following new results, improving mixing of various "base samplers" which we bootstrap as RGOs for regularized densities.

Well-conditioned densities. In an earlier version of part of this chapter [344], it was shown that a variant of Metropolized Hamiltonian Monte Carlo obtains a mixing time of $\widetilde{O}(\kappa d\log^3\frac{\kappa d}{\epsilon})$ for sampling a density on $\mathbb{R}^d$ with condition number κ. The analysis of [344] was somewhat delicate, and required reasoning about conditioning on a nonconvex set with desirable concentration properties. In Section 10.4.1, we prove Corollary 45, improving [344] by roughly a logarithmic factor with a significantly simpler analysis. For completeness, we include the only technical fact required in its proof from [344], a concentration bound on the norm of the gradient of a logsmooth density, in Appendix I.6.

Corollary 45. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x)\propto\exp(-f(x))$ such that f is L-smooth and μ-strongly convex, and let $\epsilon\in(0,1)$, $\kappa = \frac{L}{\mu}$. Assume access to $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d} f(x)$. Algorithm 59 with $\eta = \frac{1}{8Ld\log(\kappa)}$ using Algorithm 60 as a restricted Gaussian oracle for f uses $O(\kappa d\log\kappa\log\frac{\kappa d}{\epsilon})$ gradient queries in expectation, and obtains ϵ total variation distance to π.*

[2] No sampler with mixing time scaling as $\text{poly}(\kappa)\sqrt{d}$ is currently known.

We include Corollary 45 as a warmup for our more complicated results, as a way to showcase the use of our reduction framework in a slightly different way than the one outlined earlier. In particular, in proving Corollary 45, we will choose a significantly smaller value of η, at which point a simple rejection sampling scheme implements each RGO with expected constant gradient queries.

We give another algorithm matching Corollary 45 with a deterministic query complexity bound as Corollary 49. The algorithm of Corollary 49 is interesting in that it is entirely a *zeroth-order* algorithm, and does not require access to a gradient oracle. To our knowledge, in the well-conditioned optimization setting, no zeroth-order query complexities better than roughly $\sqrt{\kappa}d$ are known, e.g. simulating accelerated gradient descent with a value oracle; thus, our sampling algorithm has a query bound off by only $\tilde{O}(\sqrt{\kappa})$ from the best-known optimization algorithm. We are hopeful this result may help in the search for query lower bounds for structured logconcave sampling.

Composite densities with a restricted Gaussian oracle. In Section 10.5, we develop a sampler for densities on $\mathbb{R}^d$ proportional to $\exp(-f(x) - g(x))$, where f has condition number κ and g admits a restricted Gaussian oracle $\mathcal{O}$. We state its guarantees here.

Theorem 69. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp\left(-f(x) - g(x)\right)$ such that f is L-smooth and μ-strongly convex, and let $\epsilon \in (0, 1)$. Let $\eta \leq \frac{1}{32L\kappa d \log(\kappa/\epsilon)}$ (where $\kappa = \frac{L}{\mu}$), $T = \Theta(\frac{1}{\eta\mu}\log(\frac{\kappa d}{\epsilon}))$, and let $\mathcal{O}$ be a η-RGO for g. Further, assume access to the minimizer $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d}\{f(x) + g(x)\}$. There is an algorithm which runs in T iterations in expectation, each querying a gradient oracle of f and $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

The assumption that the composite component g admits an RGO can be thought of as a measure of "simplicity" of g. This mirrors the widespread use of a proximal oracle as a measure of simplicity in the context of composite optimization [66], which we now define.

Definition 45 (Proximal oracle)**.** *$\mathcal{O}(\lambda, v)$ is a* proximal oracle *for convex $g : \mathbb{R}^d \to \mathbb{R}$ if it returns*

$$\mathcal{O}(\lambda, v) \leftarrow \operatorname{argmin}_{x\in\mathbb{R}^d}\left\{\frac{1}{2\lambda}\|x - v\|_2^2 + g(x)\right\}.$$

Many regularizers g in defining composite optimization objectives, which are often used to enforce a quality such as sparsity or "simplicity" in a solution, admit efficient proximal oracles. In particular, if the proximal oracle subproblem admits a closed form solution (or otherwise is computable in $O(d)$ time), the regularized objective can be optimized at essentially no asymptotic loss. It is readily apparent that our RGO (Definition 43) is the extension of Definition 45 to the sampling setting. In [401], a variety of regularizations arising in practical applications including coordinate-separable g (such as restrictions to a coordinate-wise box, e.g. for a Bayesian inference task where we have side information on the ranges of parameters) and ℓ_1 or group Lasso regularized densities were shown to admit RGOs. Our composite sampling results achieve a similar "no loss" phenomenon for such regularizations, with respect to existing well-conditioned samplers.

By choosing the largest possible value of η in Theorem 69, we obtain an iteration bound of $\widetilde{O}(\kappa^2 d)$. In Section 10.4.2, we prove Corollary 46, which improves Theorem 69 by roughly a κ factor.

Corollary 46. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f(x)-g(x))$ such that f is L-smooth and μ-strongly convex, and let $\epsilon \in (0,1)$, $\kappa = \frac{L}{\mu}$. Assume access to $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d}\{f(x)+g(x)\}$ and let $\mathcal{O}$ be a restricted Gaussian oracle for g. There is an algorithm (Algorithm 59 using Theorem 69 as a restricted Gaussian oracle) which runs in $O(\kappa d \log^3 \frac{\kappa d}{\epsilon})$ iterations in expectation, each querying a gradient of f and $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

To sketch the proof, choosing $\eta = \frac{1}{L}$ in Theorem 68 yields an algorithm running in $\widetilde{O}(\frac{1}{\eta\mu}) = \widetilde{O}(\kappa)$ iterations. In each iteration, the RGO subproblem asks to sample from the distribution whose negative log-density is $f(x) + g(x) + \frac{L}{2}\|x - v\|_2^2$ for some $v \in \mathbb{R}^d$, so we can call Theorem 69, where the "well-conditioned" portion $f(x) + \frac{L}{2}\|x - v\|_2^2$ has constant condition number. Thus, Theorem 69 runs in $\widetilde{O}(d)$ iterations to solve the subproblem, yielding the result. In fact, Corollary 46 nearly matches Corollary 45 in the case $g = 0$ uniformly. Surprisingly, this recovers the runtime of [344] without appealing to strong gradient concentration bounds (e.g. [344], Theorem 3.2).

Logconcave finite sums. In Section 10.6, we initiate the study of mixing times for sampling logconcave finite sums with polylogarithmic dependence on accuracy. We give the following result.

Theorem 70. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-F(x))$, where $F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ is μ-strongly convex, f_i is L-smooth and convex $\forall i \in [n]$, $\kappa = \frac{L}{\mu}$, and $\epsilon \in (0,1)$. Assume access to $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d} F(x)$. Algorithm 64 uses $O\left(\kappa^2 d \log^4 \frac{n\kappa d}{\epsilon}\right)$ value queries to summands $\{f_i\}_{i\in[n]}$, and obtains ϵ total variation distance to π.*

For a zeroth-order algorithm, Theorem 70 serves as a surprisingly strong baseline as it nearly matches the previously best-known bound for zeroth-order well-conditioned sampling when $n = 1$; however, when e.g. $\kappa \approx d$, the complexity bound is at least cubic. By using Theorem 70 within the framework of Theorem 68, we obtain the following improved result.

Corollary 47 (Improved first-order logconcave finite sum sampling)**.** *In the setting of Theorem 70, Algorithm 59 using Algorithm 64 and SVRG [300] as a restricted Gaussian oracle for F uses*

$$O\left(n \log\left(\frac{n\kappa d}{\epsilon}\right) + \kappa\sqrt{nd}\log^{3.5}\left(\frac{n\kappa d}{\epsilon}\right) + \kappa d \log^5\left(\frac{n\kappa d}{\epsilon}\right)\right) = \widetilde{O}\left(n + \kappa \max\left(d, \sqrt{nd}\right)\right)$$

queries to first-order oracles for summands $\{f_i\}_{i\in[n]}$, and obtains ϵ total variation distance to π.

Corollary 47 has several surprising properties. First, its bound when $n = 1$ gives yet another way of (up to polylogarithmic factors) recovering the runtime of [344] without gradient concentration. Second, up to a $\widetilde{O}(\max(1, \sqrt{\frac{n}{d}}))$ factor, it is essentially the best runtime one could hope for without an improvement when $n = 1$. This is in the sense that $\widetilde{O}(\kappa d)$ is the best known runtime for $n = 1$ (with near-matching lower bounds, developed later in this chapter), and to our knowledge every efficient

well-conditioned sampler requires minimizer access, i.e. $\widetilde{O}(n)$ gradient queries [539]. Interestingly, when $n = 1$, Algorithm 64 can be significantly simplified, and becomes the standard Metropolized random walk [210]; this yields Corollary 49, an algorithm attaining the iteration complexity of Corollary 45 while only querying a value oracle for f.

10.1.2 Our results: lower bounds

The restricted problem of sampling from a particular family of distributions, which we call "well-conditioned distributions," has garnered a substantial amount of recent research effort from the algorithmic learning and statistics communities. This specific family is interesting for a number of reasons. First of all, it is *practically relevant*: Bayesian methods have found increasing use in machine learning applications [60], and many distributions arising from these methods are well-conditioned, such as multivariate Gaussians, mixture models with small separation, and densities arising from Bayesian logistic regression with a Gaussian prior [210]. Moreover, for several of the most widely-used sampler implementations in popular packages [1, 113], such as the Metropolis-adjusted Langevin algorithm (MALA) and Hamiltonian Monte Carlo (HMC), the target density having a small condition number is in some sense a *minimal assumption for known provable guarantees*.

Finally, the highly-documented success of first-order (gradient-based) methods in optimization [65], which are particularly favorable in the well-conditioned setting, has driven a recent interest in connections between optimization and sampling. Exploring this connection has been highly fruitful: since seminal work of [301], which demonstrated that the continuous-time Langevin dynamics which MALA and HMC discretize has an interpretation as gradient descent on density space, a flurry of work including [153, 130, 210, 156, 206, 205, 128, 129, 480, 402, 137], as well as upper bound results in this chapter (see Section 10.1.1), have obtained improved upper bounds for the mixing of various discretizations of the Langevin dynamics for sampling from well-conditioned densities. Many of these works have drawn inspiration from techniques from first-order optimization.

On the other hand, demonstrating *lower bounds* on the complexity of sampling tasks (in the well-conditioned regime or otherwise) has proven to be a remarkably challenging problem. To our knowledge, there are very few unconditional lower bounds for sampling tasks (i.e. the complexity of sampling from a family of distributions under some query model). This is in stark contrast to the theory of optimization, where there are matching upper and lower bounds for a variety of fundamental tasks and query models, such as optimization of a convex function under first-order oracle access [418]. This gap in the development of the algorithmic theory of sampling is the primary motivation for our work, wherein we aim to answer the following more restricted question.

What is the complexity of the popular sampling methods, MALA and HMC,
for sampling well-conditioned distributions?

The problem we study is still less general than *unconditional query lower bounds* for sampling, in that our lower bounds are *algorithm-specific*; we characterize the performance of particular algorithms for sampling a distribution family. However, we believe asking this question, and developing an understanding of it, is an important first step towards a theory of complexity for sampling. On the one hand, lower bounds for specific algorithms highlight weaknesses in their performance, pinpointing their shortcomings in attaining faster rates. This is useful from an algorithm design perspective, as it clarifies what the key technical barriers are to overcome. On the other hand, the hard instances which arise in designing lower bounds may have important structural properties which pave the way to stronger and more general (i.e. *algorithm-agnostic*) lower bounds.

For these reasons, in this work we focus on characterizing the complexity of the MALA and HMC algorithms (see Sections 10.2.3 and 10.2.4 for algorithm definitions), which are often the samplers of choice in practice, by lower bounding their performance when they are used to sample from densities proportional to $\exp(-f(x))$, where $f : \mathbb{R}^d \to \mathbb{R}$ has a finite condition number. In particular, f is said to have a condition number of $\kappa < \infty$ if it is L-smooth and μ-strongly convex (has second derivatives in all directions in the range $[\mu, L]$), where $\kappa = \frac{L}{\mu}$. We will also overload this terminology and say the density itself has condition number κ. We call such a density (with finite κ) "well-conditioned." Finally, we explicitly assume throughout that $\kappa = O(d^4)$, as otherwise in light of our lower bounds the general-purpose logconcave samplers of [373, 295, 127] are preferable.

Our primary contribution is a nearly-tight characterization of the performance of MALA for sampling from two high-dimensional distribution families without a warm start assumption: well-conditioned Gaussians, and the more general family of well-conditioned densities. In Sections 10.7 and 10.8, we prove the following two lower bounds on MALA's complexity, which is a one-parameter algorithm (for a given target distribution) depending only on step size. We also note that we fix a scale $[1, \kappa]$ on the eigenvalues of the function Hessian up front, because otherwise the non-scale-invariance of the step size can be exploited to give much more trivial lower bounds (cf. Appendix I.7).

Theorem 71. *For every step size, there is a target Gaussian on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of MALA is $\Omega(\frac{\kappa\sqrt{d}}{\sqrt{\log d}})$.*

Theorem 72. *For every step size, there is a target density on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of MALA is $\Omega(\frac{\kappa d}{\log d})$.*

To give more context on Theorems 71 and 72, MALA is an example of a Metropolis-adjusted Markov chain, which in every step performs updates which preserve the stationary distribution. Indeed, it can be derived by applying a Metropolis filter on the standard forward Euler discretization of the Langevin dynamics, a stochastic differential equation with stationary density $\propto \exp(-f(x))$:

$$dx_t = -\nabla f(x_t)dt + \sqrt{2}dW_t,$$

where W_t is Brownian motion. Such *Metropolis-adjusted* methods typically provide total variation distance guarantees, and attain logarithmic dependence on the target accuracy.[3] The mixing of such chains is governed by their relaxation time, also known as the inverse *spectral gap* (the difference between 1 and the second-largest eigenvalue of the Markov chain transition operator).

However, in the continuous-space setting, it is not always clear how to relate the relaxation time to the *mixing time*, which we define as the number of iterations it takes to reach total variation distance $\frac{1}{e}$ from the stationary distribution from a given warm start (we choose $\frac{1}{e}$ for consistency with the literature, but indeed any constant bounded away from 1 will do). There is an extensive line of research on when it is possible to relate these two quantities (see e.g. [52]), but typically these arguments are tailored to properties of the specific Markov chain, causing relaxation time lower bounds to not be entirely satisfactory in some cases. We thus complement Theorems 71 and 72 with a *mixing time* lower bound from an exponentially warm start, as follows.

Theorem 73. *For every step size, there is a target density on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that MALA initialized at an $\exp(d)$-warm start requires $\Omega(\frac{\kappa d}{\log^2 d})$ iterations to reach e^{-1} total variation distance to the stationary distribution.*

We remark that Theorem 73 is the first *mixing time* lower bound for discretizations of the Langevin dynamics we are aware of, as other related lower bounds have primarily been on relaxation times [129, 344, 137]. Up to now, it is unknown how to obtain a starting distribution for a general distribution with condition number κ with warmness better than κ^d (which is obtained by the starting distribution $\mathcal{N}(x^*, \frac{1}{L}\mathbf{I})$ where L is the smoothness parameter and x^* is the mode).[4] A line of work [210, 128, 344] analyzed the performance of MALA under this warm start, culminating in a mixing time of $\widetilde{O}(\kappa d)$, where $\widetilde{O}$ hides logarithmic factors in κ, d, and the target accuracy. On the other hand, a recent work [137] demonstrated that MALA obtains a mixing time scaling as $\widetilde{O}(\text{poly}(\kappa)\sqrt{d})$, when initialized at a *polynomially* warm start,[5] and further showed that such a mixing time is tight (in its dependence on d). They posed as an open question whether it was possible to obtain $\widetilde{O}(\text{poly}(\kappa)d^{1-\Omega(1)})$ mixing from an explicit starting distribution.

We address this question by proving Theorem 73, showing that the $\widetilde{O}(\kappa d)$ rate of [344] for MALA (recovered by the upper bound developments of this chapter) applied to a κ-conditioned density is tight up to logarithmic factors from an explicit "bad" warm start. Concretely, to prove Theorems 71-73, in each case we exhibit an $\exp(-d)$-sized set according to the stationary measure where either the chain cannot move in $\text{poly}(d)$ steps with high probability, or must choose a very small step size. Beyond exhibiting a mixing bound, this demonstrates the subexponential warmness assumption in [137] is truly necessary for their improved bound. To our knowledge, this is the first

[3] We note this is in contrast with a different family of *unadjusted* discretizations, which are analyzed by coupling them with the stochastic differential equation they simulate (see e.g. [153, 130] for examples), at the expense of a polynomial dependence on the target accuracy; we focus on Metropolis-adjusted discretizations in this work.

[4] The warmness of a distribution is the worst-case ratio between the measures it and the stationary assign to a set.

[5] As discussed, it is currently unknown how to obtain such a warm start generically.

nearly-tight characterization of a specific sampling algorithm's performance in all parameters, and improves lower bounds of [137, 344]. It also implies that to go beyond $\widetilde{O}(\kappa d)$ mixing requires a subexponential warm start.

The lower bound statement of Theorem 73 is warmness-sensitive, and is of the following (somewhat non-standard) form: for $\beta = \exp(d)$, we provide a lower bound on the quantity

$$\inf_{\text{algorithm parameters}} \sup_{\substack{\text{starts of warmness } \leq \beta \\ \text{densities in target family}}} \text{mixing time of algorithm.}$$

In other words, we are allowed to choose both the hard density and starting distribution adaptively based on the algorithm parameters (in the case of MALA, our choices respond to the step size). We note that this type of lower bound is compatible with standard conductance-based upper bound analyses, which typically only depend on the starting distribution through the warmness parameter.

In Section 10.10, we further study the multi-step generalization of MALA, known in the literature as Hamiltonian Monte Carlo with a leapfrog integrator (which we refer to in this paper as HMC). In addition to a step size η, HMC is parameterized by a number of steps per iteration K; in particular, HMC makes K gradient queries in every step to perform a K-step discretization of the Langevin dynamics, before applying a Metropolis filter. It was recently shown in [128] that under higher derivative bounds, balancing η and K more carefully depending on problem parameters could break the apparent κd barrier of MALA, even from an exponentially warm start.

It is natural to ask if there is a stopping point for improving HMC. We demonstrate that HMC cannot obtain a better relaxation time than $\widetilde{O}(\kappa\sqrt{d}K^{-1})$ for any K, even when the target is a Gaussian. Since every HMC step requires K gradients, this suggests $\widetilde{\Omega}(\kappa\sqrt{d})$ queries are necessary.

Theorem 74. *For every step size and count, there is a target Gaussian on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of HMC is $\Omega(\frac{\kappa\sqrt{d}}{K\sqrt{\log d}})$.*

In Appendix I.8, we also give some lower bounds on how much increasing K can help the performance of HMC in the in-between range $\kappa\sqrt{d}$ to κd. In particular, we demonstrate that if $K \leq d^c$ for some constant $c \approx 0.1$, then the K-step HMC Markov chain can only improve the relaxation time of Theorem 74 by roughly a factor K^2, showing that to truly go beyond a κd relaxation time by more than a $d^{o(1)}$ factor, the step size must scale polynomially with the dimension (Proposition 84). We further demonstrate how to extend the mixing time lower bound of Theorem 73 in a similar manner, demonstrating formally for small K that (up to logarithmic factors) the gradient query complexity of HMC cannot be improved beyond κd by more than roughly a K factor (Proposition 85).

Our mixing lower bound technique in Theorem 73 does not directly extend to give a complementary mixing lower bound for Theorem 74 for all K, but we defer this to interesting future work.

10.1.3 Previous work

Logconcave sampling is a problem that, within the theoretical computer science field, has its origins in convex body sampling (a problem it generalizes). A long sequence of developments have made significant advances in the general model, where only convexity is assumed about the negative log-density, and only value oracle access is given. In this prior work discussion, we focus on more structured cases where all or part of the negative log-density has finite condition number, and refer the reader to [519, 370, 150] for an account on progress in the general case.

Well-conditioned densities. Significant recent efforts in the machine learning and statistics communities focused on obtaining provable guarantees for well-conditioned distributions, starting from pioneering work of [153], and continued in e.g. [130, 156, 129, 128, 210, 206, 205, 351, 402, 480, 344]. In this setting, many methods based on discretizations of continuous-time first-order processes (such as the Langevin dynamics) have been proposed. Typically, error guarantees come in two forms: either in the 2-Wasserstein (W_2) distance, or in total variation (TV). The line [210, 128, 344] has brought the gradient complexity for obtaining ϵ TV distance to $\widetilde{O}(\kappa d)$ where d is the dimension, by exploiting gradient concentration properties. For progress in complexities depending polynomially on ϵ^{-1}, attaining W_2 guarantees (typically incomparable to TV bounds), we defer to [480], the state-of-the-art using $\widetilde{O}(\kappa^{\frac{7}{6}}\epsilon^{-\frac{1}{3}} + \kappa\epsilon^{-\frac{2}{3}})$ queries to obtain W_2 distance $\epsilon\sqrt{d\mu^{-1}}$ from the target.[6] We note incomparable guarantees can be obtained by assuming higher derivative bounds (e.g. a Lipschitz Hessian); our work uses only the minimal assumption of bounded second derivatives.

Composite densities. Recent works have studied sampling from densities of the form (10.1), or its specializations (e.g. restrictions to a convex set). Several [443, 96, 74] are based on Moreau envelope or proximal regularization strategies, and demonstrate efficiency under more stringent assumptions on the structure of the composite function g, but under minimal assumptions obtain fairly large provable mixing times $\Omega(d^5)$. Proximal regularization algorithms have also been considered for non-composite sampling [537]. Another discretization strategy based on projections was studied by [101], but obtained mixing time $\Omega(d^7)$. Finally, improved algorithms for special constrained sampling problems have been proposed, such as simplex restrictions [276].

Of particular relevance and inspiration to this work is [401]. By generalizing and adapting Metropolized HMC algorithms of [210, 128], adopting a Moreau envelope strategy, and using (a stronger version of) the RGO access model, [401] obtained a runtime which in the best case scales as $\widetilde{O}\left(\kappa^2 d\right)$, similar to the guarantee of our base sampler in Theorem 69. However, this result required a variety of additional assumptions, such as access to the normalization factor of restricted Gaussians, Lipschitzness of g, warmness of the start, and various problem parameter tradeoffs. The general problem of sampling from (10.1) under minimal assumptions more efficiently than general-purpose logconcave algorithms is to the best of our knowledge unresolved (even under restricted Gaussian oracle access), a novel contribution of our mixing time bound. Our results also suggest that the

[6] Here, $\sqrt{d\mu^{-1}}$ is the effective diameter; this accuracy measure allows for scale-invariant W_2 guarantees.

Method	Gradient oracle complexity	
	$W_2 \le \epsilon$, $\mu = 1$	$W_2 \le \epsilon\sqrt{d\mu^{-1}}$
SAGA-LD [119]	$n + \frac{\kappa^{1.5}\sqrt{d}+\kappa d+Md}{\epsilon} + \frac{\kappa d^{4/3}}{\epsilon^{2/3}}$	$n + \frac{\kappa^{1.5}+\kappa\sqrt{d}+M\sqrt{d}}{\epsilon} + \frac{\kappa d^{2/3}}{\epsilon^{2/3}}$
SVRG-LD [119]	$n + \frac{\kappa^{1.5}\sqrt{d}+\kappa d+Md}{\epsilon} + \frac{\kappa d^{4/3}}{\epsilon^{2/3}}$	$n + \frac{\kappa^3}{\epsilon^2} + \frac{\kappa^{1.5}+M\sqrt{d}}{\epsilon}$
CV-ULD [119]	$n + \frac{\kappa^4 d^{1.5}}{\epsilon^3}$	$n + \frac{\kappa^4}{\epsilon^3}$
SVRG-LD [554]	$n + \frac{\kappa^{1.5}\sqrt{d}+Md}{\epsilon} + \frac{\kappa\sqrt{nd}}{\epsilon}$	$n + \frac{\kappa^{1.5}+M\sqrt{d}}{\epsilon} + \frac{\kappa\sqrt{n}}{\epsilon}$
State-of-the-art, $n = 1$ [480]	$\frac{\kappa^{7/6}d^{1/6}}{\epsilon^{1/3}} + \frac{\kappa d^{1/3}}{\epsilon^{2/3}}$	$\frac{\kappa^{7/6}}{\epsilon^{1/3}} + \frac{\kappa}{\epsilon^{2/3}}$

Method	Gradient oracle complexity (TV $\le \epsilon$)
Corollary 47	$n + \kappa d + \kappa\sqrt{nd}$
State-of-the-art, $n = 1$ [344]	κd

Table 10.1: Complexity of sampling from $e^{-F(x)}$ where $F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x)$ on $\mathbb{R}^d$ is μ-strongly convex, each f_i is convex and L-smooth, and $\kappa = \frac{L}{\mu}$. For relevant lines, M is the Lipschitz constant of the Hessian $\nabla^2 F$, which our algorithm has no dependence on. Complexity is measured in terms of the number of calls to f_i or ∇f_i for summands $\{f_i\}_{i\in[n]}$. We hide polylog$(\frac{n\kappa d}{\epsilon})$ factors for simplicity.

RGO is a natural notion of tractability for the composite sampling problem.

Logconcave finite sums. Since [536] proposed the stochastic gradient Langevin dynamics, which at each step stochastically estimates the full gradient of the function, there has been a long line of work giving bounds for this method and other similar algorithms [155, 229, 467, 61, 413]. These convergence rates depend heavily on the variance of the stochastic estimates. Inspired by variance-reduced methods in convex optimization, samplers based on low-variance estimators have also been proposed [202, 207, 80, 51, 406, 123, 554, 119]. Although our reduction-based approach is not designed specifically for solving problems of finite sum structure, our speedup can be viewed as due to a lower variance estimator implicitly defined through the oracle subproblems of Theorem 68 via repeated re-centering.

In Table 10.1, we list prior runtimes [554, 119] for sampling logconcave finite sums; note these results additionally require bounded higher derivatives (with the exception of the κ^4 dependence), obtain guarantees only in Wasserstein distance, and have polynomial dependences on ϵ^{-1}. On the other hand, our reduction-based approach obtains total variation bounds with linear dependence on κ and polylogarithmic dependence on ϵ^{-1}. Our bound also simultaneously matches the state-of-the-art bound when $n = 1$, a feature not shared by prior stochastic algorithms. To our knowledge, no previous nontrivial[7] bounds were known in the high-accuracy regime before our work.

Theoretical analyses of MALA and HMC. MALA was originally proposed in [76], and subsequently its practical and theoretical performance in different settings has received extensive treatment in the literature (cf. the survey [444]). A number of theoretical analyses related to the well-conditioned setting we study predate the work of [210], such as [457, 94], but they typically consider

[7] As mentioned previously, one can always compute the full ∇F in every iteration in a well-conditioned sampler.

more restricted settings or do not state explicit dependences on κ and d.

Recently, a line of work has obtained a sequence of stronger upper bounds on the mixing of MALA. First, [210] demonstrated that MALA achieves a mixing time of $\widetilde{O}(\kappa d + \kappa^{1.5}\sqrt{d})$ from a polynomially warm start, and the same set of authors later proved the same mixing time under an exponentially warm start (which can be explicitly constructed) in [128]. It was later demonstrated in [344] that under an appropriate averaging scheme, the mixing time could be improved to $\widetilde{O}(\kappa d)$ from an exponentially warm start with no low-order dependence. Finally, a recent work [137] demonstrated that from a polynomially warm start, MALA mixes in time $\widetilde{O}(\mathrm{poly}(\kappa)\sqrt{d})$ for general κ-conditioned distributions and in time $\widetilde{O}(\mathrm{poly}(\kappa)\sqrt[3]{d})$ for κ-conditioned Gaussians, and posed the open question of attaining similar bounds from an explicit (exponentially) warm start. This latter work was a primary motivation for our exploration.

The HMC algorithm with a leapfrog integrator (which we refer to as HMC for simplicity) can be viewed as a multi-step generalization of MALA, as it has two parameters (a step size η and a step count K), and when $K = 1$ the implementation matches MALA exactly. For larger K, the algorithm simulates the (continuous-time) Hamiltonian dynamics with respect to the potential $f(x) + \frac{1}{2}\|v\|_2^2$ where f is the target's negative log-density and v is an auxiliary "velocity" variable. The intuition is that larger K leads to more faithful discretizations of the true dynamics.

However, there are few explicit analyses of the (Metropolis-adjusted) HMC algorithm, applied to well-conditioned distributions.[8] To our knowledge, the only theoretical upper bound for the mixing of (multi-step) HMC stronger than known analyses of its one-step specialization MALA is by [128], which gave a suite of bounds trading off three problem parameters: the conditioning κ, the dimension d, and the *Hessian Lipschitz parameter* L_H, under the additional assumption that the log-density has bounded third derivatives. Assuming that L_H is polynomially bounded by the problem smoothness L, they demonstrate that HMC with an appropriate K can sometimes achieve sublinear dependence on d in number of gradient queries, where the quality of this improvement depends on κ and d (e.g. if $\kappa \in [d^{\frac{1}{3}}, d^{\frac{2}{3}}]$ and $L_H \le L^{1.5}$, $\kappa d^{\frac{11}{12}}$ gradients suffice). This prompts the question: can HMC attain query complexity *independent* of d, assuming higher derivative bounds, from an explicit warm start? Theorem 74 answers this negatively (at least in terms of relaxation time) using an exponentially-sized bad set; moreover, our hard distribution is a Gaussian, with all derivatives of order at least 3 vanishing.

Lower bounds for sampling. The bounds most closely relevant to those in this paper are given by [344], who showed that the step size of MALA must scale inversely in κ for the chain to have a constant chance of moving, and [137], who showed that the step size must scale as $d^{-\frac{1}{2}}$. Theorem 72 matches or improves both bounds simultaneously, proving that up to logarithmic factors the relaxation time of MALA scales *linearly* in both κ and d, while giving an explicit hard distribution and $\exp(-d)$-sized bad set. Moreover, both [344, 137] gave strictly spectral lower bounds, which are

[8] There has been considerably more exploration of the unadjusted variant [384, 383, 93], which typically obtain mixing guarantees scaling polynomially in the inverse accuracy (as opposed to polylogarithmic).

complemented by our Theorem 73, a mixing time lower bound.

We briefly mention several additional lower bound results in the sampling and sampling-adjacent literature, which are related to this work. Recently, [107] exhibited an information-theoretic lower bound on unadjusted discretizations simulating the *underdamped Langevin dynamics*, whose dimension dependence matches the upper bound of [480] (while leaving the precise dependence on κ open). Finally, [235] and [120] give information-theoretic lower bounds for estimating normalizing constants of well-conditioned distributions and the number of stochastic gradient queries required by first-order sampling methods under noisy gradient access respectively.

10.1.4 Technical overview: upper bounds

Composite logconcave sampling

We study the problem of approximately sampling from a distribution π on $\mathbb{R}^d$, with density

$$\frac{d\pi(x)}{dx} \propto \exp\left(-f(x) - g(x)\right). \tag{10.1}$$

Here, $f : \mathbb{R}^d \to \mathbb{R}$ is assumed to be "well-behaved" (i.e. has finite condition number), and $g : \mathbb{R}^d \to \mathbb{R}$ is a convex, but possibly non-smooth function. This problem generalizes the special case of sampling from $\exp(-f(x))$ for well-conditioned f, simply by letting g vanish. Even the specialization of (10.1) where g indicates a convex set (i.e. is 0 inside the set, and ∞ outside) is not well-understood; existing mixing time bounds for this restricted setting are large polynomials in d [96, 101], and are typically weaker than guarantees in the general logconcave setting [372, 371]. This is in contrast to the convex optimization setting, where first-order methods readily generalize to solve problem families such as $\min_{x \in \mathcal{X}} f(x)$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a convex set, as well as its generalization

$$\min_{x \in \mathbb{R}^d} f(x) + g(x), \text{ where } g : \mathbb{R}^d \to \mathbb{R} \text{ is convex and admits a proximal oracle.} \tag{10.2}$$

We defined proximal oracles in Definition 45; in short, they are prodecures which minimize the sum of a quadratic and g. Definition 45 is desirable as many natural non-smooth composite objectives arising in learning settings, such as the Lasso [505] and elastic net [555], admit efficient proximal oracles. It is clear that the definition of a proximal oracle implies it can also handle arbitrary sums of linear functions and quadratics, as the resulting function can be rewritten as the sum of a constant and a single quadratic. The seminal work [66] extends fast gradient methods to solve (10.2) via proximal oracles, and has prompted many follow-up studies.

Motivated by the success of the proximal oracle framework in convex optimization, we study sampling from the family (10.1) through the lens of RGOs, a natural extension of Definition 45. The main result of Section 10.5 is a "base" algorithm efficiently sampling from (10.1), assuming access to an RGO for g. We now survey the main components of this algorithm.

Reduction to shared minimizers. We first observe that without loss of generality, f and g share a minimizer: by shifting f and g by linear terms, i.e. $\tilde{f}(x) := f(x) - \langle \nabla f(x^*), x \rangle$, $\tilde{g}(x) := g(x) + \langle \nabla f(x^*), x \rangle$, where x^* minimizes $f + g$, first-order optimality implies both $\tilde{f}$ and $\tilde{g}$ are minimized by x^*. Moreover, implementation of a first-order oracle for $\tilde{f}$ and an RGO for $\tilde{g}$ are immediate without additional assumptions. This modification becomes crucial for our later developments, and we hope this simple observation, reminiscent of "variance reduction" techniques in stochastic optimization [300], is broadly applicable to improving algorithms for the sampling problem induced by (10.1).

Beyond Moreau envelopes: expanding the space. A typical approach in convex optimization in handling non-smooth objectives g is to instead optimize its *Moreau envelope*, defined by

$$g^{\eta}(y) := \min_{x \in \mathbb{R}^d} \left\{ g(x) + \frac{1}{2\eta} \|x - y\|_2^2 \right\}. \tag{10.3}$$

Intuitively, the envelope g^{η} trades off function value with proximity to y; a standard exercise shows that g^{η} is smooth (has a Lipschitz gradient), with smoothness depending on η, and moreover that computing gradients of g^{η} reduces to calling a proximal oracle (Definition 45). It is natural to extend this idea to the composite sampling setting, e.g. via sampling from the density

$$\exp\left(-f(x) - g^{\eta}(x)\right).$$

However, a variety of complications prevent such strategies from obtaining rates comparable to their non-composite, well-conditioned counterparts, including difficulty in bounding closeness of the resulting distribution, as well as biased drifts of the sampling process due to error in gradients.

Our approach departs from this smoothing strategy in a crucial way, inspired by Hamiltonian Monte Carlo (HMC) methods [334, 409]. HMC can be seen as a discretization of the ubiquitous Langevin dynamics, on an expanded space. In particular, discretizations of Langevin dynamics simulate the stochastic differential equation $\frac{dx_t}{dt} = -\nabla f(x_t) + \sqrt{2}\frac{dW_t}{dt}$, where W_t is Brownian motion. HMC methods instead simulate dynamics on an extended space $\mathbb{R}^d \times \mathbb{R}^d$, via an auxiliary "velocity" variable which accumulates gradient information. This is sometimes interpreted as a discretization of the underdamped Langevin dynamics [130]. HMC often has desirable stability properties, and expanding the dimension via an auxiliary variable has been used in algorithms obtaining the fastest rates in the well-conditioned logconcave sampling regime [480, 344]. Inspired by this phenomenon, we consider the density on $\mathbb{R}^d \times \mathbb{R}^d$

$$\frac{d\hat{\pi}}{dz}(z) := \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|x - y\|_2^2\right) \text{ where } z = (x, y). \tag{10.4}$$

Due to technical reasons, the family of distributions we use in our final algorithms are of slightly different form than (10.4), but this simplification is useful to build intuition. Note in particular that the form of (10.4) is directly inspired by (10.3), where rather than maximizing over x, we directly

expand the space. The idea is that for small enough η and a set on x of large measure, smoothness of f will guarantee that the marginal of (10.4) on x will concentrate y near x, a fact we make rigorous. To sample from (10.1), we then show that a rejection filter applied to a sample x from the marginal of (10.4) will terminate in constant steps. Consequently, it suffices to develop a fast sampler for (10.4).

Alternating sampling with an oracle. The form of the distribution (10.4) suggests a natural strategy for sampling from it: starting from a current state (x_k, y_k), we iterate

1. Sample $y_{k+1} \sim \exp\left(-f(y) - \frac{1}{2\eta}\|x_k - y\|_2^2\right)$.
2. Sample $x_{k+1} \sim \exp\left(-g(x) - \frac{1}{2\eta}\|x - y_{k+1}\|_2^2\right)$, via a restricted Gaussian oracle.

When f and g share a minimizer, taking a first-order approximation in the first step, i.e. sampling $y_{k+1} \sim \exp(-f(x_k) - \langle \nabla f(x_k), y - x_k\rangle - \frac{1}{2\eta}\|y - x_k\|_2^2)$, can be shown to generalize the `Leapfrog` step of HMC updates. However, for η very small (as in our setting), we observe the first step itself reduces to the case of sampling from a distribution with constant condition number, performable in $\tilde{O}(d)$ gradient calls by e.g. Metropolized HMC [210, 128, 344]. Moreover, it is not hard to see that this "alternating marginal" sampling strategy preserves the stationary distribution exactly, so no filtering is necessary. Directly bounding the conductance of this random walk, for small enough η, leads to an algorithm running in $\widetilde{O}\left(\kappa^2 d^2\right)$ iterations, each calling an RGO once, and a gradient oracle for f roughly $\widetilde{O}\left(d\right)$ times. This latter guarantee is by an appeal to known bounds [128, 344] on the mixing time in high dimensions of Metropolized HMC for a well-conditioned distribution, a property satisfied by the y-marginal of (10.4) for small η.

Stability of Gaussians under bounded perturbations. To obtain our tightest runtime result, we use that η is chosen to be much smaller than L^{-1} to show structural results about distributions of the form (10.4), yielding tighter concentration for bounded perturbations of a Gaussian (i.e. the Gaussian has covariance $\frac{1}{\eta}\mathbf{I}$, and is restricted by L-smooth f for $\eta \ll L^{-1}$). To illustrate, let

$$\frac{d\mathcal{P}_x(y)}{dy} \propto \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right)$$

and let its mean and mode be $\bar{y}_x$, y_x^*. It is standard that $\|\bar{y}_x - y_x^*\|_2 \le \sqrt{d\eta}$, by η^{-1}-strong logconcavity of $\mathcal{P}_x$. Informally, we show that for $\eta \ll L^{-1}$ and x not too far from the minimizer of f, we can improve this to $\|\bar{y}_x - y_x^*\|_2 = O(\sqrt{\eta})$; see Proposition 83 for a precise statement.

Using our structural results, we sharpen conductance bounds, improve the warmness of a starting distribution, and develop a simple rejection sampling scheme for sampling the y variable in expected constant gradient queries. Our proofs are continuous in flavor and based on gradually perturbing the Gaussian and solving a differential inequality; we believe they may of independent interest. These improvements lead to an algorithm running in $\widetilde{O}\left(\kappa^2 d\right)$ iterations; ultimately, we use our reduction framework, stated in Theorem 68, to improve this dependence to $\widetilde{O}\left(\kappa d\right)$.

Logconcave finite sums

We initiate the algorithmic study of the following task in the high-accuracy regime: sample $x \sim \pi$ within total variation distance ϵ, where $\frac{d\pi}{dx}(x) \propto \exp(-F(x))$ and

$$F(x) = \frac{1}{n} \sum_{i \in [n]} f_i(x), \tag{10.5}$$

all $f_i : \mathbb{R}^d \to \mathbb{R}$ are convex and L-smooth, and F is μ-strongly convex. We call such a distribution π a (well-conditioned) *logconcave finite sum*.

In applications (where summands correspond to points in a dataset, e.g. in Bayesian linear and logistic regression tasks [210]), querying ∇F may be prohibitively expensive, so a natural goal is to obtain bounds on the number of required queries to summands ∇f_i for $i \in [n]$. This motivation also underlies the development of stochastic gradient methods in optimization, a foundational tool in modern statistics and data processing. Naïvely, one can complete the task by using existing samplers for well-conditioned distributions and querying the full gradient ∇F in each iteration, resulting in a summand gradient query complexity of $\widetilde{O}(n\kappa d)$ [344]. Many recent works, inspired from recent developments in the complexity of optimizing a well-conditioned finite sum, have developed subsampled gradient methods for the sampling problem. However, to our knowledge, all such guarantees depend polynomially on the accuracy ϵ and are measured in the 2-Wasserstein distance; in the high-accuracy, total variation case no nontrivial query complexity is currently known.

We show in Section 10.6 that given access to the minimizer of F, a simple zeroth-order algorithm which queries $\widetilde{O}(\kappa^2 d)$ values of summands $\{f_i\}_{i \in [n]}$ succeeds (i.e. it never requires a full value or gradient query of F). The algorithm is essentially the Metropolized random walk proposed in [210] for the $n = 1$ case with a cheaper subsampled filter step. Notably, because the random walk is conducted with respect to F, we cannot efficiently query the function value at any point; nonetheless, by randomly sampling to compute a nearly-unbiased estimator of the rejection probability, we do not incur too much error. This random walk was shown in [128] to mix in $\widetilde{O}(\kappa^2 d)$ iterations; we implement each step to sufficient accuracy using $\widetilde{O}(1)$ function evaluations.

It is natural to ask if first-order information can be used to improve this query complexity, perhaps through "variance reduction" techniques (e.g. [300]) developed for stochastic optimization. The idea behind variance reduction is to recenter gradient estimates in phases, occasionally computing full gradients to improve the estimate quality. One fundamental difficulty which arises from using variance reduction in high-accuracy sampling is that the resulting algorithms are not *stateless*. By this, we mean that the variance-reduced estimates depend on the history of the algorithm, and thus it is difficult to ascertain correctness of the stationary distribution. We take a different approach to achieve a linear query dependence on the conditioning κ, described in the following section.

Proximal point reduction framework

To motivate Theorem 68, we first recast existing "proximal point" reduction-based optimization methods through the lens of composite optimization, and subsequently show that similar ideas underlying our composite sampler in Section 10.1.4 yield an analagous "proximal point reduction framework" for sampling. Chronologically, our composite sampler (originally announced in [481]) predates our reduction framework, which was then inspired by the perspective given here. We hope these insights prove fruitful for further development of proximal approaches to sampling.

Proximal point methods as composite optimization. Proximal point methods are a well-studied primitive in optimization, developed by [459]; cf. [435] for a modern survey. The principal idea is that to minimize convex $F : \mathbb{R}^d \to \mathbb{R}$, it suffices to solve a sequence of subproblems

$$x_{k+1} \leftarrow \text{argmin}_{x \in \mathbb{R}^d} \left\{ F(x) + \frac{1}{2\lambda} \|x - x_k\|_2^2 \right\}. \tag{10.6}$$

Intuitively, by tuning the parameter $\lambda \geq 0$, we trade off how regularized the subproblems (10.6) are with how rapidly the overall method converges. Smaller values of λ result in larger regularization amounts which are amenable to algorithms for minimizing well-conditioned objectives.

For optimizing functions of the form (10.5) via stochastic gradient estimates to ϵ error, [300, 168, 472] developed variance-reduced methods obtaining a query complexity of $\widetilde{O}(n + \kappa)$. To match a known lower bound of $\widetilde{O}(n + \sqrt{n\kappa})$ due to [539], two works [364, 227] appropriately applied instances of accelerated proximal point methods [257] with careful analyses of how accurately subproblems (10.6) needed to be solved. These algorithms black-box called the $\widetilde{O}(n + \kappa)$ runtime as an oracle to solve the subproblems (10.6) for an appropriate choice of λ, obtaining an accelerated rate.[9] To shed some light on this acceleration procedure, we adopt an alternative view on proximal point methods.[10] Consider the following known composite optimization result.

Proposition 42 (Informal statement of [66])**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and μ-strongly convex, and $g : \mathbb{R}^d \to \mathbb{R}$ admit a proximal oracle $\mathcal{O}(\lambda, v)$ (cf. Definition 45). There is an algorithm taking $\widetilde{O}(\sqrt{\kappa})$ iterations for $\kappa = \frac{L}{\mu}$ to find an ϵ-approximate minimizer to $f + g$, each querying ∇f and $\mathcal{O}$ a constant number of times. Further, $\lambda = \frac{1}{L}$ in all calls to $\mathcal{O}$.*

Ignoring subtleties of the error tolerance of $\mathcal{O}$, we show how to use an instance of Proposition 42 to recover the $\widetilde{O}(n + \sqrt{n\kappa})$ query complexity for optimizing (10.5). Let $f(x) = \frac{\mu}{2} \|x\|_2^2$, and $g = F - f$. For any $\Lambda \geq \mu$, f is both μ-strongly convex and Λ-smooth. Moreover, note that all calls to the proximal oracle $\mathcal{O}$ for g require solving subproblems of the form

$$\text{argmin}_{x \in \mathbb{R}^d} \left\{ F(x) - \frac{\mu}{2} \|x\|_2^2 + \frac{\Lambda}{2} \|x - v\|_2^2 \right\}. \tag{10.7}$$

[9] We note that an improved runtime without extraneous logarithmic factors was later obtained by [17].
[10] This perspective can also be found in the lecture notes [342].

The upshot of choosing a smoothness bound $\Lambda \geq \mu$ is that the regularization amount in (10.7) increases, improving the conditioning of the subproblem, which is Λ-strongly convex and $L + \Lambda$-smooth. The algorithm of e.g. [300] solves each subproblem (10.7) in $\widetilde{O}(n + \frac{L+\Lambda}{\Lambda})$ gradient queries, leading to an overall query complexity (for Proposition 42) of

$$\widetilde{O}\left(\sqrt{\frac{\Lambda}{\mu}} \cdot \left(n + \frac{L}{\Lambda}\right)\right).$$

Optimizing over $\Lambda \geq \mu$, i.e. taking $\Lambda = \max(\mu, \frac{L}{n})$, yields the desired bound of $\widetilde{O}(n + \sqrt{n\kappa})$.

Applications to sampling. In Sections 10.5 and 10.6, we develop samplers for structured families with quadratic dependence on the conditioning κ. The proximal point approach suggests a strategy for accelerating these runtimes. Namely, if there is a framework which repeatedly calls a sampler for a regularized density (analogous to calls to (10.6)), one could trade off the regularization with the rate of the outer loop. Fortunately, in the spirit of interpreting proximal point methods as composite optimization, the composite sampler of Section 10.5 itself meets these reduction framework criteria.

We briefly recall properties of our composite sampler here. Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f_{\text{wc}}(x) - f_{\text{oracle}}(x))$,[11] where f_{wc} is well-conditioned (has finite condition number κ) and f_{oracle} admits an RGO, which solves subproblems of the form

$$\mathcal{O}(\eta, v) \sim \text{the density proportional to } \exp\left(-\frac{1}{2\eta}\left\|x - v\right\|_2^2 - f_{\text{oracle}}(x)\right). \tag{10.8}$$

The algorithm of Section 10.5 only calls $\mathcal{O}$ with a fixed η; note the strong parallel between the RGO subproblem and the proximal oracle of Proposition 42. For a given value of $\eta \geq 0$, our composite sampler runs in $\widetilde{O}(\frac{1}{\eta\mu})$ iterations, each requiring a call to $\mathcal{O}$. Smaller η improve the conditioning of the negative log-density of subproblem (10.8), but increase the overall iteration count, yielding a range of trade-offs. The algorithm of Section 10.5 has an upper bound requirement on η (cf. Theorem 69); in Section 10.3, we observe that this may be lifted when $f_{\text{wc}} = 0$ uniformly, allowing for a full range of choices. Moreover, the analysis of the composite sampler becomes much simpler when $f_{\text{wc}} = 0$, as in Theorem 68. We give the framework as Algorithm 59, as well as a full (fairly short) convergence analysis. By trading off the regularization amount with the cost of implementing (10.8) via bootstrapping base samplers, we obtain a host of improved runtimes.

Beyond our specific applications, the framework we provide has strong implications as a generic reduction from mixing times scaling polynomially in κ to improved methods scaling linearly in κ. This is akin to the observation in [364] that accelerated proximal point methods generically improve $\text{poly}(\kappa)$ dependences to $\sqrt{\kappa}$ dependences for optimization. We are optimistic this insight will find further implications in the logconcave sampling literature.

[11] To disambiguate, we sometimes also use the notation $f_{\text{wc}} + f_{\text{oracle}}$ rather than $f + g$ in defining instances of our reduction framework or composite samplers, when convenient in the context.

10.1.5 Technical overview: lower bounds

In this section, we give an overview of the techniques we use to show our lower bounds. Throughout for the sake of fixing a scale, we assume the negative log-density has Hessian between $\mathbf{I}$ and $\kappa\mathbf{I}$.

MALA. Our starting point is the observation made in [137] that for a MALA step size h, the spectral gap of the MALA Markov chain scales no better than $O(h+h^2)$, witnessed by a simple one-dimensional Gaussian. Thus, our strategy for proving Theorems 71 and 72 is to show a dichotomy on the choice of step size: either h is so large such that we can construct an $\exp(d)$-warm start where the chain is extremely unlikely to move (e.g. the step almost always is filtered), or it is small enough to imply a poor spectral gap. In the Gaussian case, we achieve this by explicitly characterizing the rejection probability and demonstrating that choosing the "small ball" warm start where $\|x\|_2^2$ is smaller than its expectation by a constant ratio suffices to upper bound h.

Given the result of Theorem 71, we see that if MALA is to move at all with decent probability from an exponentially warm start, we must take $h \ll 1$, so the spectral gap in this regime is simply $O(h)$. We now move onto the more general well-conditioned setting. As a thought experiment, we note that the upper bound analyses of [210, 128, 344] for MALA have a dimension dependence which is bottlenecked by the *noise term* only. In particular, the MALA iterates apply a filter to the move $x' \leftarrow x - h\nabla f(x) + \sqrt{2h}g$, where $g \sim \mathcal{N}(0, \mathbf{I})$ is a standard Gaussian vector. However, even for the more basic "Metropolized random walk" where the proposal is simply $x' \leftarrow x + \sqrt{2h}g$, the dimension dependence of upper bound analyses scales linearly in d. Thus, it is natural to study the effect of the noise, and construct a hard distribution based around it.

We first formalize this intuition, and demonstrate that for step sizes not ruled out by Theorem 71, all terms in the rejection probability calculation other than those due to the effect of the noise g are low-order. Moreover, because the effect of the noise is coordinatewise separable (since $\mathcal{N}(0, \mathbf{I})$ is a product distribution), to demonstrate a $\widetilde{O}(\frac{1}{\kappa d})$ upper bound on h it suffices to show a hard one-dimensional distribution where the log-rejection probability has expectation $-\Omega(h\kappa)$, and apply sub-Gaussian concentration to show a product distribution has expectation $-\Omega(h\kappa d)$.

At this point, we reduce to the following self-contained problem: let $x \in \mathbb{R}$, let $\pi^* \propto \exp(-f_{1\mathrm{d}})$ be one-dimensional with second derivative $\le \kappa$, and let $x_g = x + \sqrt{2h}g$ for $g \sim \mathcal{N}(0, 1)$. We wish to construct $f_{1\mathrm{d}}$ such that for x in a constant probability region over $\exp(-f_{1\mathrm{d}})$ (the "bad set"),

$$\mathbb{E}_{g\sim\mathcal{N}(0,1)}\left[-f_{1\mathrm{d}}(x_g) + f_{1\mathrm{d}}(x) - \frac{1}{2}\left\langle x - x_g, f'_{1\mathrm{d}}(x) + f'_{1\mathrm{d}}(x_g)\right\rangle\right] = -\Omega(h\kappa), \tag{10.9}$$

where the contents of the expectation in (10.9) are the log-rejection probability along one coordinate by a straightforward calculation. By forming a product distribution using $f_{1\mathrm{d}}$ as a building block, and combining with the remaining low-order terms due to the drift $\nabla f(x)$, we attain an $\exp(-d)$-sized region where the rejection probability is $\exp(-\Omega(h\kappa d))$, completing Theorem 72.

It remains to construct such a hard $f_{1\mathrm{d}}$. The calculation

$$-f_{1\mathrm{d}}(x_g) + f_{1\mathrm{d}}(x) - \frac{1}{2}\langle x - x_g, f'_{1\mathrm{d}}(x) + f'_{1\mathrm{d}}(x_g)\rangle = -2h\int_0^1 \left(\frac{1}{2} - s\right) g^2 f''_{1\mathrm{d}}(x + s(x_g - s))ds$$

suggests the following approach: because the above integral places more mass closer to the starting point, we wish to make sure our bad set has large second derivative, but most moves g result in a much smaller second derivative. Our construction patterns this intuition: we choose[12]

$$f_{1\mathrm{d}}(x) = \frac{\kappa}{3}x^2 - \frac{\kappa h}{3}\cos\frac{x}{\sqrt{h}} \implies f''_{1\mathrm{d}}(x) = \frac{2\kappa}{3} + \frac{\kappa}{3}\cos\frac{x}{\sqrt{h}},$$

such that our bad set is when $\cos\frac{x}{\sqrt{h}}$ is relatively large (which occurs with probability $\to \frac{1}{2}$ for small h in one dimension). The period of our construction scales with $\sqrt{h}$, so that most moves $\sqrt{2hg}$ of size $O(\sqrt{h})$ will "skip a period" and hence hit a region with small second derivative, satisfying (10.9).

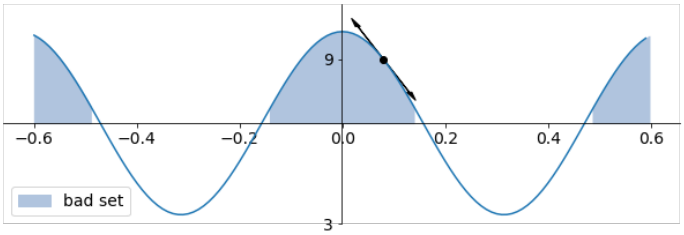


Figure 10.1: Second derivative of our hard function $f_{1\mathrm{d}}$, $\kappa = 10$, $h = 0.01$. Starting from inside the hard region, on average over $g \sim \mathcal{N}(0, \mathbf{I})$, a move by $\sqrt{2hg}$ decreases the second derivative.

HMC. We further demonstrate that similar hard Gaussians as the one we use for MALA also place an upper bound on the step size of HMC for any number of steps K. Our starting point is a novel characterization of HMC iterates on Gaussians: namely, when the negative log-density is quadratic, we show that the HMC iterates implement a linear combination between the starting position and velocity, where the coefficients are given by *Chebyshev polynomials*. For step size η of size $\Omega(\frac{1}{K\sqrt{\kappa}})$ for specific constants, we show the HMC chain begins to cycle because of the locations of the Chebyshev polynomials' zeroes, and cannot move. Moreover, for sufficiently small step size η outside of this range, it is straightforward by examining the coefficients of Chebyshev polynomials to show that they are the same (up to constant factors) as in the MALA case, at which point our previous lower bound holds. It takes some care to modify our hard Gaussian construction to rule out all constant ranges in the $\eta \approx \frac{1}{K\sqrt{\kappa}}$ region, but by doing so we obtain Theorem 74.

We remark that the observation that HMC iterates are implicitly implementing a Chebyshev polynomial approximation appears to be unknown in the literature, and is a novel contribution of

[12] We note [137] also used a (different, but similar) cosine-based construction for their lower bound.

our work. We believe understanding this connection is a worthwhile endeavor, as a similar connection between polynomial approximation and first-order convex optimization has led to various interesting interpretations of Nesterov's accelerated gradient descent method [266, 49].

10.1.6 Roadmap

We give notations and preliminaries in Section 10.2. In Section 10.3 we give our framework for bootstrapping fast regularized samplers, and prove its correctness (Theorem 68). Assuming the "base samplers" of Theorems 69 and 70, in Section 10.4 we apply our reduction to obtain all of our strongest guarantees, namely Corollaries 45, 46, and 47. We then prove Theorems 69 and 70 in Sections 10.5 and 10.6. We present our lower bound constructions (proofs of Theorems 71, 72, 73, and 74 respectively) in Sections 10.7, 10.8, 10.9, and 10.10. Finally, we give concluding thoughts on our lower bound results in Section 10.11.

10.2 Preliminaries

10.2.1 Notation

General notation. For $d \in \mathbb{N}$, $[d]$ refers to the set of naturals $1 \le i \le d$; $\|\cdot\|_2$ is the Euclidean norm on $\mathbb{R}^d$ when d is clear from context. $\mathcal{N}(\mu, \mathbf{\Sigma})$ is the multivariate Gaussian of specified mean and variance, $\mathbf{I}$ is the identity of appropriate dimension when clear from context, and $\preceq$ is the Loewner order on symmetric matrices. For any positive semidefinite matrix $\mathbf{A}$, we let $\|\cdot\|_{\mathbf{A}}$ be its induced seminorm $\|x\|_{\mathbf{A}} = \sqrt{x^\top \mathbf{A} x}$. We let $\{W_t\}_{t \ge 0} \subset \mathbb{R}^d$ denote the standard Brownian motion when dimensions are clear from context.

Functions. We say twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex if $\mu \mathbf{I} \preceq \nabla^2 f(x) \preceq L\mathbf{I}$ for all $x \in \mathbb{R}^d$; it is well-known that L-smoothness implies that f has an L-Lipschitz gradient, and that for any $x, y \in \mathbb{R}^d$,

$$f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|_2^2 \le f(y) \le f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|_2^2 .$$

If f is L-smooth and μ-strongly convex, we say it has a condition number $\kappa := \frac{L}{\mu}$. We call a zeroth-order oracle, or "value oracle", an oracle which returns $f(x)$ on any input point $x \in \mathbb{R}^d$; similarly, a first-order oracle, or "gradient oracle", returns both the value and gradient $(f(x), \nabla f(x))$.

For our lower bound results, we explicitly assume that κ is at least a constant for convenience of stating bounds; a lower bound of 10 suffices for all our results.

Distributions. We call distribution π on $\mathbb{R}^d$ logconcave if $\frac{d\pi}{dx}(x) = \exp(-f(x))$ for convex f; π is μ-strongly logconcave if f is μ-strongly convex. For $A \subseteq \mathbb{R}^d$, A^c is its complement, and we let $\pi(A) := \int_{x \in A} d\pi(x)$. We say distribution ρ is β-warm with respect to π if $\frac{d\pi}{d\rho}(x) \le \beta$ everywhere, and define the total variation $\|\pi - \rho\|_{\mathrm{TV}} := \sup_{A \subseteq \mathbb{R}^d} \pi(A) - \rho(A)$. We will frequently use the fact

that $\|\pi - \rho\|_{\mathrm{TV}}$ is also the probability that $x \sim \pi$ and $x' \sim \rho$ are unequal under the best coupling of (π, ρ); this allows us to "locally share randomness" when comparing two random walk procedures. We define the expectation $\mathbb{E}_\pi$ and variance Var_π with respect to distribution π in the standard way,

$$\mathbb{E}_\pi[h(x)] := \int h(x) d\pi(x), \ \mathrm{Var}_\pi[h(x)] := \mathbb{E}_\pi\left[(h(x))^2\right] - \left(\mathbb{E}_\pi[h(x)]\right)^2.$$

Structured distributions. This work considers two types of distributions with additional structure, which we call *composite logconcave densities* and *logconcave finite sums*. A composite logconcave density has the form $\exp(-f(x) - g(x))$, where both f and g are convex. In this context throughout, f will either be uniformly 0 or have a finite condition number (be "well-conditioned"), and g will represent a "simple" but possibly non-smooth function, as measured by admitting an RGO (cf. Definition 43). We will sometimes refer to the components as f and g as f_{wc} and f_{oracle} respectively, to disambiguate when the functions f and g are already defined in context. In our reduction-based approaches, we have additional structure on the parameter λ which an RGO is called with (cf. Definition 44). Specifically, in our instances typically $\lambda^{-1} \gg L$ (or some other "niceness" parameter associated with the negative log-density); this can be seen as heavily regularizing the negative log-density, and often makes the implementation simpler.

Finally, a logconcave finite sum has density of the form $\exp(-F(x))$ where $F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x)$. When treating such densities, we make the assumption that each constituent summand f_i is L-smooth and convex, and the overall function F is μ-strongly convex. We measure complexity of algorithms for logconcave finite sums by gradient queries to summands, i.e. $\nabla f_i(x)$ for some $i \in [n]$.

Optimization. Throughout this work, we are somewhat liberal with assuming access to minimizers to various functions (namely, the negative log-densities of target distributions). We give a more thorough discussion of this assumption in Appendix I.1, but note here that for all function families we consider (well-conditioned, composite, and finite sum), efficient first-order methods exist for obtaining high accuracy minimizers, and this optimization query complexity is never the leading-order term in any of our algorithms assuming polynomially bounded initial error.

Sampling. Consider a Markov chain defined on $\mathbb{R}^d$ with transition kernel $\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}$, so that $\int \mathcal{T}_x(y)dy = 1$ for all x. Further, denote the stationary distribution of the Markov chain by π^*. Define the Dirichlet form of functions $g, h : \mathbb{R}^d \to \mathbb{R}$ with respect to the Markov chain by

$$\mathcal{E}(g,h) := \int g(x)h(x)d\pi^*(x) - \iint g(y)h(x)\mathcal{T}_x(y)d\pi^*(x)dy.$$

A standard calculation demonstrates that

$$\mathcal{E}(g,g) = \frac{1}{2}\iint (g(x) - g(y))^2 \mathcal{T}_x(y)d\pi^*(x)dy.$$

The mixing of the chain is governed by its spectral gap, a classical quantity we now define:

$$\lambda\left(\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}\right) := \inf_g \left\{\frac{\mathcal{E}(g,g)}{\mathrm{Var}_{\pi^*}[g]}\right\}. \tag{10.10}$$

The relaxation time is the inverse spectral gap. We also recall a result of Cheeger [122], showing the spectral gap is $O(\Phi)$, where Φ is the conductance of the chain:

$$\Phi\left(\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}\right) := \inf_{A\subset\mathbb{R}^d | \pi^*(A)\leq\frac{1}{2}} \frac{\int_{x\in A}\mathcal{T}_x(A^c)d\pi^*(x)}{\pi^*(A)} \tag{10.11}$$

Finally, we recall the definition of a Metropolis filter. A Markov chain with transitions $\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}$ and stationary distribution π^* is said to be reversible if for all $x, y \in \mathbb{R}^d$,

$$d\pi^*(x)\mathcal{T}_x(y) = d\pi^*(y)\mathcal{T}_y(x).$$

The Metropolis filter is a way of taking an arbitrary set of proposal distributions $\{\mathcal{P}_x\}_{x\in\mathbb{R}^d}$ and defining a reversible Markov chain with stationary distribution π^*. In particular, the Markov chain induced by the Metropolis filter has transition distributions $\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}$ defined by

$$\mathcal{T}_x(y) := \mathcal{P}_x(y)\min\left(1, \frac{d\pi^*(y)\mathcal{P}_y(x)}{d\pi^*(x)\mathcal{P}_x(y)}\right) \text{ for all } y \neq x. \tag{10.12}$$

Whenever the proposal is rejected by the modified distributions above, the chain does not move.

10.2.2 Technical facts

We will repeatedly use the following results.

Fact 27 (Gaussian integral)**.** *For any $\lambda \geq 0$ and $v \in \mathbb{R}^d$,*

$$\int \exp\left(-\frac{1}{2\lambda}\|x-v\|_2^2\right)dx = (2\pi\lambda)^{\frac{d}{2}}.$$

Fact 28 ([210], Lemma 1)**.** *Let π be a μ-strongly logconcave distribution, and let x^* minimize its negative log-density. Then, for $x \sim \pi$ and any $\delta \in [0,1]$, with probability at least $1-\delta$,*

$$\|x - x^*\|_2 \leq \sqrt{\frac{d}{\mu}}\left(2 + 2\max\left(\sqrt[4]{\frac{\log(1/\delta)}{d}}, \sqrt{\frac{\log(1/\delta)}{d}}\right)\right).$$

Fact 29 ([267], Theorem 1.1)**.** *Let π be a μ-strongly logconcave density. Let $d\gamma_\mu(x)$ be the Gaussian density with covariance matrix $\mu^{-1}\mathbf{I}$. For any convex function h,*

$$\mathbb{E}_\pi[h(x - \mathbb{E}_\pi[x])] \leq \mathbb{E}_{\gamma_\mu}[h(x - \mathbb{E}_{\gamma_\mu}[x])].$$

Fact 30 ([206], Theorem 1)**.** *Let π be a μ-strongly logconcave distribution, and let x^* minimize its negative log-density. Then, $\mathbb{E}_\pi[\|x - x^*\|_2^2] \le \frac{d}{\mu}$.*

Fact 31 (Mill's inequality)**.** *For one-dimensional Gaussian random variable $Z \sim \mathcal{N}(0, \sigma^2)$,*

$$\Pr[Z > t] \le \sqrt{\frac{2}{\pi}} \frac{\sigma}{t} \exp\left(-\frac{t^2}{2\sigma^2}\right).$$

Fact 32 (χ^2 tail bounds, Lemma 1 [340])**.** *Let $\{Z_i\}_{i\in[n]} \sim_{\text{i.i.d.}} \mathcal{N}(0,1)$ and $a \in \mathbb{R}^n_{\ge 0}$. Then*

$$\Pr\left[\sum_{i\in[n]} a_i Z_i^2 - \|a\|_2^2 \ge 2\|a\|_2 \sqrt{t} + 2\|a\|_\infty t\right] \le \exp(-t),$$
$$\Pr\left[\sum_{i\in[n]} a_i Z_i^2 - \|a\|_2^2 \le -2\|a\|_2 \sqrt{t}\right] \le \exp(-t).$$

Fact 33 (Bernstein's inequality)**.** *Let $\{Z_i\}_{i\in[n]}$ be independent mean-zero random variables with sub-exponential parameter λ. Then*

$$\Pr\left[\left|\sum_{i\in[n]} Z_i\right| > t\right] \le \exp\left(-\frac{1}{2}\min\left(\frac{t^2}{n\lambda^2}, \frac{t}{\lambda}\right)\right).$$

10.2.3 Metropolis-adjusted Langevin algorithm

In this section, we formally define the Metropolis-adjusted Langevin algorithm (MALA) which we study in Sections 10.7 and 10.8. Throughout this discussion, fix a distribution π on $\mathbb{R}^d$, with density $\frac{d\pi}{dx}(x) = \exp(-f(x))$, and suppose that f is twice-differentiable for simplicity.

The MALA Markov chain is given by a discretization of the (continuous-time) Langevin dynamics

$$dx_t = -\nabla f(x_t)dt + \sqrt{2}dW_t,$$

which is well-known to have stationary density $\exp(-f(x))$. MALA is defined by performing a simple Euler discretization of the Langevin dynamics up to time $h > 0$, and then applying a Metropolis filter. In particular, define the proposal distribution at a point x by

$$\mathcal{P}_x := \mathcal{N}(x - h\nabla f(x), 2h\mathbf{I}).$$

We obtain the MALA transition distribution by applying the definition (10.12), which yields

$$\mathcal{T}_x(y) \propto \exp\left(-\frac{\|y - (x - h\nabla f(x))\|_2^2}{4h}\right) \min\left(1, \frac{\exp\left(-f(y) - \frac{\|x-(y-h\nabla f(y))\|_2^2}{4h}\right)}{\exp\left(-f(x) - \frac{\|y-(x-h\nabla f(x))\|_2^2}{4h}\right)}\right). \tag{10.13}$$

The normalization constant above is that of the multivariate Gaussian with covariance $2h\mathbf{I}$.

10.2.4 Hamiltonian Monte Carlo

In this section, we formally define the (Metropolized) Hamiltonian Monte Carlo (HMC) method which we study in Section 10.10. We assume the same setting as Section 10.2.3.

The Metropolized HMC algorithm is governed by two parameters, a step size $\eta > 0$ and a step count $K \in \mathbb{N}$, and can be viewed as a multi-step generalization of MALA. In particular, when $K = 1$ it is straightforward to show that HMC is a reparameterization of MALA, see e.g. Appendix I.5. More generally, from an iterate x, HMC performs the following updates.

1. $x_0 \leftarrow x$, $v_0 \sim \mathcal{N}(0, \mathbf{I})$
2. For $0 \le k < K$:
   (a) $v_{k+\frac{1}{2}} \leftarrow v_k - \frac{\eta}{2}\nabla f(x_k)$
   (b) $x_{k+1} \leftarrow x_k + \eta v_{k+\frac{1}{2}}$
   (c) $v_{k+1} \leftarrow v_k - \frac{\eta}{2}\nabla f(x_{k+1})$
3. Return x_K

Each loop of step 2 is known in the literature as a "leapfrog" step, and is a discretization of Hamilton's equations for the Hamiltonian function $\mathcal{H}(x, v) := f(x) + \frac{1}{2}\|v\|_2^2$; for additional background, we refer the reader to [128]. This discretization is well-known to have reversible transition probabilities (i.e. the transition density is the same if the endpoints are swapped) because it satisfies a property known as *symplecticity*. Moreover, the Markov chain has stationary density on the expanded space $(x, v) \in \mathbb{R}^d \times \mathbb{R}^d$ proportional to $\exp(-\mathcal{H}(x, v))$. Correspondingly, the Metropolized HMC Markov chain performs the above algorithm from a point x, and accepts with probability

$$\min\left\{1, \frac{\exp\left(-\mathcal{H}(x_K, v_K)\right)}{\exp\left(-\mathcal{H}(x_0, v_0)\right)}\right\}. \tag{10.14}$$

10.3 Proximal reduction framework

The reduction framework of Theorem 68 can be thought of as a specialization of a more general composite sampler which we develop in Section 10.5, whose guarantees are reproduced here.

Theorem 69. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp\left(-f(x) - g(x)\right)$ such that f is L-smooth and μ-strongly convex, and let $\epsilon \in (0, 1)$. Let $\eta \le \frac{1}{32L\kappa d\log(\kappa/\epsilon)}$ (where $\kappa = \frac{L}{\mu}$), $T = \Theta(\frac{1}{\eta\mu}\log(\frac{\kappa d}{\epsilon}))$, and let $\mathcal{O}$ be a η-RGO for g. Further, assume access to the minimizer $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d}\{f(x) + g(x)\}$. There is an algorithm which runs in T iterations in expectation, each querying a gradient oracle of f and $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

Our main observation, elaborated on more formally for specific applications in Section 10.4, is that a variety of structured logconcave densities have negative log-densities f_{oracle}, where we can implement an efficient restricted Gaussian oracle for f_{oracle} via calling an existing sampling method. Crucially, in these instantiations we use the fact that the distributions which $\mathcal{O}$ is required to sampled from are heavily regularized (restricted by a quadratic with large leading coefficient) to obtain fast samplers. We further note that the upper bound requirement on η in Theorem 69 can be lifted when the "well-conditioned" component is uniformly 0. Instead of setting $f = 0$ and $g = f_{\text{oracle}}$ in Theorem 69, and refining the analysis for this special case to tolerate arbitrary η, we provide a self-contained proof here. This particular structure (the composite setting where f_{wc} is uniformly zero and f_{oracle} is strongly convex) admits significant simplifications from the more general case, so using slightly different proof techniques, we are able to obtain stronger convergence guarantees for this particular problem allowing for mixing in fewer than d iterations from a feasible start.

Theorem 68. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f_{\text{oracle}}(x))$ such that f_{oracle} is μ-strongly convex, and let $\epsilon \in (0,1)$. Let $\eta \leq \frac{1}{\mu}$, $T = \Theta(\frac{1}{\eta\mu}\log\frac{d}{\eta\mu\epsilon})$ for some $\beta \geq 1$, and $\mathcal{O}$ be a η-RGO for f_{oracle}. Algorithm 59, initialized at the minimizer of f_{oracle}, runs in T iterations, each querying $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

For simplicity of notation, we replace f_{oracle} in the statement of Theorem 68 with g throughout just this section. Let π be a density on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-g(x))$ where g is μ-strongly convex (but possibly non-smooth), and let $\mathcal{O}$ be a restricted Gaussian oracle for g. Consider the joint distribution $\hat{\pi}$ supported on an expanded space $z = (x, y) \in \mathbb{R}^d \times \mathbb{R}^d$ with density, for some $\eta > 0$,

$$\frac{d\hat{\pi}}{dz}(z) \propto \exp\left(-g(x) - \frac{1}{2\eta}\|x - y\|_2^2\right).$$

Note that the x-marginal of $\hat{\pi}$ is precisely π, so it suffices to sample from the x-marginal. We consider a simple alternating Markov chain for sampling from $\hat{\pi}$, described in the following Algorithm 59.

Algorithm 59: `AlternateSample`(g, η, T)

Input: μ-strongly convex $g : \mathbb{R}^d \to \mathbb{R}$, $\eta > 0$, $T \in \mathbb{N}$, $x_0 = \min_x g(x)$;
for $k \in [T]$ **do**
  Sample $y_k \sim \pi_{x_{k-1}}$, where for all $x \in \mathbb{R}^d$, $\frac{d\pi_x}{dy}(y) \propto \exp\left(-\frac{1}{2\eta}\|x - y\|_2^2\right)$;
  Sample $x_k \sim \pi_{y_k}$, where for all $y \in \mathbb{R}^d$, $\frac{d\pi_y}{dx}(x) \propto \exp\left(-g(x) - \frac{1}{2\eta}\|x - y\|_2^2\right)$;
Return: x_T;

By observing that the distributions π_x and π_y in the above method are precisely the marginal distributions of $\hat{\pi}$ with one variable fixed, it is straightforward to see that $\hat{\pi}$ is a stationary distribution of the process. We make this formal in the following lemma.

Lemma 181 (Alternating marginal sampling)**.** *Let $\hat{\pi}$ be a density on two blocks (x, y). Sample $(x, y) \sim \hat{\pi}$, and then sample $\tilde{x} \sim \hat{\pi}(\cdot, y)$, $\tilde{y} \sim \hat{\pi}(\tilde{x}, \cdot)$. Then, the distribution of $(\tilde{x}, \tilde{y})$ is $\hat{\pi}$. Moreover, the alternating marginal sampling Markov chain on either marginal is reversible.*

Proof. The density of the resulting distribution at $(\tilde{x}, y)$ is proportional to the product of the (marginal) density at y and the conditional distribution of $\tilde{x} \mid y$, which by definition is $\hat{\pi}$. Therefore, $(\tilde{x}, y)$ is distributed as $\hat{\pi}$, and the argument for $\tilde{y}$ follows symmetrically. To see reversibility on the x marginal, it suffices to note that the probability we move from x to x' is proportional to

$$\int_y \hat{\pi}(x, y)\hat{\pi}(x', y)dy,$$

which is a symmetric function of x and x'. A similar argument holds for the y marginals. □

We also state a simple observation about alternating schemes such as Algorithm 59, which will be useful later. Let $\mathcal{P}_x$ be the density of y_k after one step of the above procedure starting from $x_{k-1} = x$, and let $\mathcal{T}_x$ be the resulting density of x_k.

Observation 1. *For any two points x, $x' \in \mathbb{R}^d$, $\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\mathrm{TV}} \le \|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}}$.*

Proof. This follows by the coupling characterization of total variation (see e.g. Chapter 5 of [354]). Per the optimal coupling of $y \sim \mathcal{P}_x$ and $y' \sim \mathcal{P}_{x'}$, whenever the total variation sets $y = y'$ in Line 2 of `AlternateSample`, we can couple the resulting distributions in Line 3 as well. □

In order to prove Theorem 45, we first show that the random walk in Algorithm 59 converges rapidly in the 2-Wasserstein distance (denoted W_2 in this section).

Lemma 182. *Let π_0 be the starting distribution of x in Algorithm 59. Let π_k be the distribution of x_k and π be the x-marginal of $\hat{\pi}$. For all $k \ge 0$,*

$$W_2^2(\pi_{k+1}, \pi) \le \frac{1}{(1 + \eta\mu)^2} W_2^2(\pi_k, \pi).$$

Hence, for any $\eta \le \frac{1}{\mu}$, in $T' = O\left(\frac{1}{\eta\mu} \log \frac{d}{\mu\Delta}\right)$ iterations, the random walk mixes to

$$W_2(\pi_{T'}, \pi) \le \Delta.$$

Proof. Let Γ_{x_k} be the optimal coupling between $x_k \sim \pi_k$ and $\hat{x} \sim \pi$ according to the W_2 distance. Coupling the Gaussian random variable generating $y_{k+1} \sim \pi_{x_k}$ and $\hat{y} \sim \pi_{\hat{x}}$ gives a coupling $\Gamma_{y_{k+1}}$ between y_{k+1} and $\hat{y}$ such that

$$\mathbb{E}_{\Gamma_{y_{k+1}}}\left[\|y_{k+1} - \hat{y}\|_2^2\right] = \mathbb{E}_{\Gamma_{x_k}}\left[\|x_k - \hat{x}\|_2^2\right]. \tag{10.15}$$

Then, let π_y be the distribution of x_{k+1} in a run of Line 3 of Algorithm 59 starting from $y_{k+1} = y$, and $\pi_{\hat{y}}$ be the distribution of $\hat{x}$ in Line 3 starting from $\hat{y}$, respectively. Since $\pi_{\hat{y}}$ is $\mu + \frac{1}{\eta}$ strongly log-concave, $\pi_{\hat{y}}$ satisfies a log-Sobolev inequality with constant $\mu + \frac{1}{\eta}$ (Theorem 2 of [430]). Hence,

$$\begin{aligned} W_2^2(\pi_y, \pi_{\hat{y}}) &\le \frac{2}{\mu + \frac{1}{\eta}} d_{\mathrm{KL}}(\pi_y \| \pi_{\hat{y}}) \\ &\le \frac{1}{\left(\mu + \frac{1}{\eta}\right)^2} \mathbb{E}_{\pi_y}\left[\left\| \nabla \log \frac{\pi_y}{\pi_{\hat{y}}} \right\|_2^2\right] \\ &\le \frac{1}{(1+\eta\mu)^2} \left\|y - \hat{y}\right\|_2^2. \end{aligned}$$

The first step used the Talagrand transportation inequality (Theorem 1 of [430]). The second step used the log-Sobolev inequality. The third step used

$$\begin{aligned} \nabla \log \frac{\pi_y(x)}{\pi_{\hat{y}}(x)} &= \nabla \log \frac{\exp(-g(x) - \frac{1}{2\eta}\left\|x - y\right\|_2^2) \int_{x'} \exp(-g(x) - \frac{1}{2\eta}\left\|x - \hat{y}\right\|_2^2) dx'}{\exp(-g(x) - \frac{1}{2\eta}\left\|x - \hat{y}\right\|_2^2) \int_{x'} \exp(-g(x) - \frac{1}{2\eta}\left\|x - y\right\|_2^2) dx'} \\ &= \frac{1}{2\eta} \nabla \left(\left\|x - \hat{y}\right\|_2^2 - \left\|x - y\right\|_2^2\right) = \frac{1}{\eta}(y - \hat{y}). \end{aligned} \tag{10.16}$$

Taking expectation over $\Gamma_{y_{k+1}}$ and using (10.15) shows that

$$W_2^2(\pi_{k+1}, \pi) \le \frac{1}{(1+\eta\mu)^2} W_2^2(\pi_k, \pi).$$

Algorithm 59 starts from the distribution $\pi_0 = \delta_{x^*}$, where $x^* = \min_x g(x)$. Since π is μ-strongly logconcave, we have (see e.g. Proposition 1 of [206])

$$W_2^2(\pi_0, \pi) = \mathbb{E}_{\hat{\pi}}\left[\left\|x^* - x\right\|^2\right] \le \frac{d}{\mu}.$$

Then, for $\eta < \frac{1}{\mu}$, $\frac{1}{1+\eta\mu} \le 1 - \frac{\eta\mu}{2}$, so after $T' = O(\frac{1}{\eta\mu} \log \frac{d}{\mu\Delta})$ iterations, $W_2(\pi_{T'}, \pi) \le \Delta$. □

Next, we bound the KL divergence between the output of Algorithm 59 and the target distribution π. We need the following standard lemma regarding KL divergences of marginal distributions.

Lemma 183. *Let P_z and Q_z be distributions supported on $\mathcal{X}$ indexed by z, a random variable distributed as π_z. Let $\widetilde{P}$ be the joint distribution of (x, z) for $x \sim P_z$ and $z \sim \pi_z$, and $\widetilde{Q}$ be the joint distribution of (x, z) as $x \sim Q_z$ and $z \sim \pi_z$. Let P and Q be the marginal distribution of $\widetilde{P}$ and $\widetilde{Q}$ on x, averaged over z. Then,*

$$d_{\mathrm{KL}}(P\|Q) \le \mathbb{E}_{z\sim\pi_z}\left[d_{\mathrm{KL}}(P_z\|Q_z)\right].$$

Proof. By the definition of d_{KL},

$$\begin{aligned} d_{\mathrm{KL}}(\widetilde{P}\|\widetilde{Q}) &= \mathbb{E}_{(x,z)\sim\widetilde{P}}\left[\log\frac{\widetilde{P}(x,z)}{\widetilde{Q}(x,z)}\right] \\ &= \mathbb{E}_{z\sim\pi_z}\left[\mathbb{E}_{x\sim P_z}\left[\log\frac{\widetilde{P}(x,z)}{\widetilde{Q}(x,z)}\right]\right] \\ &= \mathbb{E}_{z\sim\pi_z}\left[\mathbb{E}_{x\sim P_z}\left[\log\frac{P_z(x)}{Q_z(x)}\right]\right] \\ &= \mathbb{E}_{z\sim\pi_z}\left[d_{\mathrm{KL}}(P_z\|Q_z)\right]. \end{aligned}$$

Finally, by the data processing inequality,

$$d_{\mathrm{KL}}(P\|Q) \le d_{\mathrm{KL}}(\widetilde{P}\|\widetilde{Q}) = \mathbb{E}_{z\sim\pi_z}\left[d_{\mathrm{KL}}(P_z\|Q_z)\right].$$

□

The following lemma shows that a 2-Wasserstein distance bound on the distribution at iteration k implies a KL divergence bound on iteration $k+1$.

Lemma 184. *Let π_k be the distribution of x_k for some k such that $W_2(\pi_k,\pi)\le\Delta$ and π be the x-marginal of $\hat{\pi}$. Then,*

$$d_{\mathrm{KL}}(\pi_{k+1}\|\pi) \le \frac{\Delta^2}{2\eta}.$$

Proof. As in Lemma 182, let Γ_{x_k} be the optimal coupling between $x_k\sim\pi_k$ and $\hat{x}\sim\pi$, which yields a coupling $\Gamma_{y_{k+1}}$ between y_{k+1} and $\hat{y}$ such that

$$\mathbb{E}_{\Gamma_{y_{k+1}}}\left[\|y_{k+1}-\hat{y}\|_2^2\right] = \mathbb{E}_{\Gamma_{x_k}}\left[\|x_k-\hat{x}\|_2^2\right] \le \Delta^2. \tag{10.17}$$

Then,

$$\begin{aligned} d_{\mathrm{KL}}(\pi_{k+1}\|\pi) &\le \mathbb{E}_{(y_{k+1},\hat{y})\sim\Gamma_{y_k}}\left[d_{\mathrm{KL}}(\pi_{y_{k+1}}\|\pi_{\hat{y}})\right] \\ &\le \frac{1}{2\eta^2\left(\mu+\frac{1}{\eta}\right)}\mathbb{E}_{(y_{k+1},\hat{y})\sim\Gamma_{y_{k+1}}}\left[\|y_{k+1}-\hat{y}\|_2^2\right] \le \frac{\Delta^2}{2\eta}. \end{aligned}$$

The first inequality followed from Lemma 184 by taking $P=\pi_{k+1}$, $Q=\pi$ and $z=(y_{k+1},y)$. The second inequality used the log-Sobolev inequality and (10.16). The last inequality used (10.17). □

Finally, putting the pieces together, Theorem 68 follows from Lemma 182 and Lemma 184.

Proof of Theorem 68. By Lemma 182 and Lemma 184, there is $T = O\left(\frac{1}{\eta\mu}\log\frac{d}{\eta\mu\epsilon}\right)$ so that $d_{\mathrm{KL}}(\pi_T\|\pi) \le 2\epsilon^2$. By Pinsker's inequality,

$$\|\pi_T - \pi\|_{\mathrm{TV}} \le \sqrt{\frac{1}{2}d_{\mathrm{KL}}(\pi_T\|\pi)} = \epsilon.$$

□

We note that Theorem 68 is robust to a small amount of error tolerance in the sampler $\mathcal{O}$. Specifically, if $\mathcal{O}$ has tolerance $\frac{\epsilon}{2T}$, then by calling Theorem 68 with desired accuracy $\frac{\epsilon}{2}$ and adjusting constants appropriately, the cumulative error incurred by all calls to $\mathcal{O}$ is within the total requisite bound (formally, this can be shown via the coupling characterization of total variation). We defer a more formal elaboration on this inexactness argument to Appendix I.1 and the proof of Proposition 46.

10.4 Tighter runtimes for structured densities

In this section, we use applications of Theorem 68 to obtain simple analyses of novel state-of-the-art high-accuracy runtimes for the well-conditioned densities studied in [210, 128, 344], as well as the composite and finite sum densities studied in this work. We will assume the conclusions of Theorems 69 and 70 respectively in deriving the results of Sections 10.4.2 and 10.4.3.

10.4.1 Well-conditioned logconcave sampling: proof of Corollary 45

In this section, let π be a distribution on $\mathbb{R}^d$ with density proportional to $\exp(-f(x))$, where f is L-smooth and μ-strongly convex (and $\kappa = \frac{L}{\mu}$) and has pre-computed minimizer x^*. We will instantiate Theorem 68 with $f_{\mathrm{oracle}}(x) = f(x)$, and choose $\eta = \frac{1}{8Ld\log(\kappa)}$. We now require an η-RGO $\mathcal{O}$ for $f_{\mathrm{oracle}} = f$ to use in Theorem 68.

Our implementation of $\mathcal{O}$ is a rejection sampling scheme. We use the following helpful guarantee.

Lemma 185 (Rejection sampling)**.** *Let π, $\hat{\pi}$ be distributions on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto p(x)$, $\frac{d\hat{\pi}}{dx}(x) \propto \hat{p}(x)$. Suppose for some $C \ge 1$ and all $x \in \mathbb{R}^d$, $\frac{p(x)}{\hat{p}(x)} \le C$. The following is termed "rejection sampling": repeat independent runs of the following procedure until a point is outputted.*

1. *Draw $x \sim \hat{\pi}$.*
2. *With probability $\frac{p(x)}{C\hat{p}(x)}$, output x.*

Rejection sampling terminates in $\frac{C\int\hat{p}(x)dx}{\int p(x)dx}$ runs in expectation, and the output distribution is π.

Proof. The second claim follows from Bayes' rule which implies the conditional density of the output point is proportional to $\hat{p}(x)\cdot\frac{p(x)}{C\hat{p}(x)} \propto p(x)$, so the distribution is π. To see the first claim, the

probability any sample outputs is

$$\int_x \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x) = \frac{1}{C}\int_x \frac{\int_x p(x)dx}{\int_x \hat{p}(x)dx} d\pi(x) = \frac{\int_x p(x)dx}{C\int_x \hat{p}(x)dx}.$$

The conclusion follows by independence and linearity of expectation. □

We further state a concentration bound shown first in [344] regarding the norm of the gradient of a point drawn from a logsmooth distribution. For completeness, a proof can be found in Appendix I.6; see a formal statement in slightly more generality in Theorem 95.

Proposition 43 (Logsmooth gradient concentration, Corollary 3.3, [344])**.** *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f(x))$ where f is convex and L-smooth. With probability at least $1-\kappa^{-d}$,*

$$\|\nabla f(x)\|_2 \le 3\sqrt{L}d\log\kappa \text{ for } x \sim \pi. \tag{10.18}$$

By the requirements of Theorem 68, the restricted Gaussian oracle $\mathcal{O}$ only must be able to draw samples from densities of the form, for some $y \in \mathbb{R}^d$,

$$\exp\left(-f_{\text{oracle}}(x) - \frac{1}{2\eta}\|x-y\|_2^2\right) = \exp\left(-f(x) - 4Ld\log\kappa\|x-y\|_2^2\right). \tag{10.19}$$

We will use the following Algorithm 60 to implement $\mathcal{O}$.

Algorithm 60: `XSample`(f, y, η)

1 **Input:** L-smooth, μ-strongly convex $f : \mathbb{R}^d \to \mathbb{R}$, $y \in \mathbb{R}^d$, $\eta > 0$;
2 **if** $\|\nabla f(y)\|_2 \le 3\sqrt{L}d\log\kappa$ **then**
3 **while** ***true*** **do**
4 Draw $x \sim \mathcal{N}(y - \nabla f(y), \eta\mathbf{I})$;
5 $\tau \sim \text{Unif}[0,1]$;
6 **if** $\tau \le \exp(f(y) + \langle \nabla f(y), x-y\rangle - f(x))$ **then**
7 **Return:** x;
8 Use [128] to sample x from (10.19) to total variation distance $\frac{\epsilon}{\Theta(\kappa d^2 \log^3(\frac{\kappa d}{\epsilon}))}$ using $O(d\log\frac{\kappa d}{\epsilon})$ queries to ∇f (Theorem 1, [128], where (10.19) has constant condition number);
9 **Return:** x;

Lemma 186. *Let $\eta = \frac{1}{8Ld\log(\kappa)}$, and suppose y satisfies the bound in* (10.18)*, i.e. $\|\nabla f(y)\|_2 \le 3\sqrt{L}d\log\kappa$. Then, Line 3 of Algorithm 60 runs an expected 2 times, and Algorithm 60 samples exactly from* (10.19)*, whenever the condition of Line 1 is met.*

Proof. Note that when the assumption of Line 1 is met, Algorithm 60 is an instantiation of rejection

sampling (Lemma 185) with

$$p(x) = \exp\left(-f(x) - \frac{1}{2\eta}\left\|x-y\right\|_2^2\right),$$
$$\hat{p}(x) = \exp\left(-f(y) - \langle \nabla f(y), x-y\rangle - \frac{1}{2\eta}\left\|x-y\right\|_2^2\right).$$

By convexity, we may take $C = 1$. Next, by applying Fact 27 twice and L-smoothness of f_{oracle},

$$\begin{aligned}
\int_x p(x)dx &\geq \int_x \exp\left(-f(y) - \langle \nabla f(y), x-y\rangle - \frac{1+\eta L}{2\eta}\left\|x-y\right\|_2^2\right)dx \\
&= \exp\left(-f(y) + \frac{\eta}{2(1+\eta L)}\left\|\nabla f(y)\right\|_2^2\right)\int_x \exp\left(-\frac{1+\eta L}{2\eta}\left\|x-y+\frac{\eta}{1+\eta L}\nabla f(y)\right\|_2^2\right)dx \\
&= \exp\left(-f(y) + \frac{\eta}{2(1+\eta L)}\left\|\nabla f(y)\right\|_2^2\right)\left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}}, \\
\int_x \hat{p}(x)dx &= \exp\left(-f(y) + \frac{\eta}{2}\left\|\nabla f(y)\right\|_2^2\right)(2\pi\eta)^{\frac{d}{2}},
\end{aligned}$$

which implies the desired bound (recalling Lemma 185 and our assumed bound on $\left\|\nabla f(y)\right\|_2$)

$$\begin{aligned}
\frac{\int \hat{p}(x)dx}{\int p(x)dx} &\leq \exp\left(\left(\frac{\eta}{2} - \frac{\eta}{2(1+\eta L)}\right)\left\|\nabla f(y)\right\|_2^2\right)(1+\eta L)^{\frac{d}{2}} \\
&\leq 1.5\exp\left(\frac{\eta^2 L}{2(1+\eta L)}\left\|\nabla f(y)\right\|_2^2\right) \leq 2.
\end{aligned}$$

□

We are now equipped to prove our main result concerning well-conditioned densities.

Corollary 45. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp\left(-f(x)\right)$ such that f is L-smooth and μ-strongly convex, and let $\epsilon \in (0,1)$, $\kappa = \frac{L}{\mu}$. Assume access to $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d} f(x)$. Algorithm 59 with $\eta = \frac{1}{8Ld\log(\kappa)}$ using Algorithm 60 as a restricted Gaussian oracle for f uses $O(\kappa d \log\kappa \log\frac{\kappa d}{\epsilon})$ gradient queries in expectation, and obtains ϵ total variation distance to π.*

Proof. By applying Theorem 68 with the chosen η, and noting that the cumulative error due to all calls to Line 10 cannot amount to more than $\frac{\epsilon}{2}$ total variation error throughout the algorithm, it suffices to show that Algorithm 60 uses $O(1)$ gradient queries each iteration in expectation. This happens whenever the condition in Line 1 is met via Lemma 186, so we must show Line 10 is executed with probability $O((d\log\frac{\kappa d}{\epsilon})^{-1})$.

To show this, note that combining Proposition 43 with the warmness of the start x_0 in Algorithm 60, this event occurs with probability at most $\kappa^{-\frac{d}{2}}$ in the first iteration.[13] Since warmness

[13]Formally, Line 2 of Algorithm 59 has $y_1 \sim \mathcal{N}(x_0, \eta\mathbf{I})$, but by smoothness $\left\|\nabla f(y_1)\right\|_2 \leq \left\|\nabla f(x_0)\right\|_2 + L\left\|x-y\right\|_2$ and $L\left\|x-y\right\|_2 \leq \widetilde{O}(L\sqrt{\eta})$ with high probability, adding a negligible constant to the bound of Proposition 43.

is monotonically decreasing[14] throughout using an exact oracle in Algorithm 59, and the total error accumulated due to Line 10 throughout the algorithm is $O((d \log \frac{\kappa d}{\epsilon})^{-1})$, we have the desired conclusion. □

We show a bound nearly-matching Corollary 45 using only value access to f, and with a deterministic iteration complexity (rather than an expected one), as Corollary 49 in Section 10.4.3.

10.4.2 Composite logconcave sampling: proof of Corollary 46

In this section, let π be a distribution on $\mathbb{R}^d$ with density proportional to $\exp(-f(x) - g(x))$, where f is L-smooth and μ-strongly convex (and $\kappa = \frac{L}{\mu}$), and g is convex and admits a restricted Gaussian oracle $\mathcal{O}$. Without loss of generality, we assume that f and g share a minimizer x^* which we have pre-computed; if this is not the case, we can redefine $f(x) \leftarrow f(x) - \langle \nabla f(x^*), x \rangle$ and $g(x) \leftarrow g(x) + \langle \nabla f(x^*), x \rangle$; see Section 10.5.1 for this reduction.

We will instantiate Theorem 68 with $f_{\text{oracle}} = f + g$, which is a μ-strongly convex function. Our main result of this section follows directly from Theorem 68 and using Theorem 69 as the required oracle $\mathcal{O}$, stated more precisely in the following.

Corollary 46. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-f(x)-g(x))$ such that f is L-smooth and μ-strongly convex, and let $\epsilon \in (0,1)$, $\kappa = \frac{L}{\mu}$. Assume access to $x^* = \text{argmin}_{x \in \mathbb{R}^d}\{f(x)+g(x)\}$ and let $\mathcal{O}$ be a restricted Gaussian oracle for g. There is an algorithm (Algorithm 59 using Theorem 69 as a restricted Gaussian oracle) which runs in $O(\kappa d \log^3 \frac{\kappa d}{\epsilon})$ iterations in expectation, each querying a gradient of f and $\mathcal{O}$ a constant number of times, and obtains ϵ total variation distance to π.*

Proof. As discussed at the beginning of this section, assume without loss that f and g both are minimized by x^*. We apply the algorithm of Theorem 68 with $\eta = \frac{1}{L}$ to the μ-strongly convex function $f + g$, which requires one call to $\mathcal{O}$ to implement. Thus, the iteration count parameter in Theorem 68 is $T = O(\kappa \log \frac{\kappa d}{\epsilon})$.

Recall that we chose $\eta = \frac{1}{L}$. To bound the total complexity of this algorithm, it suffices to give an η-RGO $\mathcal{O}^+$ for sampling from distributions with densities of the form, for some $y \in \mathbb{R}^d$,

$$\exp\left(-f(x) - g(x) - \frac{1}{2\eta}\|x - y\|_2^2\right) = \exp\left(-f(x) - g(x) - \frac{L}{2}\|x - y\|_2^2\right)$$

to total variation distance $\frac{\epsilon}{\Theta(T)}$ (see discussion at the end of Section 10.3). To this end, we apply Theorem 69 with the well-conditioned component $f(x) + \frac{L}{2}\|x - y\|_2^2$, the composite component $g(x)$, and the largest possible choice of η. Note that we indeed have access to a restricted Gaussian oracle for g (namely, $\mathcal{O}$), and this choice of well-conditioned component is $2L$-smooth and L-strongly convex, so its condition number is a constant. Thus, Theorem 69 requires $O(d \log^2 \frac{\kappa d}{\epsilon})$ calls to $\mathcal{O}$

[14] This is a standard fact in the literature, and can be seen as follows: each transition step in the chain is a convex combination of warm point masses, preserving warmness.

and gradients of f to implement the desired $\mathcal{O}^+$ on any query y (where we note $\frac{\epsilon}{\Theta(T)} = \frac{1}{\mathrm{poly}(\kappa,d,\epsilon^{-1})}$). Combining these complexity bounds yields the desired conclusion. □

10.4.3 Sampling logconcave finite sums: proof of Corollary 47

In this section, let π be a distribution on $\mathbb{R}^d$ with density proportional to $\exp(-F(x))$, where $F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x)$ is μ-strongly convex, and for all $i \in [n]$, f_i is L-smooth (and $\kappa = \frac{L}{\mu}$). We will instantiate Theorem 68 with $f_{\mathrm{oracle}}(x) = F(x)$, and Theorem 70 as an η-RGO for some choice of η.

More precisely, Theorem 70 shows that given access to the minimizer x^*, only zeroth-order access to the summands of F is necessary to obtain the iteration bound. In order to obtain the minimizer to high accuracy however, variance reduced stochastic gradient methods (e.g. [300]) require $\Omega(n+\kappa)$ gradient queries, which amounts to $\Omega((n+\kappa)d)$ function evaluations. We state a convenient corollary of Theorem 70 which removes the requirement of accessing x^*, via an optimization pre-processing step using the method of [300] (see further discussion in Appendix I.1). This is useful to us in proving Theorem 47 because in the sampling tasks required by the RGO, the minimizer changes (and thus must be recomputed every time).

Corollary 48 (First-order logconcave finite sum sampling)**.** *In the setting of Theorem 70, using [300] to precompute the minimizer x^* and running Algorithm 64 uses $O(n\log\frac{\kappa d}{\epsilon} + \kappa^2 d\log^4\frac{n\kappa d}{\epsilon})$ first-order oracle queries to summands $\{f_i\}_{i\in[n]}$ and obtains ϵ total variation distance to π.*

We now apply the reduction framework developed in Section 10.3 to our Algorithm 64 to obtain an improved query complexity for sampling from logconcave finite sums.

Corollary 47 (Improved first-order logconcave finite sum sampling)**.** *In the setting of Theorem 70, Algorithm 59 using Algorithm 64 and SVRG [300] as a restricted Gaussian oracle for F uses*

$$O\left(n\log\left(\frac{n\kappa d}{\epsilon}\right) + \kappa\sqrt{nd}\log^{3.5}\left(\frac{n\kappa d}{\epsilon}\right) + \kappa d\log^5\left(\frac{n\kappa d}{\epsilon}\right)\right) = \widetilde{O}\left(n + \kappa\max\left(d,\sqrt{nd}\right)\right)$$

queries to first-order oracles for summands $\{f_i\}_{i\in[n]}$, and obtains ϵ total variation distance to π.

Proof. We apply Theorem 68 with μ-strongly convex $f_{\mathrm{oracle}} = F(x)$, using Algorithm 64 as the required η-RGO $\mathcal{O}$ for sampling from distributions with densities of the form

$$\exp\left(-F(x) - \frac{1}{\eta}\left\|x-y\right\|_2^2\right)$$

for some $y \in \mathbb{R}^d$, to total variation $\frac{\epsilon}{\Theta(T)}$ (see Section 10.3) for T the iteration bound of Algorithm 59. We apply Theorem 70 to the function $\widetilde{F}(x) = F(x) + \frac{1}{\eta}\left\|x-y\right\|_2^2$; we can express this in finite sum form by adding $\frac{1}{\eta}\left\|x-y\right\|_2^2$ to every constituent function, and the effect on gradient oracles is $\frac{1}{\eta}(x-y)$.

Note $\widetilde{F}$ has condition number $O(1+\eta L)$. For a given η, the overall complexity is

$$\frac{\log \frac{\kappa d}{\epsilon}}{\eta\mu}\left(n\log\left(\frac{n\kappa d}{\epsilon}\right)+d\log^4\left(\frac{n\kappa d}{\epsilon}\right)+(\eta L)^2 d\log^4\left(\frac{n\kappa d}{\epsilon}\right)\right)$$

Here, the inner loop complexity uses Corollary 48 to also find the minimizer (for warm starts), and the outer loop complexity is by Theorem 68. The result follows by optimizing over η, namely picking $\eta = \max(\frac{1}{L}, \sqrt{\frac{n}{L^2 d\log^3(n\kappa d/\epsilon)}})$, and that Algorithm 59 always must have at least one iteration. □

Note the only place that Corollary 47 used gradient evaluations was in determining minimizers of subproblems, via the first step of Corollary 48. Consider now the $n = 1$ case. By running e.g. accelerated gradient descent for smooth and strongly convex functions, it is well-known [417] that we can obtain a minimizer in $\widetilde{O}(\sqrt{\kappa})$ iterations, each querying a gradient oracle, where κ is the condition number. By smoothness, we can approximate every coordinate of the gradient to arbitrary precision using 2 function evaluations, so this is a $\widetilde{O}(\sqrt{\kappa}d)$ value oracle complexity.

Finally, for every optimization subproblem in Corollary 47 where $\eta = (L\cdot \mathrm{polylog}\frac{\kappa d}{\epsilon})^{-1}$, the condition number is a constant, which amounts to a $\widetilde{O}(d)$ value oracle complexity for computing a minimizer. This is never the dominant term compared to Theorem 70, yielding the following conclusion.

Corollary 49. *In the setting of Corollary 45, Algorithm 59 using Algorithm 64 as a restricted Gaussian oracle uses $O(\kappa d\log^2\frac{\kappa d}{\epsilon})$ value queries and obtains ϵ total variation distance to π.*

We note that the polylogarithmic factor is significantly improved when compared to Corollary 47 by removing the random sampling steps in Algorithm 64. A precise complexity bound of the resulting Metropolized random walk, a zeroth-order algorithm mixing in $O(\kappa^2 d\log\frac{\kappa d}{\epsilon})$ for a logconcave distribution with condition number κ, is given as Theorem 2 of [128].

Finally, in the case $n \geq 1$, we also exhibit an improved query complexity in terms of an entirely zeroth-order sampling algorithm which interpolates with Corollary 49 (up to logarithmic factors). By trading off the $\widetilde{O}(nd+\kappa d)$ zeroth-order complexity of minimizing a finite sum function [300], and the $\widetilde{O}(\kappa^2 d)$ zeroth-order complexity of sampling, we can run Theorem 68 for the optimal choice of $\eta = \widetilde{O}(\frac{\sqrt{n}}{L})$. The overall zeroth-order complexity can be seen to be $\widetilde{O}(nd+\sqrt{n}\kappa d)$.

10.5 Composite logconcave sampling with a restricted Gaussian oracle

In this section, we provide our "base sampler" for composite logconcave densities as Algorithm 61, and give its guarantees by proving Theorem 69. Throughout, fix distribution π with density

$$\frac{d\pi}{dx}(x) \propto \exp\left(-f(x) - g(x)\right), \text{where } f : \mathbb{R}^d \to \mathbb{R} \text{ is } L\text{-smooth, } \mu\text{-strongly convex,} \\ \text{and } g : \mathbb{R}^d \to \mathbb{R} \text{ admits a restricted Gaussian oracle } \mathcal{O}. \tag{10.20}$$

We will define $\kappa := \frac{L}{\mu}$, and assume that we have precomputed $x^* := \text{argmin}_{x \in \mathbb{R}^d} \{f(x) + g(x)\}$. Our algorithm proceeds in stages following the outline in Section 10.1.4.

1. `Composite-Sample` is reduced to `Composite-Sample-Shared-Min`, which takes as input a distribution with negative log-density $f + g$, where f and g share a minimizer; this reduction is given in Section 10.5.1, and the remainder of the section handles the shared-minimizer case.

2. The algorithm `Composite-Sample-Shared-Min` is a rejection sampling scheme built on top of sampling from a joint distribution $\hat{\pi}$ on $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$ whose x-marginal approximates π. We give this reduction in Section 10.5.2.

3. The bulk of our analysis is for `Sample-Joint-Dist`, an alternating marginal sampling algorithm for sampling from $\hat{\pi}$. To implement marginal sampling, it alternates calls to $\mathcal{O}$ and a rejection sampling algorithm `YSample`. We prove its correctness in Section 10.5.3.

We put these pieces together in Section 10.5.4 to prove Theorem 69. We remark that for simplicity, we will give the algorithms corresponding to the largest value of step size η in the theorem statement; it is straightforward to modify the bounds to tolerate smaller values of η, which will cause the mixing time to become correspondingly larger (in particular, the value of K in Algorithm 63).

Algorithm 61: `Composite-Sample`(π, x^*, ϵ)

Input: Distribution π of form (10.20), x^* minimizing negative log-density of π, $\epsilon \in [0, 1]$;
Output: Sample x from a distribution π' with $\|\pi' - \pi\|_{\text{TV}} \le \epsilon$;
$\tilde{f}(x) \leftarrow f(x) - \langle \nabla f(x^*), x \rangle$, $\tilde{g}(x) \leftarrow g(x) + \langle \nabla f(x^*), x \rangle$;
Return: `Composite-Sample-Shared-Min`$(\pi, \tilde{f}, \tilde{g}, x^*, \epsilon)$;

10.5.1 Reduction from `Composite-Sample` to `Composite-Sample-Shared-Min`

Correctness of `Composite-Sample` is via the following properties.

Proposition 44. *Let $\tilde{f}$ and $\tilde{g}$ be defined as in `Composite-Sample`.*

Algorithm 62: `Composite-Sample-Shared-Min`$(\pi, f, g, x^*, \epsilon)$

Input: Distribution π of form (10.20), where f and g are both minimized by x^*, $\epsilon \in [0, 1]$;
Output: Sample x from a distribution π' with $\|\pi' - \pi\|_{\mathrm{TV}} \leq \epsilon$;
while ***true*** **do**
  Define the set

$$\Omega := \left\{ x \mid \|x - x^*\|_2 \leq 4\sqrt{\frac{d \log(288\kappa/\epsilon)}{\mu}} \right\} \tag{10.21}$$

  ;
  $x \leftarrow$ `Sample-Joint-Dist`$(f, g, x^*, \mathcal{O}, \frac{\epsilon}{18})$;
  if $x \in \Omega$ **then**
    $\tau \sim \mathrm{Unif}[0, 1]$;
    $y \leftarrow$ `YSample`(f, x, η);
    $\alpha \leftarrow \exp\left(f(y) - \langle \nabla f(x), y - x \rangle - \frac{L}{2}\|y - x\|_2^2 + g(x) + \frac{\eta L^2}{2}\|x - x^*\|_2^2\right)$;
    $\hat{\theta} \leftarrow \exp\left(-f(x) - g(x) + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)(1 + \eta L)^{\frac{d}{2}}\alpha$;
    if $\tau \leq \frac{\hat{\theta}}{4}$ **then**
      Return: x;

1. *The density* $\propto \exp(-f(x) - g(x))$ *is the same as the density* $\propto \exp(-\tilde{f}(x) - \tilde{g}(x))$.
2. *Assuming first-order (function and gradient evaluation) access to* f*, and restricted Gaussian oracle access to* g*, we can implement the same accesses to* $\tilde{f}$*,* $\tilde{g}$ *with constant overhead.*
3. $\tilde{f}$ *and* $\tilde{g}$ *are both minimized by* x^*.

Proof. For f and g with properties as in (10.20), with x^* minimizing $f + g$, define the functions

$$\tilde{f}(x) := f(x) - \langle \nabla f(x^*), x \rangle\,,\ \tilde{g}(x) := g(x) + \langle \nabla f(x^*), x \rangle\,,$$

and observe that $\tilde{f} + \tilde{g} = f + g$ everywhere. This proves the first claim. Further, implementation of a first-order oracle for $\tilde{f}$ and a restricted Gaussian oracle for $\tilde{g}$ are immediate assuming a first-order oracle for f and a restricted Gaussian oracle for g, showing the second claim; any quadratic shifted by a linear term is the sum of a quadratic and a constant. We now show $\tilde{f}$ and $\tilde{g}$ have the same minimizer. By strong convexity, $\tilde{f}$ has a unique minimizer; first-order optimality shows that

$$\nabla \tilde{f}(x^*) = \nabla f(x^*) - \nabla f(x^*) = 0,$$

so this unique minimizer is x^*. Moreover, optimality of x^* for $f + g$ implies that for all $x \in \mathbb{R}^d$,

$$\langle \partial g(x^*) + \nabla f(x^*), x^* - x \rangle \leq 0.$$

Algorithm 63: `Sample-Joint-Dist`$(f, g, x^*, \eta, \mathcal{O}, \delta)$

Input: f, g of form (10.20) both minimized by x^*, $\delta \in [0,1]$, $\eta > 0$, $\mathcal{O}$ restricted Gaussian oracle for g;

Output: Sample x from a distribution $\hat{\pi}'$ with $\|\hat{\pi}' - \hat{\pi}\|_{\mathrm{TV}} \leq \delta$, where we overload $\hat{\pi}$ to mean the marginal of (10.22) on the x variable;

$\eta \leftarrow \frac{1}{32L\kappa d \log(16\kappa/\delta)}$;

Let $\hat{\pi}$ be the density with

$$\frac{d\hat{\pi}}{dx}(z) \propto \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y - x\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) \tag{10.22}$$

;

Call $\mathcal{O}$ to sample $x_0 \sim \pi_{\mathrm{start}}$, for

$$\frac{d\pi_{\mathrm{start}}(x)}{dx} \propto \exp\left(-\frac{L + \eta L^2}{2}\|x - x^*\|_2^2 - g(x)\right) \tag{10.23}$$

;

$K \leftarrow \frac{2^{26} \cdot 100}{\eta\mu} \log\left(\frac{d\log(16\kappa)}{4\delta}\right)$ (see Remark 10);

for $k \in [K]$ **do**

  Call `YSample` $\left(f, x_{k-1}, \eta, \frac{\delta}{2Kd\log(\frac{d\kappa}{\delta})}\right)$ to sample $y_k \sim \pi_{x_{k-1}}$ (Algorithm 3), for

$$\frac{d\pi_x}{dy}(y) \propto \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right) \tag{10.24}$$

  ;

  Call $\mathcal{O}$ to sample $x_k \sim \pi_{y_k}$, for

$$\frac{d\pi_y}{dx}(x) \propto \exp\left(-g(x) - \frac{1}{2\eta}\|y - x\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) \tag{10.25}$$

  ;

Return: x_K;

Here, ∂g is a subgradient. This shows first-order optimality of x^* for $\tilde{g}$ also, so x^* minimizes $\tilde{g}$. □

10.5.2 Reduction from `Composite-Sample-Shared-Min` to `Sample-Joint-Dist`

`Composite-Sample-Shared-Min` is a rejection sampling scheme, which accepts samples from subroutine `Sample-Joint-Dist` in the high-probability region Ω defined in (10.21). We give a general analysis for approximate rejection sampling in Appendix I.2.1, and Appendix I.2.1 bounds relationships between distributions π and $\hat{\pi}$, defined in (10.20) and (10.22) respectively (i.e. relative densities and normalization constant ratios). Combining these pieces proves the following main claim.

Proposition 45. *Let $\eta = \frac{1}{32L\kappa d \log(288\kappa/\epsilon)}$, and assume `Sample-Joint-Dist`$(f, g, x^*, \mathcal{O}, \delta)$ samples within δ total variation of the x-marginal on* (10.22). *`Composite-Sample-Shared-Min` outputs a sample within total variation ϵ of* (10.20) *in an expected $O(1)$ calls to `Sample-Joint-Dist`.*

10.5.3 Implementing `Sample-Joint-Dist`

`Sample-Joint-Dist` alternates between sampling marginals in the joint distribution $\hat{\pi}$, as seen by definitions (10.24), (10.25). We showed that marginal sampling attains the correct stationary distribution as Lemma 181. We bound the conductance of the induced walk on iterates $\{x_k\}$ by combining an isoperimetry bound with a total variation guarantee between transitions of nearby points in Appendix I.2.2. Finally, we give a simple rejection sampling scheme `YSample` as Algorithm 3 for implementing the step (10.24). Since the y-marginal of $\hat{\pi}$ is a bounded perturbation of a Gaussian (intuitively, f is L-smooth and $\eta^{-1} \gg L$), we show in a high probability region that rejecting from the sum of a first-order approximation to f and the Gaussian succeeds in 2 iterations.

Remark 10. *For simplicity of presentation, we were conservative in bounding constants throughout; in practice, we found that the constant in Line 4 is orders of magnitude too large (a constant < 10 sufficed), which can be found as Section 4 of [481]. Several constants were inherited from prior analyses, which we do not rederive to save on redundancy.*

We now give a complete guarantee on the complexity of `Sample-Joint-Dist`.

Proposition 46. *`Sample-Joint-Dist` outputs a point with distribution within δ total variation distance from the x-marginal of $\hat{\pi}$. The expected number of gradient queries per iteration is constant.*

10.5.4 Putting it all together: proof of Theorem 69

We show Theorem 69 follows from the guarantees of Propositions 44, 45, and 46. Formally, Theorem 69 is stated for an arbitrary value of η which is upper bounded by the value in Line 1 of Algorithm 63; however, it is straightforward to see that all our proofs go through for any smaller value. By observing the value of K in `Sample-Joint-Dist`, we see that the number of total iterations in each call to `Sample-Joint-Dist` $O\left(\frac{1}{\eta\mu}\log(\frac{\kappa d}{\epsilon})\right) = O\left(\kappa^2 d \log^2\left(\frac{\kappa d}{\delta}\right)\right)$. Proposition 46

also shows that every iteration, we require an expected constant number of gradient queries and calls to $\mathcal{O}$, the restricted Gaussian oracle for g, and that the resulting distribution has δ total variation from the desired marginal of $\hat{\pi}$. Next, Proposition 45 implies that the number of calls to `Sample-Joint-Dist` in a run of `Composite-Sample-Shared-Min` is bounded by a constant, the choice of δ is $\Theta(\epsilon)$, and the resulting point has total variation ϵ from the original distribution π. Finally, Proposition 44 shows sampling from a general distribution of the form (10.1) is reducible to one call of `Composite-Sample-Shared-Min`, and the requisite oracles are implementable.

10.6 Logconcave finite sums

In this section, we provide our "base sampler" for logconcave finite sums as Algorithm 64, and give its guarantees by proving Theorem 70. Throughout, fix distribution π with density

$$\frac{d\pi}{dx}(x) \propto \exp(-F(x)), \text{ where } F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x) \text{ is } \mu\text{-strongly convex,}$$
$$\text{and for all } i \in [n],\ f_i \text{ is } L\text{-smooth.}$$

We will define $\kappa := \frac{L}{\mu}$, and assume that we have precomputed $x^* := \text{argmin}_{x\in\mathbb{R}^d}\{F(x)\}$. We will also assume explicitly that $\nabla f_i(x^*) = 0$ for all $i \in [n]$ throughout this section (i.e. all f_i are minimized at the same point); this is without loss of generality, by a similar argument as in Proposition 44.

Algorithm 64: `FiniteSum-MRW`(F, h, x_0, p, K)

Input: $F(x) = \frac{1}{n}\sum_{i\in[n]} f_i(x)$, step size $h > 0$, initial x_0, $p \in [0,1]$, iteration count $K \in \mathbb{N}$;
for $0 \le k < K$ **do**
  Draw $\xi_k \sim \mathcal{N}(0, \mathbf{I})$;
  $y_{k+1} \leftarrow x_k + \sqrt{2h}\xi_k$;
  Draw $S_k \subseteq [n]$ by including each $i \in S_k$ independently with probability p;
  For each $i \in [n]$,

$$\gamma_k^{(i)} \leftarrow \begin{cases} \frac{1}{p}\left(\sqrt{\exp\left(-\frac{1}{n}f_i(y_{k+1}) + \frac{1}{n}f_i(x_k)\right)} - 1\right) + 1 & i \in S_k \\ 1 & i \notin S_k \end{cases}$$

  ;
  $\gamma_k \leftarrow \prod_{i=1}^n \gamma_k^{(i)}$, $\tau \sim \text{Unif}[0,1]$;
  if $\tau \le \frac{3}{4}\gamma_k$ *and* $|S_k| \le 2pn$ **then**
    $x_{k+1} \leftarrow y_{k+1}$;
    else
      $x_{k+1} \leftarrow x_k$;
Return: x_K;

Algorithm 64 is the zeroth-order Metropolized random walk of [210] with an efficient, but biased, filter step; the goal of our analysis is to show this bias does not incur significant error.

10.6.1 Approximate Metropolis-Hastings

We first recall the following well-known fact underlying Metropolis-Hastings (MH) filters.

Proposition 47. *Consider a random walk on $\mathbb{R}^d$ with proposal distributions $\{\mathcal{P}_x\}_{x\in\mathbb{R}^d}$ and acceptance probabilities $\{\alpha(x,x')\}_{x,x'\in\mathbb{R}^d}$ conducted as follows: at a current point x,*

1. *Draw a point $x' \sim \mathcal{P}_x$.*
2. *Move the random walk to x' with probability $\alpha(x,x')$, else stay at x.*

Suppose $\mathcal{P}_x(x') = \mathcal{P}_{x'}(x)$ for all pairs $x, x' \in \mathbb{R}^d$, and further $\frac{d\pi}{dx}(x)\alpha(x,x') = \frac{d\pi}{dx}(x')\alpha(x',x)$. Then, π is a stationary distribution for the random walk.

Proof. This follows because the walk satisfies detailed balance (reversibility) with respect to π. □

We propose an algorithm that applies a variant of the Metropolis-Hastings filter to a Gaussian random walk. Specifically, we define the following algorithm, which we call `Inefficient-MRW`.

Definition 46 (`Inefficient-MRW`)**.** *Consider the following random walk for some step size $h > 0$: for each iteration k at a current point $x_k \in \mathbb{R}^d$,*

1. *Set $y_{k+1} \leftarrow x_k + \sqrt{2h}\xi$, where $\xi \sim \mathcal{N}(0, \mathbf{I})$.*
2. *$x_{k+1} \leftarrow y_{k+1}$ with probability $\alpha(x_k, y_{k+1})$ (otherwise, $x_{k+1} \leftarrow x_k$), where*

$$\alpha(x,y) = \begin{cases} 1 & \sqrt{\frac{\exp(-F(y))}{\exp(-F(x))}} > \frac{4}{3}, \\ \frac{3}{4}\sqrt{\frac{\exp(-F(y))}{\exp(-F(x))}} & \frac{3}{4} \le \sqrt{\frac{\exp(-F(y))}{\exp(-F(x))}} \le \frac{4}{3}, \\ \frac{\exp(-F(y))}{\exp(-F(x))} & \sqrt{\frac{\exp(-F(y))}{\exp(-F(x))}} < \frac{3}{4}. \end{cases} \tag{10.26}$$

Lemma 187. *Distribution π with $\frac{d\pi}{dx}(x) \propto \exp(-F(x))$ is stationary for `Inefficient-MRW`.*

Proof. Without loss of generality, assume that π has been normalized so that $\frac{d\pi}{dx}(x) = \exp(-F(x))$. We apply Proposition 47, dropping subscripts in the following. It is clear that $\mathcal{P}_x(y) = \mathcal{P}_y(x)$ for any x, y, so it suffices to check the second condition. When $\frac{3}{4} \le \sqrt{\frac{\exp(-F(y))}{\exp(-F(x))}} \le \frac{4}{3}$, this follows from

$$\frac{d\pi}{dx}(x)\alpha(x,x') = \frac{3}{4}\sqrt{\exp(-F(x) - F(y))} = \frac{d\pi}{dx}(x')\alpha(x',x).$$

The other case is similar (as it is a standard Metropolis-Hastings filter). □

In Algorithm 64, we implement an approximate version of the modified MH filter in Definition 46, where we always assume the pair x, y are in the second case of (10.26). In Lemma 188, we show that if a certain boundedness condition holds, then Algorithm 64 approximates `Inefficient-MRW` well. We then show that the output distributions of `Inefficient-MRW` and our Algorithm 64 have small total variation distance in Lemma 189.

Lemma 188. *Suppose that in an iteration $0 \le k < K$ of Algorithm 64, the following three conditions hold for some parameters R_x, C_ξ, $C_x \in \mathbb{R}_{\ge 0}$:*

1. $\|x_k - x^*\|_2 \le R_x$.
2. $\|\xi_k\|_2 \le C_\xi \sqrt{d}$.
3. *For all $i \in [n]$, $|\nabla f_i(x_k)^\top \xi_k| \le C_x \|\nabla f_i(x_k)\|_2$.*

Then, for any

$$h \le \frac{1}{98 C_x^2 L^2 R_x^2 + 7 L C_\xi^2 d}, \tag{10.27}$$

$\frac{3}{4} \le \sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}} \le \frac{4}{3}$. *Moreover, we have $\mathbb{E}[\gamma_k] = \sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}}$, and when $|S_k| \le 2pn$, $\gamma_k \le \frac{4}{3}$.*

Proof. We first show $\mathbb{E}[\gamma_k] = \sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}}$. Since each $i \in S_k$ is generated independently,

$$\begin{aligned}
\mathbb{E}[\gamma_k] &= \prod_{i \in [n]} \mathbb{E}\left[\gamma_k^{(i)}\right] \\
&= \prod_{i \in [n]} \left[(1-p) + p\left(\frac{1}{p}\left(\sqrt{\exp\left(-\frac{1}{n} f_i(y_{k+1}) + \frac{1}{n} f_i(x_k)\right)} - 1\right) + 1\right)\right] \\
&= \prod_{i \in [n]} \sqrt{\exp\left(-\frac{1}{n} f_i(y_{k+1}) + \frac{1}{n} f_i(x_k)\right)} = \sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}}.
\end{aligned}$$

Next, for any $i \in [n]$, we lower and upper bound $-f_i(y_{k+1}) + f_i(x_k)$. First,

$$\begin{aligned}
-f_i(y_{k+1}) + f_i(x_k) &\le \nabla f_i(x_k)^\top (x_k - y_{k+1}) \\
&\le \sqrt{2h} C_x \|\nabla f_i(x_k)\|_2 \le \sqrt{2h} C_x L R_x.
\end{aligned}$$

The first inequality followed from convexity of f_i, the second from $y_{k+1} - x_k = \sqrt{2h}\xi_k$ and our assumed bound, and the third from smoothness and $\nabla f(x^*) = 0$. To show a lower bound,

$$\begin{aligned}
f_i(y_{k+1}) - f_i(x_k) &\le \nabla f_i(x_k)^\top (y_{k+1} - x_k) + \frac{L}{2} \|y_{k+1} - x_k\|_2^2 \\
&\le \sqrt{2h} C_x L R_x + h L C_\xi^2 d.
\end{aligned}$$

The first inequality was smoothness. Repeating this argument for each $i \in [n]$ and averaging,

$$-\sqrt{2h}C_x L R_x - hLC_\xi^2 d \le -F(y_{k+1}) + F(x_k) \le \sqrt{2h}C_x L R_x. \tag{10.28}$$

Then, when $h \le \frac{1}{98C_x^2L^2R_x^2+7LC_\xi^2 d}$,

$$\frac{3}{4} \le \sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}} \le \frac{4}{3}, \text{ and for all } i \in [n],\ -f_i(y_{k+1}) + f_i(x_k) \le \frac{1}{4}.$$

Thus, we can bound each $\gamma_k^{(i)}$:

$$\gamma_k^{(i)} \le \frac{1}{p}\left(\exp\left(\frac{1}{8n}\right) - 1\right) + 1 \le 1 + \frac{1}{7pn}.$$

Finally, when $|S_k| \le 2pn$, $\gamma_k \le (1 + \frac{1}{7pn})^{2pn} \le \frac{4}{3}$ as desired. □

Lemma 189. *Draw $x_0 \sim \mathcal{N}(x^*, \frac{1}{L}\mathbf{I})$. Let $\hat{\pi}_K$ be the output distribution of the algorithm of Definition 46 for K steps starting from x_0, and let π_K be the output distribution of Algorithm 64 starting from x_0. For any $\delta \in [0,1]$, let $p = \frac{5\log\frac{12K}{\delta}}{n}$ in Algorithm 64. There exist*

$$C_\xi = O\left(1 + \sqrt{\frac{\log\frac{K}{\delta}}{d}}\right),\ C_x = O\left(\sqrt{\log\frac{nK}{\delta}}\right),\ \text{ and } R_x = O\left(\sqrt{\frac{d\log\frac{\kappa K}{\delta}}{\mu}}\right),$$

so that when $h \le \frac{1}{98C_x^2L^2R_x^2+7LC_\xi^2 d}$, we have $\|\pi_K - \hat{\pi}_K\|_{\mathrm{TV}} \le \delta$.

Proof. By the coupling definition of total variation, it suffices to upper bound the probability that the algorithms' trajectories, sharing all randomness in proposing points y_{k+1}, differ. This can happen for two reasons: either we used an incorrect filtering step (i.e. the pair (x_k, y_{k+1}) did not lie in the second case of (10.26)), or we incorrectly rejected in Line 7 of Algorithm 64 because $|S_k| \ge 2pn$. We bound the error due to either happening over any iteration by δ, yielding the conclusion.

Incorrect filtering. Consider some iteration k. Lemma 188 shows that as long as its three conditions hold in iteration k, we are in the second case of (10.26), so it suffices to show all conditions hold. By Fact 28 and as ξ_k is independent of all $\{\nabla f_i(x_k)\}_{i\in[n]}$, with probability at least $1 - \frac{\delta}{2K}$, both of the conditions $\|\xi_k\|_2 \le C_\xi\sqrt{d}$ and[15] $|\nabla f_i(x_k)^\top \xi_k| \le C_x \|\nabla f_i(x_k)\|_2$ for all $i \in [n]$ hold for some

$$C_\xi = O\left(1 + \sqrt{\frac{\log\frac{K}{\delta}}{d}}\right),\ C_x = O\left(\sqrt{\log\frac{nK}{\delta}}\right).$$

[15]We recall that the distribution of $v^\top\xi$ for $\xi \sim \mathcal{N}(0, \mathbf{I})$ is the one-dimensional $\mathcal{N}(0, \|v\|_2^2)$.

Next, $x_0 \sim \mathcal{N}(x^*, \frac{1}{L}\mathbf{I})$ is drawn from a $\kappa^{\frac{d}{2}}$ warm start for π. By Fact 28, we have $\|x_0 - x^*\|_2 \le R_x$ for x_0 drawn from π with probability at least $1 - \frac{\delta}{4K} \cdot \kappa^{-\frac{d}{2}}$, for some

$$R_x = O\left(\sqrt{\frac{d \log \frac{\kappa K}{\delta}}{\mu}}\right).$$

Since warmness of the exact algorithm of Definition 46 is monotonic, as long as the trajectories have not differed up to iteration k, $\|x_k - x^*\|_2 \le R_x$ also holds with probability $\ge 1 - \frac{\delta}{4K}$. Inductively, the total variation error caused by incorrect filtering over K steps is at most $\frac{3\delta}{4}$.

Error due to large $|S_k|$. Supposing all the conditions of Lemma 188 are satisfied in iteration k, we show that with high probability, `Inefficient-MRW` and Algorithm 64 make the same accept or reject decision. By Lemma 188, `Inefficient-MRW` (10.26) accepts with probability $\alpha'_k = \frac{3}{4}\sqrt{\frac{\exp(-F(y_{k+1}))}{\exp(-F(x_k))}}$. On the other hand, Algorithm 64 accepts with probability

$$\alpha_k = \frac{3}{4}\mathbb{E}\left[\gamma_k \mid |S_k| \le 2pn\right] \cdot \Pr[|S_k| \le 2pn].$$

The total variation between the output distributions is $|\alpha_k - \alpha'_k|$. Further, since by Lemma 188,

$$\begin{aligned}
\alpha'_k &= \frac{3}{4}\mathbb{E}\left[\gamma_k\right] \\
&= \frac{3}{4}\left(\mathbb{E}\left[\gamma_k \mid |S_k| \le 2pn\right] \cdot \Pr[|S_k| \le 2pn] + \mathbb{E}\left[\gamma_k \mid |S_k| > 2pn\right] \cdot \Pr[|S_k| > 2pn]\right) \\
&= \alpha_k + \frac{3}{4}\mathbb{E}\left[\gamma_k \mid |S_k| > 2pn\right] \cdot \Pr[|S_k| > 2pn],
\end{aligned}$$

it suffices to upper bound this latter quantity. First, by Lemma 190, when $p = \frac{5 \log \frac{12K}{\delta}}{n}$, we have $\Pr[|S_k| > 2pn] \le \frac{\delta}{12K}$. Finally, since each $i \in S_k$ is generated independently,

$$\begin{aligned}
\mathbb{E}\left[\gamma_k \mid |S_k| > 2pn\right] &\le \max_{S' : |S'| = 2pn} \mathbb{E}\left[\prod_{i \in [n]} \gamma_k^{(i)} \mid S' \subseteq S_k\right] \\
&\le 2\mathbb{E}\left[\prod_{i \in [n] \setminus S'} \gamma_k^{(i)}\right] = 2\sqrt{\prod_{[n] \setminus S'} \exp\left(-\frac{1}{n} f_i(y_{k+1}) + \frac{1}{n} f_i(x_k)\right)} \le 4.
\end{aligned}$$

Here, we used Lemma 188 applied to the set S', and the upper bound (10.28) we derived earlier. Combining these calculations shows that the total variation distance incurred in any iteration k due to $|S_k|$ being too large is at most $\frac{\delta}{4K}$, so the overall contribution over K steps is at most $\frac{\delta}{4}$. □

We used the following helper lemma in our analysis.

Lemma 190. *Let $S \subseteq [n]$ be formed by independently including each $i \in [n]$ with probability p. Then,*

$$\Pr\left[|S| > 2pn\right] \le \exp\left(-\frac{3pn}{14}\right).$$

Proof. For $i \in [n]$, let $\mathbf{1}_{i\in S}$ be the indicator random variable of the event $i \in S$, so $\mathbb{E}\left[\mathbf{1}_{i\in S}\right] = p$ and

$$\mathrm{Var}\left[\mathbf{1}_{i\in S} - p\right] = p(1-p)^2 + (1-p)p^2 \le 2p.$$

By Bernstein's inequality,

$$\Pr\left[\sum_{i\in[n]} \mathbf{1}_{i\in S} \ge np + r\right] \le \exp\left(-\frac{\frac{1}{2}r^2}{2np + \frac{1}{3}r}\right).$$

In particular, when $r = pn$, we have the desired conclusion. □

10.6.2 Conductance analysis

We next bound the mixing time of `Inefficient-MRW`, using the following result from prior work. We remark that in our application, the $\log\beta$ term is non-dominant.

Proposition 48 (Lemma 1, Lemma 2, [128])**.** *Let a random walk with a μ-strongly logconcave stationary distribution π on $x \in \mathbb{R}^d$ have transition distributions $\{\mathcal{T}_x\}_{x\in\mathbb{R}^d}$. For some $\epsilon \in [0,1]$, let convex set $\Omega \subseteq \mathbb{R}^d$ have $\pi(\Omega) \ge 1 - \frac{\epsilon^2}{2\beta^2}$. Let π_{start} be a β-warm start for π, and let the algorithm be initialized at $x_0 \sim \pi_{\mathrm{start}}$. Suppose for any $x, x' \in \Omega$ with $\|x - x'\|_2 \le \Delta$,*

$$\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\mathrm{TV}} \le \frac{7}{8}. \tag{10.29}$$

Then, the random walk mixes to total variation distance within ϵ of π in $O(\log\beta + \frac{1}{\Delta^2\mu}\log\frac{\log\beta}{\epsilon})$ iterations.

Consider an iteration of `Inefficient-MRW` from $x_k = x$. Let $\mathcal{P}_x$ be the density of y_{k+1}, and let $\mathcal{T}_x$ be the density of x_{k+1} after filtering. Define a convex set $\Omega \subseteq \mathbb{R}^d$ parameterized by $R_\Omega \in \mathbb{R}_{\ge 0}$:

$$\Omega = \{x \in \mathbb{R}^d : \|x - x^*\|_2 \le R_\Omega\}.$$

We show that for two close points $x, x' \subseteq \Omega$, the total variation between $\mathcal{T}_x$ and $\mathcal{T}_{x'}$ is small.

Lemma 191. *For some $h = O(\frac{1}{L^2R_\Omega^2 + Ld})$ and $x, x' \subseteq \Omega$ with $\|x - x'\|_2 \le \frac{1}{8}\sqrt{h}$, $\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\mathrm{TV}} \le \frac{7}{8}$.*

Proof. By the triangle inequality of total variation distance,

$$\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\mathrm{TV}} \le \|\mathcal{T}_x - \mathcal{P}_x\|_{\mathrm{TV}} + \|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}} + \|\mathcal{T}_{x'} - \mathcal{P}_{x'}\|_{\mathrm{TV}}.$$

First, by Pinsker's inequality and the KL divergence between Gaussian distributions,

$$\|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}} \leq \sqrt{2\mathrm{KL}(\mathcal{P}_x||\mathcal{P}_{x'})} = \frac{\|x - x'\|_2}{\sqrt{2h}}.$$

When $\|x - x'\|_2 \leq \frac{1}{8}\sqrt{h}$, $\|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}} \leq \frac{1}{8}$. Next, we bound $\|\mathcal{T}_x - \mathcal{P}_x\|_{\mathrm{TV}}$: by a standard calculation (e.g. Lemma D.1 of [344]), we have

$$\|\mathcal{T}_x - \mathcal{P}_x\|_{\mathrm{TV}} = 1 - \frac{3}{4}\mathbb{E}_{\xi_{k+1}}\left[\sqrt{\frac{\exp\left(-F(y_{k+1})\right)}{\exp\left(-F(x_k)\right)}}\right].$$

We show that $\|\mathcal{T}_x - \mathcal{P}_x\|_{\mathrm{TV}} \leq \frac{3}{8}$. It suffices to show that $\mathbb{E}_{\xi_{k+1}}\left[\sqrt{\exp\left(-F(y_{k+1}) + F(x_k)\right)}\right] \geq \frac{5}{6}$.

Since $\frac{15}{16}\sqrt{\exp\left(-\frac{1}{16}\right)} \geq \frac{5}{6}$, it suffices to show that with probability at least $\frac{15}{16}$ over the randomness of ξ_{k+1}, $-F(y_{k+1}) + F(x_k) \geq -\frac{1}{16}$. As $\xi_{k+1} \sim \mathcal{N}(0, \mathbb{I}_d)$, by applying Fact 28 twice,

$$\begin{aligned} \Pr\left[\|\xi_{k+1}\|_2^2 > 36d\right] &\leq \exp(-4) \leq \frac{1}{32}, \\ \Pr\left[\left|\nabla F(x_k)^\top \xi_{k+1}\right|^2 \geq 36\left\|\nabla F(x_k)\right\|_2^2\right] &\leq \frac{1}{32}. \end{aligned} \tag{10.30}$$

We upper bound the term $F(y_{k+1}) - F(x_k)$ by smoothness and Cauchy-Schwarz:

$$\begin{aligned} F(y_{k+1}) - F(x_k) &\leq \nabla F(x_k)^\top (y_{k+1} - x_k) + \frac{L}{2}\|y_{k+1} - x_k\|_2^2 \\ &\leq \sqrt{2h}\left|\nabla F(x_k)^\top \xi_{k+1}\right| + hL\|\xi_{k+1}\|_2^2. \end{aligned}$$

Then, since $\|\nabla F(x_k)\| \leq LR_\Omega$ when $x \in \Omega$, it is enough to choose $h = O(\frac{1}{L^2R_\Omega^2 + Ld})$ so that

$$-F(y_{k+1}) + F(x_k) \geq -\frac{1}{16},$$

as long as the events of (10.30) hold, which occurs with probability at least $\frac{15}{16}$. Similarly, we can show that $\|\mathcal{T}_{x'} - \mathcal{P}_{x'}\|_{\mathrm{TV}} \leq \frac{3}{8}$. Combining the three bounds, we have the desired conclusion.

□

Theorem 70. *Let π be a distribution on $\mathbb{R}^d$ with $\frac{d\pi}{dx}(x) \propto \exp(-F(x))$, where $F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ is μ-strongly convex, f_i is L-smooth and convex $\forall i \in [n]$, $\kappa = \frac{L}{\mu}$, and $\epsilon \in (0, 1)$. Assume access to $x^* = \mathrm{argmin}_{x\in\mathbb{R}^d} F(x)$. Algorithm 64 uses $O\left(\kappa^2 d \log^4 \frac{n\kappa d}{\epsilon}\right)$ value queries to summands $\{f_i\}_{i\in[n]}$, and obtains ϵ total variation distance to π.*

Proof. First, $\mathcal{N}(x^*, \frac{1}{L}\mathbf{I})$ yields a $\beta = \kappa^{\frac{d}{2}}$-warm start for π (see e.g. [210]). For this value of β, by

Fact 28 it suffices to choose

$$R_\Omega = \Theta\left(\sqrt{\frac{d \log \frac{\kappa}{\epsilon}}{\mu}}\right)$$

for $\pi(\Omega) \geq 1 - \frac{\epsilon^2}{2\beta^2}$. Letting $\delta = \frac{\epsilon}{2}$, we will choose the step size h and iteration count K so that

$$\frac{1}{h} = \Theta\left(L\kappa d \log^2 \frac{n\kappa d}{\epsilon}\right), \; K = \Theta\left(\kappa^2 d \log^3 \frac{n\kappa d}{\epsilon}\right)$$

have constants compatible with Lemma 189. Note that this choice of h is also sufficiently small to apply Lemma 191 for our choice of R_Ω. By applying Proposition 48 to the algorithm of Definition 46, and using the bound from Lemma 191, in K iterations `Inefficient-MRW` will mix to total variation distance δ to π. Furthermore, applying Lemma 189, we conclude that Algorithm 64 has total variation distance at most $2\delta = \epsilon$ from π.

It remains to bound the oracle complexity of Algorithm 64. Note in every iteration, we never compute more than $4pn$ values of $\{f_i\}_{i\in[n]}$, since we always reject if $|S_k| \geq 2pn$, and we only compute values for indices in S_k. For the value of p in Lemma 189, this amounts to $O(\log \frac{n\kappa d}{\epsilon})$ value queries. □

10.7 Lower bound for MALA on Gaussians

In this section, we derive a upper bound on the spectral gap of MALA when the target distribution is restricted to being a multivariate Gaussian (i.e. its negative log-density is a quadratic in some well-conditioned matrix $\mathbf{A}$). Throughout this section we will let $f(x) = \frac{1}{2}x^\top \mathbf{A} x$ for some $\mathbf{I} \preceq \mathbf{A} \preceq \kappa \mathbf{I}$. We remark here that without loss of generality, we have assumed that the minimizer of f is the all-zeros vector and the strong convexity parameter is $\mu = 1$. These follow from invariance of condition number under linear translations and scalings of the variable.

Next, we define a specific hard quadratic function we will consider in this section, $f_{\mathrm{hq}} : \mathbb{R}^d \to \mathbb{R}$. Specifically, f_{hq} will be a quadratic in a diagonal matrix $\mathbf{A}$ which has $\mathbf{A}_{11} = 1$ and $\mathbf{A}_{ii} = \kappa$ for $2 \leq i \leq d$. We can rewrite this as

$$f_{\mathrm{hq}}(x) := \sum_{i\in[d]} f_i(x_i), \text{ where } f_i(c) = \begin{cases} \frac{1}{2}c^2 & i = 1 \\ \frac{\kappa}{2}c^2 & 2 \leq i \leq d \end{cases}. \tag{10.31}$$

Notice that f_{hq} is coordinate-wise separable, and behaves identically on coordinates $2 \leq i \leq d$ (and differently on coordinate 1). To this end for a vector $v \in \mathbb{R}^d$, we will denote its first coordinate by $v_1 \in \mathbb{R}$, and its remaining coordinates by $v_{-1} \in \mathbb{R}^{d-1}$. This will help us analyze the behavior of these components separately, and simplify notation.

We next show that for coordinate-separable functions with well-behaved first coordinate, such

as our f_{hq}, the spectral gap (defined in (10.10)) of the MALA Markov chain is governed by the step size h. The following is an extension of an analogous proof in [137].

Lemma 192. *Consider the MALA Markov chain* (10.13)*, with stationary distribution π^* with negative log-density f. Suppose f is coordinate-wise separable (i.e. $f(x) = \sum_{i\in[d]} f_i(x_i)$). If $f(x) = f(-x)$ for all $x \in \mathbb{R}^d$, f_1 is $O(1)$-smooth, and $\mathbb{E}_{x_1\sim\exp(-f_1)}[x_1^2] = \Theta(1)$, the spectral gap* (10.10) *is $O(h + h^2)$.*

Proof. Recalling the definition (10.10), we choose $g(x) = x_1$; note that by symmetry of f around the origin, we have $\mathbb{E}_{\pi^*}[g] = 0$, and thus by our assumption,

$$\mathrm{Var}_{\pi^*}[g] = \mathbb{E}_{x\sim\pi^*}[x_1^2] = \Theta(1).$$

Here we used that π^* is a product distribution. Thus it suffices to upper bound $\mathcal{E}(g, g)$:

$$\begin{aligned}
\mathcal{E}(g,g) &= \frac{1}{2}\iint (x_1 - y_1)^2 \mathcal{T}_x(y) d\pi^*(x) dy \\
&\le \frac{1}{2}\iint (x_1 - y_1)^2 \mathcal{P}_x(y) d\pi^*(x) dy \\
&= \frac{1}{2}\mathbb{E}_{x\sim\pi^*,\xi\sim\mathcal{N}(0,1)}\left[\left(hf_1'(x_1) - \sqrt{2h}\xi\right)^2\right] \\
&\le \mathbb{E}_{x\sim\pi^*}\left[h^2\left(f_1'(x_1)\right)^2\right] + 2\mathbb{E}_{\xi\sim\mathcal{N}(0,1)}\left[h\xi^2\right] \\
&\le O(h^2)\mathbb{E}_{x\sim\pi^*}\left[x_1^2\right] + 2h = O\left(h + h^2\right).
\end{aligned}$$

In the second line, we used that whenever the Markov chain rejects the distribution both terms are zero; in the third, we used the definition of the MALA proposals; in the fourth, we used $(a + b)^2 \le 2a^2 + 2b^2$ for $a, b \in \mathbb{R}$. Finally, the last line used that symmetry implies that the minimizer of f is the origin, so applying Lipschitzness and $f_1'(0) = 0$ yields the desired bound. □

This immediately implies a spectral gap bound on our hard function f_{hq}.

Corollary 50. *The spectral gap of the MALA Markov chain for sampling from the density proportional to $\exp(-f_{\mathrm{hq}})$, where f_{hq} is defined in* (10.31)*, is $O(h + h^2)$.*

It remains to give a lower bound on the step size h, which we accomplish by upper bounding the acceptance probability of MALA. We will give a step size analysis for a fairly general characterization of Markov chains, where the proposal distribution from a point x is

$$\begin{aligned}
y = \begin{pmatrix} y_1 \\ y_{-1} \end{pmatrix}, \text{ where } y_1 &= (1 - \alpha_1)x_1 + \beta_1 g_1 \\
\text{and } y_{-1} &= (1 - \alpha_{-1})x_{-1} + \beta_{-1} g_{-1}, \text{ for } g \sim \mathcal{N}(0, \mathbf{I}).
\end{aligned} \tag{10.32}$$

To be concrete, recall that the proposal distribution for MALA (10.13) is given by $y = x - h\mathbf{A}x + \sqrt{2h}g$. For the $\mathbf{A}$ used in defining f_{hq}, this is of the form (10.32) with the specific parameters

$$\alpha_1 = h,\ \alpha_{-1} = h\kappa,\ \beta_1 = \beta_{-1} = \sqrt{2h}.$$

However, this more general characterization will save significant amounts of recalculation when analyzing updates of the HMC Markov chain in Section 10.10. Recalling the formula (10.13), we first give a closed form for the acceptance probability.

Lemma 193. *For $f(x) = \frac{1}{2}x^\top \mathbf{A}x$, we have*

$$f(x) - f(y) + \frac{1}{4h}\left(\|y - (x - h\nabla f(x))\|_2^2 - \|x - (y - h\nabla f(y))\|_2^2\right) = \frac{h}{4}\|x\|_{\mathbf{A}^2}^2 - \frac{h}{4}\|y\|_{\mathbf{A}^2}^2.$$

Supposing y is of the form in (10.32) *and $\mathbf{A}$ is as in* (10.31)*, we have*

$$\begin{aligned}\frac{h}{4}\|x\|_{\mathbf{A}^2}^2 - \frac{h}{4}\|y\|_{\mathbf{A}^2}^2 &= \frac{h}{4}\left(\left(2\alpha_1 - \alpha_1^2\right)x_1^2 - \beta_1^2 g_1^2 - 2(1-\alpha_1)\beta_1 x_1 g_1\right) \\ &\quad + \frac{h\kappa^2}{4}\left(\left(2\alpha_{-1} - \alpha_{-1}^2\right)\|x_{-1}\|_2^2 - \beta_{-1}^2\|g_{-1}\|_2^2 - 2(1-\alpha_{-1})\beta_{-1}\langle x_{-1}, g_{-1}\rangle\right).\end{aligned}$$

Proof. This is a direct computation which we perform here for completeness: the given quantity is

$$\begin{aligned}&\frac{1}{2}\|x\|_{\mathbf{A}}^2 - \frac{1}{2}\|y\|_{\mathbf{A}}^2 + \frac{1}{4h}\left(\|y - x + h\mathbf{A}x\|_2^2 - \|x - y + h\mathbf{A}y\|_2^2\right) \\ &= \frac{1}{2}\|x\|_{\mathbf{A}}^2 - \frac{1}{2}\|y\|_{\mathbf{A}}^2 + \frac{1}{2}\langle y - x, \mathbf{A}x\rangle + \frac{h}{4}\|x\|_{\mathbf{A}^2}^2 - \frac{1}{2}\langle x - y, \mathbf{A}y\rangle - \frac{h}{4}\|y\|_{\mathbf{A}^2}^2 = \frac{h}{4}\|x\|_{\mathbf{A}^2}^2 - \frac{h}{4}\|y\|_{\mathbf{A}^2}^2.\end{aligned}$$

The second equality follows from expanding the definition of y:

$$\begin{aligned}\|x\|_{\mathbf{A}^2}^2 - \|y\|_{\mathbf{A}^2}^2 &= x_1^2 - ((1-\alpha_1)x_1 + \beta_1 g_1)^2 + \kappa^2\left(\|x_{-1}\|_2^2 - \|(1-\alpha_{-1})x_{-1} + \beta_{-1}g_{-1}\|_2^2\right) \\ &= \left(2\alpha_1 - \alpha_1^2\right)x_1^2 - \beta_1^2 g_1^2 - 2(1-\alpha_1)\beta_1 x_1 g_1 \\ &\quad + \kappa^2\left(\left(2\alpha_{-1} - \alpha_{-1}^2\right)\|x_{-1}\|_2^2 - \beta_{-1}^2\|g_{-1}\|_2^2 - 2(1-\alpha_{-1})\beta_{-1}\langle x_{-1}, g_{-1}\rangle\right).\end{aligned}$$

□

Corollary 51. *For any fixed $x \in \mathbb{R}^d$, and supposing y is of the form in* (10.32) *and $\mathbf{A}$ is as in* (10.31)*,*

$$\begin{aligned}&\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[f(x) - f(y) + \frac{1}{4h}\left(\|y - (x - h\nabla f(x))\|_2^2 - \|x - (y - h\nabla f(y))\|_2^2\right)\right] \\ &= \frac{h}{4}\left(\left(2\alpha_1 - \alpha_1^2\right)x_1^2 - \beta_1^2\right) + \frac{h\kappa^2}{4}\left(\left(2\alpha_{-1} - \alpha_{-1}^2\right)\|x_{-1}\|_2^2 - \beta_{-1}^2(d-1)\right).\end{aligned}$$

Proof. This follows from Lemma 193, independence of g and x, and linearity of trace and expectation applied on squared coordinates of g, where we recognize $\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}[gg^\top] = \mathbf{I}$. □

Next, for a fixed x, consider the random variables R_i^x:

$$R_i^x = \begin{cases} \frac{h}{4}\left(\left(2\alpha_1 - \alpha_1^2\right)x_1^2 - \beta_1^2 g_1^2 - 2(1-\alpha_1)\beta_1 x_1 g_1\right) & i = 1 \\ \frac{h\kappa^2}{4}\left(\left(2\alpha_{-1} - \alpha_{-1}^2\right)x_i^2 - \beta_{-1}^2 g_i^2 - 2(1-\alpha_{-1})\beta_{-1} x_i g_i\right) & 2 \le i \le d \end{cases}$$

where $g \sim \mathcal{N}(0, \mathbf{I})$ is a standard Gaussian random vector. Notice that for a given realization of g, we have by Lemma 193 that

$$\sum_{i\in[d]} R_i^x = \frac{h}{4}\left\|x\right\|_{\mathbf{A}^2} - \frac{h}{4}\left\|y\right\|_{\mathbf{A}^2}. \tag{10.33}$$

We computed the expectation of $\sum_{i\in[d]} R_i^x$ in Corollary 51. We next give a high-probability bound on the deviation of $\sum_{i\in[d]} R_i^x$ from its expectation.

Lemma 194. *With probability at least $1-\delta$ over the randomness of $g \sim \mathcal{N}(0, \mathbf{I})$,*

$$\sum_{i\in[d]} R_i^x - \mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i\in[d]} R_i^x\right] \le 2h|\alpha_1 - 1|\beta_1|x_1|\sqrt{\log\left(\frac{4}{\delta}\right)} + h\beta_1^2\sqrt{\log\left(\frac{4}{\delta}\right)} + 2h\kappa^2|\alpha_{-1} - 1|\beta_{-1}\left\|x_{-1}\right\|_2\sqrt{\log\left(\frac{4}{\delta}\right)} + h\kappa^2\beta_{-1}^2\sqrt{d\log\left(\frac{4}{\delta}\right)}.$$

Proof. In defining $\{R_i^x\}_{i\in[d]}$, the terms involving $\{x_i^2\}_{i\in[d]}$ are deterministic. Thus, we need to upper bound the deviations of the remaining terms, namely

$$S_1^{(1)} := \frac{h}{2}(\alpha_1 - 1)\beta_1 x_1 g_1,\ S_1^{(2)} := \frac{h\beta_1^2}{4}\left(1 - g_1^2\right),$$
$$S_{-1}^{(1)} := \frac{h\kappa^2}{2}(\alpha_{-1} - 1)\beta_{-1}\sum_{2\le i\le d} x_i g_i,\ S_{-1}^{(2)} := \frac{h\kappa^2\beta_{-1}^2}{4}\sum_{2\le i\le d}\left(1 - g_i^2\right).$$

To motivate these definitions, $S_1^{(1)} + S_1^{(2)} + S_{-1}^{(1)} + S_{-1}^{(2)}$ is the left hand side of the display in the lemma statement. We begin with $S_{-1}^{(1)}$. Notice that this is a one-dimensional Gaussian random variable distributed as

$$\mathcal{N}\left(0, \sigma_1^2\right) \text{ where } \sigma_1 := \frac{h\kappa^2}{2}|\alpha_{-1} - 1|\beta_{-1}\left\|x_{-1}\right\|_2.$$

Thus, applying Mill's inequality yields

$$\Pr\left[S_{-1}^{(1)} > t\right] \le \sqrt{\frac{2}{\pi}}\frac{\sigma_1}{t}\exp\left(-\frac{t^2}{2\sigma_1^2}\right) \le \frac{\delta}{4},\ \text{for } t = 4\sigma_1\sqrt{\log\left(\frac{4}{\delta}\right)}.$$

Next, to bound the term $S_{-1}^{(2)}$, define

$$\sigma_2 := \frac{h\kappa^2\beta_{-1}^2}{4}\sqrt{d-1}.$$

Standard χ^2 concentration results (Fact 32) then yield

$$\Pr\left[S_{-1}^{(2)} > t\right] \le \exp\left(-\frac{t^2}{4\sigma_2^2}\right) \le \frac{\delta}{4}, \text{ for } t = 2\sigma_2\sqrt{\log\left(\frac{4}{\delta}\right)}.$$

Similar bounds follow for $S_1^{(1)}$ and $S_1^{(2)}$, whose computations we omit for brevity. Taking a union bound over these four terms yields the desired claim. □

Finally, we have a complete characterization of a bad set $\Omega \subset \mathbb{R}^d$ where, with high probability over the proposal distribution, the acceptance probability is extremely small.

Proposition 49. *Let $x \in \mathbb{R}^d$ satisfy $\|x_{-1}\|_2 \le \sqrt{\frac{2d}{3\kappa}}$ and $|x_1| \le 5\sqrt{\log d}$, and suppose y is of the form in* (10.32) *and* $\mathbf{A}$ *is as in* (10.31)*. Also suppose that*

$$|\alpha_{-1}| \le \frac{3}{5}\beta_{-1}^2\kappa,\ \beta_{-1} = \omega\left(\sqrt{\frac{\log d}{\kappa d}}\right),\ |\alpha_1| = O(|\alpha_{-1}|),\ \beta_1 = O(\beta_{-1}).$$

Then with probability at least $1 - d^{-5}$ over the randomness of $g \sim \mathcal{N}(0, \mathbf{I})$, we have

$$\frac{h}{4}\|x\|_{\mathbf{A}^2} - \frac{h}{4}\|y\|_{\mathbf{A}^2} = -\Omega\left(h\kappa^2\beta_{-1}^2 d\right).$$

Proof. We first handle terms involving x_{-1} and g_{-1}. Combining (10.33), Corollary 51, and Lemma 194, we have with probability at least $1 - \frac{1}{2}d^{-5}$ over the randomness of $g \sim \mathcal{N}(0, \mathbf{I})$ that $\|x_{-1}\|_{\mathbf{A}_{-1}^2}^2 - \|y_{-1}\|_{\mathbf{A}_{-1}^2}^2$ (where $\mathbf{A}_{-1}$ is the Hessian of f_{hq} on the last $d-1$ coordinates) is upper bounded by

$$\begin{aligned}\frac{h\kappa^2}{4}\left(\left(2\alpha_{-1} - \alpha_{-1}^2\right)\|x_{-1}\|_2^2 - \beta_{-1}^2(d-1)\right) + 5h\kappa^2|\alpha_{-1} - 1|\beta_{-1}\|x_{-1}\|_2\sqrt{\log d} + 3h\kappa^2\beta_{-1}^2\sqrt{d\log d}\\ \le -\frac{h\kappa^2}{4.5}\beta_{-1}^2 d + \frac{h\kappa^2}{4}(2\alpha_{-1} - \alpha_{-1}^2)\|x_{-1}\|_2^2 + 5h\kappa^2|\alpha_{-1} - 1|\beta_{-1}\|x_{-1}\|_2\sqrt{\log d}.\end{aligned} \tag{10.34}$$

Here we dropped the last term in the first line by adjusting a constant since it is dominated for sufficiently large d. It remains to show that all the terms in the second line other than $-\frac{h\kappa^2}{4.5}\beta_{-1}^2 d$ are bounded by $O(h\kappa^2\beta_{-1}^2 d)$. We will perform casework on the size of α_{-1}.

Case 1: $|\alpha_{-1}| > 3$. In this case, we have for sufficiently large d, by Young's inequality

$$\begin{aligned}5h\kappa^2|\alpha_{-1}-1|\beta_{-1}\left\|x_{-1}\right\|_2\sqrt{\log d} &\le \frac{1}{40}h\kappa^2|\alpha_{-1}|\beta_{-1}\left\|x_{-1}\right\|_2\sqrt{d}\\ &\le \frac{1}{80}h\kappa^2\beta_{-1}^2 d + \frac{1}{80}h\kappa^2\alpha_{-1}^2\left\|x_{-1}\right\|_2^2.\end{aligned}$$

Plugging this bound into (10.34), we have the desired

$$\left\|x_{-1}\right\|_{\mathbf{A}_{-1}^2}^2 - \left\|y_{-1}\right\|_{\mathbf{A}_{-1}^2}^2 \le -\frac{h\kappa^2}{5}\beta_{-1}^2 d + \frac{h\kappa^2}{4}\left(2\alpha_{-1} - 0.9\alpha_{-1}^2\right)\left\|x_{-1}\right\|_2^2 \le -\frac{h\kappa^2}{5}\beta_{-1}^2 d.$$

In the last inequality we used $2\alpha_{-1} - 0.9\alpha_{-1}^2 \le 0$ for $|\alpha_{-1}| > 3$.

Case 2: $|\alpha_{-1}| \le 3$. In this case, we first observe by our assumed bounds on $\left\|x_{-1}\right\|_2$ and β_{-1},

$$5h\kappa^2|\alpha_{-1}-1|\beta_{-1}\left\|x_{-1}\right\|_2\sqrt{\log d} \le 20h\kappa^{1.5}\beta_{-1}\sqrt{d\log d} = o\left(h\kappa^2\beta_{-1}^2 d\right).$$

Thus, substituting into (10.34) and dropping the (nonpositive) term corresponding to α_{-1}^2,

$$\begin{aligned}\left\|x_{-1}\right\|_{\mathbf{A}_{-1}^2}^2 - \left\|y_{-1}\right\|_{\mathbf{A}_{-1}^2}^2 &\le -\frac{h\kappa^2}{4.8}\beta_{-1}^2 d + \frac{h\kappa^2}{2}\alpha_{-1}\left\|x_{-1}\right\|_2^2\\ &\le -\frac{h\kappa^2}{4.8}\beta_{-1}^2 d + \frac{h\kappa\alpha_{-1}d}{3} = -\Omega\left(h\kappa^2\beta_{-1}^2 d\right).\end{aligned}$$

In the second inequality, we used the assumed bound on $\left\|x_{-1}\right\|_2^2$, and in the last we used the bound $|\alpha_{-1}| \le \frac{3}{5}\beta_{-1}^2\kappa$ to reach the conclusion.

To complete the proof we need to show terms involving x_1 and g_1 are small. In particular, combining (10.33), Corollary 51, and Lemma 194 and dropping nonnegative terms, it suffices to argue

$$\frac{h}{2}\alpha_1 x_1^2 + 5h|\alpha_1 - 1|\beta_1|x_1|\sqrt{\log d} + 3h\beta_1^2\sqrt{\log d} = o\left(h\kappa^2\beta_{-1}^2 d\right).$$

This bound clearly holds for the last term $h\beta_1^2\sqrt{\log d}$ using $\beta_1 = O(\beta_{-1})$. For the first term, it suffices to use our assumed bounds on $|\alpha_1|$ and x_1. Finally, the middle term $5h|\alpha_1 - 1|\beta_1|x_1|\sqrt{\log d}$ is low-order compared to the term $5h\kappa^2|\alpha_{-1}-1|\beta_{-1}\left\|x_{-1}\right\|_2\sqrt{\log d}$ which we argued about earlier, and hence does not affect any of our earlier bounds by more than a constant. The left-hand side of the above display is an upper bound of the first coordinate's contribution with probability at least $1 - \frac{1}{2}d^{-5}$, so a union bound shows the proof succeeds with probability $\ge 1 - d^{-5}$. □

Finally, we are ready to give the main lower bound of this section.

Theorem 71. *For every step size, there is a target Gaussian on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of MALA is $\Omega(\frac{\kappa\sqrt{d}}{\sqrt{\log d}})$.*

Proof. Let π^* be the Gaussian with log-density $-f_{\mathrm{hq}}$ (10.31) throughout this proof. If $h = O\left(\frac{\sqrt{\log d}}{\kappa\sqrt{d}}\right)$,

then Corollary 50 immediately implies the result, so for the remainder of the proof suppose

$$h = \omega\left(\frac{\sqrt{\log d}}{\kappa\sqrt{d}}\right). \tag{10.35}$$

We first recall that MALA Markov chains are an instance of (10.32) with

$$\alpha_1 = h,\ \alpha_{-1} = h\kappa,\ \beta_1 = \beta_{-1} = \sqrt{2h}.$$

It is easy to see that these parameters satisfy the assumptions in Proposition 49, for the given range of h. We define a "bad starting set" as follows:

$$\Omega := \left\{x \;\middle|\; \|x_{-1}\|_2^2 \le \frac{2d}{3\kappa},\ x_1^2 \le 25\log d\right\}. \tag{10.36}$$

For any $x \in \Omega$, and h satisfying (10.35), Proposition 49 is applicable, and by our definition of Ω, any $x \in \Omega$ has proposals which will be accepted with probability

$$\exp\left(-\Omega\left(h\kappa^2\beta_{-1}^2 d\right)\right) = \exp\left(-\Omega(h^2\kappa^2 d)\right) \le \frac{1}{d^{10}}.$$

The conductance of the Markov chain (10.11) is then at most $\frac{2}{d^5}$ by the witness set Ω and the failure probability of Proposition 49, which concludes the proof by Cheeger's inequality [122], where we use the assumption that $\kappa = O(d^4)$. □

Finally, as it clarifies the required warmness to escape the bad set in the proof of Theorem 71 (and is used in our mixing time bounds in Section 10.9), we lower bound the measure of Ω according to π^*. Applying Lemma 195 shows with probability at least $\exp(-\frac{1}{12}d)$, $\|x_{-1}\|_2^2 \le \frac{d}{2\kappa}$, and Fact 31 shows that $x_1^2 \le 25\log d$ with probability at least $\frac{1}{2}$; combining shows that the measure is at least $\exp(-d)$. We required one helper technical fact, a small-ball probability bound for Gaussians.

Lemma 195. *Let $v \sim \mathcal{N}(0, \mathbf{I})$ be a random Gaussian vector in n dimensions. For large enough n,*

$$\Pr\left[\|v\|_2^2 \le \frac{n}{2}\right] \ge \exp\left(-\frac{n}{12}\right).$$

Proof. Observe that $\|v\|_2^2$ follows a χ^2 distribution with n degrees of freedom. Thus this probability is governed by the χ^2 cumulative density function, and is

$$\frac{1}{\Gamma(k)}\gamma\left(k, ck\right)$$

where we define $k := \frac{n}{2}$ and $c := \frac{1}{2}$; here Γ is the standard gamma function, and γ is the lower

incomplete gamma function. Next, we have the bound from [428]

$$\frac{1}{\Gamma(k)}\gamma\left(k, ck\right) \geq \left(1 - \exp\left(-\ell ck\right)\right)^k, \ \ell := \left(\Gamma(k+1)\right)^{-\frac{1}{k-1}}.$$

A direct calculation yields $\ell \geq \frac{2.5}{k} \implies 1 - \exp(-\ell ck) \geq \exp(-\frac{1}{6})$ for large enough k. Recalling we defined $k = \frac{n}{2}$ yields the conclusion. □

10.8 Lower bound for MALA on well-conditioned distributions

In this section, we derive a lower bound on the relaxation time of MALA for sampling from a distribution with density proportional to $\exp(-f(x))$, where $f : \mathbb{R}^d \to \mathbb{R}$ is a (non-quadratic) target function with condition number κ. In particular, by exploiting the structure of non-cancellations which do not occur for quadratics, we will attain a stronger lower bound.

Our first step is to derive an upper bound on the acceptance probability for a general target function f according to the MALA updates (10.13), analogously to Lemma 193 in the Gaussian case.

Lemma 196. *For any function $f : \mathbb{R}^d \to \mathbb{R}$, we have*

$$\begin{aligned} &f(x) - f(y) + \frac{1}{4h}\left(\|y - (x - h\nabla f(x))\|_2^2 - \|x - (y - h\nabla f(y))\|_2^2\right) \\ &= -f(y) + f(x) - \frac{1}{2}\left\langle x - y, \nabla f(x) + \nabla f(y)\right\rangle + \frac{h}{4}\|\nabla f(x)\|_2^2 - \frac{h}{4}\|\nabla f(y)\|_2^2. \end{aligned}$$

Proof. This is a direct computation which we perform here for completeness:

$$\begin{aligned} &f(x) - f(y) + \frac{1}{4h}\left(\|y - (x - h\nabla f(x))\|_2^2 - \|x - (y - h\nabla f(y))\|_2^2\right) \\ &= f(x) - f(y) + \frac{1}{2h}\left\langle y - x, h\nabla f(x)\right\rangle - \frac{1}{2h}\left\langle x - y, h\nabla f(y)\right\rangle + \frac{h}{4}\|\nabla f(x)\|_2^2 - \frac{h}{4}\|\nabla f(y)\|_2^2 \\ &= -f(y) + f(x) - \frac{1}{2}\left\langle x - y, \nabla f(x) + \nabla f(y)\right\rangle + \frac{h}{4}\|\nabla f(x)\|_2^2 - \frac{h}{4}\|\nabla f(y)\|_2^2. \end{aligned}$$

□

Next, recall the proposal distribution of the MALA updates (10.13) sets $y = x - h\nabla f(x) + \sqrt{2h}g$ where $g \sim \mathcal{N}(0, \mathbf{I})$. We further split this update into a random step and a deterministic step, by defining

$$x_g := x + \sqrt{2h}g, \text{ where } g \sim \mathcal{N}(0, \mathbf{I}) \text{ and } y = x_g - h\nabla f(x). \tag{10.37}$$

This will allow us to reason about the effects of the stochastic and drift terms separately. We

crucially will use the following decomposition of the equation in Lemma 196:

$$\begin{aligned}
&-f(y)+f(x)-\frac{1}{2}\langle x-y,\nabla f(x)+\nabla f(y)\rangle+\frac{h}{4}\|\nabla f(x)\|_2^2-\frac{h}{4}\|\nabla f(y)\|_2^2\\
&\quad=-f(x_g)+f(x)-\frac{1}{2}\langle x-x_g,\nabla f(x)+\nabla f(x_g)\rangle\\
&\quad\quad+f(x_g)-f(y)-\frac{1}{2}\langle x-x_g,\nabla f(y)-\nabla f(x_g)\rangle\\
&\quad\quad-\frac{1}{2}\langle x_g-y,\nabla f(x)+\nabla f(y)\rangle+\frac{h}{4}\|\nabla f(x)\|_2^2-\frac{h}{4}\|\nabla f(y)\|_2^2.
\end{aligned}\tag{10.38}$$

We will use the following observation, which gives an alternate characterization of the second line of (10.38), as well as a bound on the third and fourth lines for smooth functions.

Lemma 197. *For twice-differentiable $f:\mathbb{R}^d\to\mathbb{R}$, letting $x_s := x+s(x_g-x)$ for $s\in[0,1]$, we have*

$$-f(x_g)+f(x)-\frac{1}{2}\langle x-x_g,\nabla f(x)+\nabla f(x_g)\rangle=-2h\int_0^1\left(\frac{1}{2}-s\right)g^\top\nabla^2 f(x_s)g\,ds.$$

Moreover, assuming f is κ-smooth,

$$\begin{aligned}
&f(x_g)-f(y)-\frac{1}{2}\langle x-x_g,\nabla f(y)-\nabla f(x_g)\rangle\\
&-\frac{1}{2}\langle x_g-y,\nabla f(x)+\nabla f(y)\rangle+\frac{h}{4}\|\nabla f(x)\|_2^2+\frac{h}{4}\|\nabla f(y)\|_2^2\\
&\le 2\left(h^2\kappa+h^3\kappa^2\right)\|\nabla f(x)\|_2^2+3\left(h^{1.5}\kappa+h^{2.5}\kappa^2\right)\|g\|_2\|\nabla f(x)\|_2+h^2\kappa^2\|g\|_2^2.
\end{aligned}$$

Proof. By integrating twice and using the definition $x_g = x+\sqrt{2h}g$,

$$\begin{aligned}
f(x_g)&=f(x)+\int_0^1\langle\nabla f(x_s),x_g-x\rangle\,ds\\
&=f(x)+\langle\nabla f(x),x_g-x\rangle+\int_0^1\left\langle\int_0^s\nabla^2 f(x_t)\,(x_g-x)\,dt,x_g-x\right\rangle ds\\
&=f(x)+\langle\nabla f(x),x_g-x\rangle+2h\int_0^1(1-s)\,g^\top\nabla^2 f(x_s)g\,ds.
\end{aligned}\tag{10.39}$$

Similarly,

$$f(x)=f(x_g)+\langle\nabla f(x_g),x-x_g\rangle+2h\int_0^1 s\,g^\top\nabla^2 f(x_s)g\,ds.\tag{10.40}$$

The first conclusion follows from combining (10.39) and (10.40). Next, assuming f is κ-smooth,

$$\begin{aligned}
&f(x_g) - f(y) - \frac{1}{2}\langle x - x_g, \nabla f(y) - \nabla f(x_g)\rangle - \frac{1}{2}\langle x_g - y, \nabla f(x) + \nabla f(y)\rangle \\
&= f(x_g) - f(y) + \frac{\sqrt{2h}}{2}\langle g, \nabla f(y) - \nabla f(x_g)\rangle - \langle x_g - y, \nabla f(y)\rangle - \frac{h}{2}\langle \nabla f(x), \nabla f(x) - \nabla f(y)\rangle \\
&\le f(x_g) - f(y) - \langle x_g - y, \nabla f(y)\rangle + \frac{\sqrt{2h}}{2}\|g\|_2 \|\nabla f(x_g) - \nabla f(y)\|_2 + \frac{h}{2}\|\nabla f(x)\|_2 \|\nabla f(x) - \nabla f(y)\|_2 \\
&\le \frac{\kappa}{2}\|x_g - y\|_2^2 + \frac{\sqrt{2h}\kappa}{2}\|g\|_2 \|x_g - y\|_2 + \frac{h\kappa}{2}\|\nabla f(x)\|_2 \|x - y\|_2 \\
&\le \frac{h^2\kappa}{2}\|\nabla f(x)\|_2^2 + \frac{\sqrt{2}}{2}h^{1.5}\kappa\|g\|_2\|\nabla f(x)\|_2 + \frac{h\kappa}{2}\|\nabla f(x)\|_2\left(\sqrt{2h}\|g\|_2 + h\|\nabla f(x)\|_2\right) \\
&= h^2\kappa\|\nabla f(x)\|_2^2 + \sqrt{2}h^{1.5}\kappa\|g\|_2\|\nabla f(x)\|_2 .
\end{aligned} \tag{10.41}$$

The second line used the definitions of x_g and y in (10.37), and the third used Cauchy-Schwarz. The fourth used smoothness (which implies gradient Lipschitzness), and the fifth again used (10.37) and the triangle inequality. Next, we bound the remaining terms $\frac{h}{4}\|\nabla f(x)\|_2^2 - \frac{h}{4}\|\nabla f(y)\|_2^2$:

$$\begin{aligned}
&\frac{h}{4}\|\nabla f(x)\|_2^2 - \frac{h}{4}\|\nabla f(y)\|_2^2 = \frac{h}{4}\langle \nabla f(x) + \nabla f(y), \nabla f(x) - \nabla f(y)\rangle \\
&\le \frac{h\kappa}{4}\left(2\|\nabla f(x)\|_2 + \kappa\|x - y\|_2\right)\|x - y\|_2 \\
&\le \frac{h\kappa}{4}\left(2\|\nabla f(x)\|_2 + h\kappa\|\nabla f(x)\|_2 + \sqrt{2h}\kappa\|g\|_2\right)\left(h\|\nabla f(x)\|_2 + \sqrt{2h}\|g\|_2\right) \\
&\le \frac{1}{2}\left(h^2\kappa + h^3\kappa^2\right)\|\nabla f(x)\|_2^2 + \frac{\sqrt{2}}{2}\left(h^{1.5}\kappa + h^{2.5}\kappa^2\right)\|g\|_2\|\nabla f(x)\|_2 + h^2\kappa^2\|g\|_2^2 .
\end{aligned} \tag{10.42}$$

Combining (10.41) and (10.42) yields the conclusion. □

We will ultimately use the second bound in Lemma 197 to argue that the third and fourth lines in (10.38) are low-order, so it remains to concentrate on the remaining term,

$$-f(x_g) + f(x) - \frac{1}{2}\langle x - x_g, \nabla f(x) + \nabla f(x_g)\rangle = -2h\int_0^1 \left(\frac{1}{2} - s\right) g^\top \nabla^2 f(x_s) g ds. \tag{10.43}$$

Our goal is to demonstrate this term is $-\Omega(h\kappa d)$ over an inverse-exponentially sized region, for a particular hard distribution. As it is coordinate-wise separable, our proof strategy will be to construct a hard one-dimensional function, and replicate it to obtain a linear dependence on d.

We now define the specific hard function $f_{\text{hard}} : \mathbb{R}^d \to \mathbb{R}$ we work with for the remainder of the section; it is straightforward to see f_{hard} is κ-smooth and 1-strongly convex.

$$f_{\text{hard}}(x) := \sum_{i\in[d]} f_i(x_i), \text{ where } f_i(c) = \begin{cases} \frac{1}{2}c^2 & i = 1 \\ \frac{\kappa}{3}c^2 - \frac{\kappa h}{3}\cos\frac{c}{\sqrt{h}} & 2 \le i \le d \end{cases}. \tag{10.44}$$

We will now show that sampling from the distribution with density proportional to $\exp(-f_{\text{hard}})$ is hard. First, notice that the function f_{hard} has condition number κ and is coordinate-wise separable. It immediately follows from Lemma 192 that the spectral gap (defined in (10.10)) of the MALA Markov chain is governed by the step size h as follows.

Corollary 52. *The spectral gap of the MALA Markov chain for sampling from the density proportional to* $\exp(-f_{\text{hard}})$*, where* f_{hard} *is defined in* (10.44)*, is* $O(h + h^2)$*.*

For the remainder of the section, we focus on upper bounding (10.43) over a large region according to the density proportional to $\exp(-f_{\text{hard}})$. Recall $\{f_i\}_{i\in[d]}$ are the summands of f_{hard}. For a fixed x, consider the random variables S_i^x:

$$S_i^x = -f_i([x_g]_i) + f_i(x_i) - \frac{1}{2}(x_i - [x_g]_i)(f_i'(x_i) + f_i'([x_g]_i)).$$

It is easy to check that for a given realization of g, we have

$$\sum_{i\in[d]} S_i^x = -f(x_g) + f(x) - \frac{1}{2}\left\langle x - x_g, \nabla f(x) + \nabla f(x_g)\right\rangle,$$

where the right-hand side of the above display is the left-hand side of (10.43). We bound the expectation of $\sum_{i\in[d]} S_i^x$, and its deviation from its expectation, in Lemma 198 and Lemma 199 respectively.

Lemma 198. *For any fixed* $x \in \left\{x \;\middle|\; -\frac{1}{2}\pi\sqrt{h} + 2\pi k_i\sqrt{h} \le x_i \le \frac{1}{2}\pi\sqrt{h} + 2\pi k_i\sqrt{h}, k_i \in \mathbb{N}, \forall 2 \le i \le d\right\}$ *and* $h \le 1$*, the random variables* S_i^x*,* $1 \le i \le d$ *satisfy*

$$\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[S_i^x\right] \le \begin{cases} 0 & i = 1 \\ -0.08h\kappa\cos\frac{x_i}{\sqrt{h}} & 2 \le i \le d \end{cases}.$$

Proof. We remark that the condition on x simply enforces coordinatewise in $2 \le i \le d$, $\cos\frac{x_i}{\sqrt{h}} > 0$. Consider some coordinate $2 \le i \le d$: since $[x_g]_i = x_i + \sqrt{2h}g_i$,

$$\begin{aligned}
S_i^x &= -f_i\left(x_i + \sqrt{2h}g_i\right) + f_i(x_i) + \frac{\sqrt{2h}}{2}g_i\left(f_i'(x_i) + f_i'\left(x_i + \sqrt{2h}g_i\right)\right) \\
&= -\frac{\kappa}{3}\left(x_i + \sqrt{2h}g_i\right)^2 + \frac{\kappa h}{3}\cos\left(\frac{x_i}{\sqrt{h}} + \sqrt{2}g_i\right) + \frac{\kappa}{3}x_i^2 - \frac{\kappa h}{3}\cos\left(\frac{x_i}{\sqrt{h}}\right) \\
&\quad + \frac{\sqrt{2h}}{2}g_i\left(\frac{4\kappa}{3}x_i + \frac{2\sqrt{2h}\kappa}{3}g_i + \frac{\kappa\sqrt{h}}{3}\sin\left(\frac{x_i}{\sqrt{h}} + \sqrt{2}g_i\right) + \frac{\kappa\sqrt{h}}{3}\sin\left(\frac{x_i}{\sqrt{h}}\right)\right) \\
&= \frac{\kappa h}{3}\left(\cos\left(\frac{x_i}{\sqrt{h}} + \sqrt{2}g_i\right) - \cos\left(\frac{x_i}{\sqrt{h}}\right)\right) + \frac{\sqrt{2}h\kappa}{6}g_i\left(\sin\left(\frac{x_i}{\sqrt{h}} + \sqrt{2}g_i\right) + \sin\left(\frac{x_i}{\sqrt{h}}\right)\right)
\end{aligned}$$

Here, we used that the quadratic terms in the second and third lines cancel (this also follows from

examining the proof of Lemma 193):

$$-\frac{\kappa}{3}\left(x_i+\sqrt{2h}g_i\right)^2+\frac{\kappa}{3}x_i^2+\frac{\sqrt{2h}}{2}g_i\left(\frac{4\kappa}{3}x_i+\frac{2\sqrt{2h}\kappa}{3}g_i\right)=0.$$

By a direct computation, taking an expectation over $g_i \sim \mathcal{N}(0,1)$ yields

$$\mathbb{E}_{g_i\sim\mathcal{N}(0,1)}\left[\cos\left(\frac{x_i}{\sqrt{h}}+\sqrt{2}g_i\right)\right]=\frac{\cos\left(\frac{x_i}{\sqrt{h}}\right)}{\exp(1)},$$
$$\mathbb{E}_{g_i\sim\mathcal{N}(0,1)}\left[\sin\left(\frac{x_i}{\sqrt{h}}+\sqrt{2}g_i\right)g_i\right]=\frac{\sqrt{2}\cos\left(\frac{x_i}{\sqrt{h}}\right)}{\exp(1)}.$$

Putting these pieces together,

$$\mathbb{E}_{g_i\sim\mathcal{N}(0,1)}\left[S_i^x\right]=\frac{\kappa h}{3}\left(\frac{2}{\exp(1)}-1\right)\cos\left(\frac{x_i}{\sqrt{h}}\right)\quad\leq -0.08\kappa h\cos\left(\frac{x_i}{\sqrt{h}}\right).$$

Here, we used $\cos\frac{x_i}{\sqrt{h}}>0$. For $i=1$, Lemma 193 shows $\mathbb{E}_{g_1\sim\mathcal{N}(0,1)}\left[S_1^x\right]=0$. □

Lemma 199. *With probability at least $1-\frac{1}{d^5}$ over the randomness of $g\sim\mathcal{N}(0,\mathbf{I})$,*

$$\sum_{i\in[d]}S_i^x-\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i\in[d]}S_i^x\right]\leq 10h\kappa\sqrt{d\log d}.$$

Proof. By Lemma 197, for coordinate $1\leq i\leq d$,

$$S_i^x=-2h\int_0^1\left(\frac{1}{2}-s\right)f_i''([x_s]_i)dsg_i^2,\text{ where }\left|2h\int_0^1\left(\frac{1}{2}-s\right)f_i''([x_s]_i)ds\right|\leq\frac{h\kappa}{2}.$$

We attained the latter bound by smoothness. Now, each random variable $S_i^x-\mathbb{E}[S_i^x]$ is sub-exponential with parameter $\frac{h\kappa}{2}$ (for coordinates where the coefficient is negative, note the negation of a sub-exponential random variable is still sub-exponential). Hence, by Fact 33,

$$\Pr\left[\sum_{i\in[d]}S_i^x-\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i\in[d]}S_i^x\right]\geq 10h\kappa\sqrt{d\log d}\right]\leq\frac{1}{d^5}.$$

□

Now, we build a bad set Ω_{hard} with lower bounded measure that starting from a point $x\in\Omega_{\text{hard}}$,

with high probability, $-\mathbb{E}_{g\sim\mathcal{N}(0,\mathbf{I})}\left[\sum_{i\in[d]} S_i^x\right]$ is negative:

$$\Omega_{\text{hard}} = \Bigg\{ x \;\Big|\; |x_1| \le 2, \forall 2 \le i \le d, \exists k_i \in \mathbb{Z}, |k_i| \le \lfloor \frac{5}{\pi\sqrt{h\kappa}} \rfloor, \text{ such that } \\ -\frac{9}{20}\pi\sqrt{h} + 2\pi k_i\sqrt{h} \le x_i \le \frac{9}{20}\pi\sqrt{h} + 2\pi k_i\sqrt{h} \Bigg\}. \tag{10.45}$$

In other words, Ω_{hard} is the set of points where $\cos x_i$ is large for $2 \le i \le d$, and coordinates are bounded. We first lower bound the measure of Ω_{hard}, and show $\|\nabla f(x)\|_2$ is small within Ω_{hard}. Our measure lower bound will not be used in this section, but will become relevant in Section 10.9.

Lemma 200. *Let $h \le \frac{1}{10000\pi^2\kappa}$. Let π^* have log-density $-f_{\text{hard}}$ (10.44). Then, $\pi^*(\Omega_{\text{hard}}) \ge \exp(-d)$. Moreover, for all $x \in \Omega_{\text{hard}}$, $\|\nabla f(x)\|_2 \le 10\sqrt{\kappa d}$.*

Proof. We first consider a superset of Ω_{hard}. We define the set, for $K := \lfloor \frac{5}{\pi\sqrt{h\kappa}} \rfloor$,

$$\Omega' = \left\{ x \;\Big|\; |x_1| \le 2, \forall 2 \le i \le d, -\frac{9}{20}\pi\sqrt{h} - 2\pi K\sqrt{h} \le x_i \le \frac{9}{20}\pi\sqrt{h} + 2\pi K\sqrt{h} \right\}.$$

It is easy to verify that $\Omega' \supseteq \Omega_{\text{hard}}$. We first show $\pi^*(\Omega')$ is lower bounded by 1.1^{-d}. Since f_{hard} is separable, the coordinates are independent, so it suffices to show each one-dimensional measure is lower bounded by $\frac{1}{1.1}$. This is a standard computation of Gaussian measure for the first coordinate, which we omit. For $2 \le i \le d$, since the marginal distribution is $\frac{\kappa}{3}$-strongly logconcave, it is sub-Gaussian with parameter $\frac{3}{\kappa}$ (see Lemma 1, [210]). It follows from a standard sub-Gaussian tail bound that the measure of the set $|x_i| \le \frac{9}{\sqrt{\kappa}}$ is at least $\frac{1}{1.1}$. For our choice of K, by assumption on h, $2\pi\sqrt{h}K \ge \frac{10}{\sqrt{\kappa}} - 2\pi\sqrt{h} \ge \frac{9}{\sqrt{\kappa}}$. Combining across coordinates gives $\pi^*(\Omega') \ge 1.1^{-d}$.

Next, we lower bound $\frac{\pi^*(\Omega_{\text{hard}})}{\pi^*(\Omega')}$. We divide the support of the set Ω_{hard} and Ω' into small disjoint regions and bound $\frac{\pi^*(\Omega_{\text{hard}})}{\pi^*(\Omega')}$ for each small region and each coordinate separately. For $2 \le i \le d$, $k \in \left[-\lfloor \frac{5}{\pi\sqrt{h\kappa}} \rfloor - 1, \lfloor \frac{5}{\pi\sqrt{h\kappa}} \rfloor\right]$, $k \in \mathbb{Z}$, let

$$\Omega'^{(i,k)} = \left(2\pi k\sqrt{h}, 2\pi(k+1)\sqrt{h}\right],$$

and

$$\Omega_{\text{hard}}^{(i,k)} = \left(2\pi k\sqrt{h}, 2\pi k\sqrt{h} + \frac{9}{20}\pi\sqrt{h}\right] \cup \left[2\pi(k+1)\sqrt{h} - \frac{9}{20}\pi\sqrt{h}, 2\pi(k+1)\sqrt{h}\right].$$

Then, letting π_i^* be the marginal of π^* on coordinate i, we have

$$\begin{aligned}\frac{\pi_i^*\left(\Omega_{\text{hard}}^{(i,k)}\right)}{\pi_i^*\left(\Omega'^{(i,k)}\right)} &= \frac{\int_{2\pi k\sqrt{h}}^{2\pi k\sqrt{h}+\frac{9}{20}\pi\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2+\frac{\kappa h}{3}\cos\frac{x_i}{\sqrt{h}}\right)dx_i + \int_{2\pi(k+1)\sqrt{h}-\frac{9}{20}\pi\sqrt{h}}^{2\pi(k+1)\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2+\frac{\kappa h}{3}\cos\frac{x_i}{\sqrt{h}}\right)dx_i}{\int_{2\pi k\sqrt{h}}^{2\pi(k+1)\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2+\frac{\kappa h}{3}\cos\frac{x_i}{\sqrt{h}}\right)dx_i} \\ &\geq \frac{\int_{2\pi k\sqrt{h}}^{2\pi k\sqrt{h}+\frac{9}{20}\pi\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2\right)dx_i + \int_{2\pi(k+1)\sqrt{h}-\frac{9}{20}\pi\sqrt{h}}^{2\pi(k+1)\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2\right)dx_i}{\int_{2\pi k\sqrt{h}}^{2\pi(k+1)\sqrt{h}} \exp\left(-\frac{\kappa}{3}x_i^2\right)dx_i \exp\left(\frac{\kappa h}{3}\right)} \\ &\geq \exp\left(-\frac{\kappa h}{3}\right)\cdot\frac{\frac{9}{10}\pi\sqrt{h}}{2\pi\sqrt{h}}\cdot\frac{\exp\left(-\frac{\kappa}{3}\left(2\pi(k+1)\sqrt{h}\right)^2\right)}{\exp\left(-\frac{\kappa}{3}\left(2\pi k\sqrt{h}\right)^2\right)} \geq 0.42.\end{aligned}$$

The second step used $\cos\frac{x_i}{\sqrt{h}} \geq 0$ for $x_i \in \Omega_{\text{hard}}^{(i,k)}$. The fourth step used the assumption $\kappa h \leq \frac{1}{10000\pi^2}$.

Finally, letting $\Omega'^{(i)}$ and $\Omega_{\text{hard}}^{(i)}$ be the projections of Ω' and Ω_{hard} on the i^{th} coordinate. For any $x_i \in \Omega_i'$ with $x \notin \Omega'^{(i,k)}$, and for all integers $k \in [-K-1, K]$, $x_i \in \Omega_{\text{hard}}^{(i)}$, so $\frac{\pi^*\left(\Omega_{\text{hard}}^{(i)}\right)}{\pi^*\left(\Omega'^{(i)}\right)} \geq 0.42$. Since the coordinates are independent under π^*, $\frac{\pi^*(\Omega_{\text{hard}})}{\pi^*(\Omega')} \geq 0.42^d$. Combining our lower bounds,

$$\pi^*\left(\Omega_{\text{hard}}\right) = \pi^*(\Omega')\frac{\pi^*(\Omega_{\text{hard}})}{\pi^*(\Omega')} \geq \left(\frac{1.1}{0.42}\right)^{-d} \geq \exp\left(-d\right).$$

Finally, we bound $\left\|\nabla f_{\text{hard}}(x)\right\|_2$ for $x \in \Omega'$, from the definition of f_{hard} (10.44),

$$\left\|\nabla f_{\text{hard}}(x)\right\|_2 = \sqrt{f_1'(x)^2 + \sum_{i=2}^d f_i'(x)^2} = \sqrt{x_1^2 + \sum_{i=2}^d \left(\frac{2\kappa}{3}x_i + \frac{\kappa\sqrt{h}}{3}\sin\frac{x_i}{\sqrt{h}}\right)^2}.$$

Then directly plugging in the definition of Ω_{hard} and using $|\sin c| \leq |c|$ for all c,

$$\left\|\nabla f_{\text{hard}}(x)\right\|_2 \leq \sqrt{1.5^2 + (d-1)\left(9\sqrt{\kappa}\right)^2} \leq 10\sqrt{\kappa d}.$$

□

Finally, we combine the bounds we derived to show the acceptance probability is small within Ω_{hard}.

Lemma 201. *Let $h = o\left(\frac{1}{\kappa \log d}\right)$. For any $x \in \Omega_{\text{hard}}$, let $y = x - h\nabla f_{\text{hard}}(x) + \sqrt{2h}g$ for $g \sim \mathcal{N}(0, \mathbf{I})$. With probability at least $1 - \frac{2}{d^5}$, we have*

$$f_{\text{hard}}(x) - f_{\text{hard}}(y) + \frac{1}{4h}\left(\left\|y - (x - h\nabla f_{\text{hard}}(x))\right\|_2^2 - \left\|x - (y - h\nabla f_{\text{hard}}(y))\right\|_2^2\right) = -\Omega\left(h\kappa d\right).$$

Proof. By combining Lemma 196 and the decomposition (10.38), the conclusion is equivalent to

showing that the following quantity is $-\Omega(h\kappa d)$:

$$\begin{aligned}
&-f_{\text{hard}}(x_g) + f_{\text{hard}}(x) - \frac{1}{2}\left\langle x - x_g, \nabla f_{\text{hard}}(x) + \nabla f_{\text{hard}}(x_g)\right\rangle \\
&+f_{\text{hard}}(x_g) - f_{\text{hard}}(y) - \frac{1}{2}\left\langle x - x_g, \nabla f_{\text{hard}}(y) - \nabla f_{\text{hard}}(x_g)\right\rangle \\
&-\frac{1}{2}\left\langle x_g - y, \nabla f_{\text{hard}}(x) + \nabla f_{\text{hard}}(y)\right\rangle + \frac{h}{4}\left\|\nabla f_{\text{hard}}(x)\right\|_2^2 - \frac{h}{4}\left\|\nabla f_{\text{hard}}(y)\right\|_2^2.
\end{aligned}$$

For $x \in \Omega_{\text{hard}}$, every x_i for $2 \le i \le d$ has $\cos\frac{x_i}{\sqrt{h}}$ bounded away from 0 by a constant and hence combining Lemmas 198 and 199 implies the first line is $-\Omega(h\kappa d)$ with probability at least $\frac{1}{d^5}$. Regarding the second and third lines, Lemma 197 shows that it suffices to bound (over the set Ω_{hard})

$$(h^2\kappa + h^3\kappa^2)\left\|\nabla f_{\text{hard}}(x)\right\|_2^2 + \left(h^{1.5}\kappa + h^{2.5}\kappa^2\right)\|g\|_2\left\|\nabla f_{\text{hard}}(x)\right\|_2 + h^2\kappa^2\|g\|_2^2 = o(h\kappa d).$$

Fact 32 implies $\|g\|_2 \le \sqrt{2d}$ with probability at least $1 - \frac{1}{d^5}$. Combining this bound, the bound on $\left\|\nabla f_{\text{hard}}(x)\right\|_2$ from Lemma 200, and the upper bound on h yields the conclusion. □

We conclude by giving the main result of this section.

Proposition 50. *For $h = o(\frac{1}{\kappa \log d})$, there is a target density on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of MALA is $\Omega(\frac{\kappa d}{\log d})$.*

Proof. The proof is identical to that of Theorem 71, where we use Lemma 201 in place of Proposition 49. □

Theorem 72. *For every step size, there is a target density on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that the relaxation time of MALA is $\Omega(\frac{\kappa d}{\log d})$.*

Proof. This is immediate from combining Theorem 71 (with the hard function f_{hq} in the range $h = \Omega(\frac{1}{\kappa \log d})$) and Proposition 50 (with the hard function f_{hard} in the range $h = o(\frac{1}{\kappa \log d})$). □

10.9 Mixing time lower bound for MALA

In this section, we derive a mixing time lower bound for MALA. Concretely, we show that for any step size h, there is a hard distribution $\pi^* \propto \exp(-f)$ such that $\nabla^2 f$ always has eigenvalues in $[1, \kappa]$, yet there is a $\exp(d)$-warm start π_0 such that the chain cannot mix in $o(\frac{\kappa d}{\log^2 d})$ iterations, starting from π_0. We begin by giving such a result for $h = O(\frac{\log d}{\kappa d})$ in Section 10.9.1, and combine it with our developments in Sections 10.7 and 10.8 to prove our main mixing time result.

10.9.1 Mixing time lower bound for small h

Throughout this section, let $h = O\left(\frac{\log d}{\kappa d}\right)$, and let $\pi^* = \mathcal{N}(0, \mathbf{I})$ be the standard d-dimensional multivariate Gaussian. We will let π_0 be the marginal distribution of π^* on the set

$$\Omega := \left\{ x \mid \|x\|_2^2 \le \frac{1}{2} d \right\}.$$

Recall from Lemma 195 that π_0 is a $\exp(d)$-warm start. Our main proof strategy will be to show that for such a small value of h, after $T = O(\frac{\kappa d}{\log^2 d})$ iterations, with constant probability both of the following events happen: no rejections occur throughout the Markov chain, and $\|x_t\|_2^2 \le \frac{9}{10} d$ holds for all $t \in [T]$. Combining these two facts will demonstrate our total variation lower bound.

Lemma 202. *Let $\{x_t\}_{0 \le t < T}$ be the iterates of the MALA Markov chain with step size $h = O\left(\frac{\log d}{\kappa d}\right)$, for $T = o(\frac{\kappa d}{\log^2 d})$ and $x_0 \sim \pi_0$. With probability at least $\frac{99}{100}$, both of the following events occur:*

1. *Throughout the Markov chain, $\|x_t\|_2 \le 0.9\sqrt{d}$.*
2. *Throughout the Markov chain, the Metropolis filter never rejected.*

Proof. We inductively bound the failure probability of the above events in every iteration by $\frac{0.01}{T}$, which will yield the claim via a union bound. Take some iteration $t+1$, and note that by triangle inequality, and assuming all prior iterations did not reject,

$$\|x_{t+1}\|_2 \le \|x_0\|_2 + h \sum_{s=0}^{t} \|x_s\|_2 + \sqrt{2h} \left\| \sum_{s=0}^{t} g_s \right\|_2 \le \|x_0\|_2 + 0.9hT\sqrt{d} + \sqrt{2h}\, \|G_t\|_2 \le 0.8\sqrt{d} + \sqrt{2h}\, \|G_t\|_2 .$$

Here, we applied the inductive hypothesis on all $\|x_s\|_2$, the initial bound $\|x_0\|_2 \le \sqrt{\frac{1}{2}d}$, and that $hT = o(1)$ by assumption. We also defined $G_t = \sum_{s=0}^{t} g_s$, where g_s is the random Gaussian used by MALA in iteration s; note that by independence, $G_t \sim \mathcal{N}(0, t+1)$. By Fact 32, with probability at least $\frac{1}{200T}$, $\|G_t\|_2 \le 2\sqrt{Td}$, and hence $0.8\sqrt{d} + \sqrt{2h}\, \|G_t\|_2 \le 0.9\sqrt{d}$, as desired.

Next, we prove that with probability $\ge 1 - \frac{1}{200T}$, step t does not reject. This concludes the proof by union bounding over both events in iteration t, and then union bounding over all iterations. By the calculation in Lemma 193, the accept probability is

$$\min\left(1, \exp\left(\frac{h}{4}\left(\left(2h - h^2\right)\|x_t\|_2^2 - 2h\,\|g\|_2^2 - 2\sqrt{2h}\,(1-h)\,\langle x_t, g\rangle\right)\right)\right).$$

We lower bound the argument of the exponential as follows. With probability at least $1 - d^{-5} \ge 1 - \frac{1}{400T}$, Facts 31 and 32 imply both of the events $\|g\|_2^2 \le 2d$ and $\langle x_t, g\rangle \le 10\sqrt{\log d}\,\|x\|_2$ occur. Conditional on these bounds, we compute (using $2h \ge h^2$ and the assumption $\|x_t\|_2 \le 0.9\sqrt{d}$)

$$\left(2h - h^2\right)\|x_t\|_2^2 - 2h\,\|g\|_2^2 - 2\sqrt{2h}\,(1-h)\,\langle x_t, g\rangle \ge -4hd - 40\sqrt{hd\log d} \ge -44\log d.$$

Hence, the acceptance probability is at least

$$\exp\left(-11h\log d\right) \geq 1 - \frac{1}{400T},$$

by our choice of T with $Th\log d = o(1)$, concluding the proof. □

Proposition 51. *The MALA Markov chain with step size $h = O\left(\frac{\log d}{\kappa d}\right)$ requires $\Omega(\frac{\kappa d}{\log^2 d})$ iterations to reach total variation distance $\frac{1}{e}$ to π^*, starting from π_0.*

Proof. Let $\tilde{\pi}$ be the distribution of the MALA Markov chain after $T = o(\frac{\kappa d}{\log^2 d})$ steps without applying a Metropolis filter in any step, and let $\hat{\pi}$ be the distribution after applying the actual MALA chain (including rejections). To show $\|\hat{\pi} - \pi^*\|_{\mathrm{TV}} \geq \frac{1}{e}$, it suffices to show the bounds

$$\|\tilde{\pi} - \pi^*\|_{\mathrm{TV}} \geq \frac{2}{5},\ \|\tilde{\pi} - \hat{\pi}\|_{\mathrm{TV}} \leq 0.01,$$

and then we apply the triangle inequality. By the coupling characterization of total variation, the second bound follows immediately from the second claim in Lemma 202, wherein we couple the two distributions whenever a rejection does not occur. To show the first bound, the measure of

$$\Omega_{\mathrm{large}} := \left\{x \mid \|x\|_2^2 \geq 0.81d\right\}$$

according to π^* is at least 0.99 by Fact 32, and according to $\tilde{\pi}$ it can be at most 0.01 by the first conclusion of Lemma 202. This yields the bound via the definition of total variation. □

10.9.2 Proof of Theorem 73

Finally, we put together the techniques of Sections 10.7, 10.8, and 10.9.1 to prove Theorem 73.

Theorem 73. *For every step size, there is a target density on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1, \kappa]$, such that MALA initialized at an $\exp(d)$-warm start requires $\Omega(\frac{\kappa d}{\log^2 d})$ iterations to reach e^{-1} total variation distance to the stationary distribution.*

Proof. We consider three ranges of h. First, if $h = \Omega\left(\frac{1}{\kappa\log d}\right)$, we use the hard function f_{hq} and the hard set in (10.36), which has measure at least $\exp(-d)$ according to the stationary distribution by Lemma 195. Then, applying Proposition 49 demonstrates that the chance the Markov chain can move over d^5 iterations is $o(\frac{1}{d})$, and hence it does not reach total variation $\frac{1}{e}$ in this time. Next, if $h = o\left(\frac{1}{\kappa\log d}\right) \cap \omega\left(\frac{\log d}{\kappa d}\right)$, we use the hard function f_{hard} and the hard set in (10.45), which has measure at least $\exp(-d)$ by Lemma 200. Applying Lemma 201 again implies the chain does not mix in d^5 iterations. Finally, if $h = O\left(\frac{\log d}{\kappa d}\right)$, applying Proposition 51 yields the claim. □

10.10 Lower bounds for HMC

In this section, we derive a lower bound on the spectral gap of HMC. We first analyze some general structural properties of HMC in Section 10.10.1, as a prelude to later sections. We then provide a lower bound for HMC on quadratics in Section 10.10.2, with any number of leapfrog steps K.

10.10.1 Structure of HMC: a detour to Chebyshev polynomials

We begin our development with a bound on the acceptance probability for general HMC Markov chains. Recall from (10.14) that this probability is (for $\mathcal{H}(x,v) := f(x) + \frac{1}{2}\|v\|_2^2$)

$$\min\left\{1, \frac{\exp\left(-\mathcal{H}(x_K, v_K)\right)}{\exp\left(-\mathcal{H}(x_0, v_0)\right)}\right\}. \tag{10.14}$$

We first state a helper calculation straightforwardly derived from the exposition in Section 10.2.4.

Fact 34. *One step of the HMC Markov chain starting from x_0 generates iterates $\{(v_{k-\frac{1}{2}}, x_k, v_k)\}_{0\le k\le K}$ defined recursively by the closed-form equations:*

$$\begin{aligned}
v_{k-\frac{1}{2}} &= v_0 - \frac{\eta}{2}\nabla f(x_0) - \eta \sum_{j\in[k-1]} \nabla f(x_j),\\
v_k &= v_0 - \frac{\eta}{2}\nabla f(x_0) - \eta \sum_{j\in[k-1]} \nabla f(x_j) - \frac{\eta}{2}\nabla f(x_k),\\
x_k &= x_0 + \eta k v_0 - \frac{\eta^2 k}{2}\nabla f(x_0) - \eta^2 \sum_{j\in[k-1]} (k-j)\nabla f(x_j).
\end{aligned}$$

When expanding the acceptance probability (10.14) using the equations in Fact 34, many terms conveniently cancel, which we capture in Lemma 203. This phenomenon underlies the improved performance of HMC on densities with highly-Lipschitz Hessians [128].

Lemma 203. *For the iterates given by Fact 34,*

$$\begin{aligned}
\mathcal{H}\left(x_0, v_0\right) - \mathcal{H}\left(x_K, v_K\right) = \sum_{0\le k\le K-1} \left(f(x_k) - f(x_{k+1}) + \frac{1}{2}\left\langle \nabla f(x_k) + \nabla f(x_{k+1}), x_{k+1} - x_k\right\rangle\right)\\
+ \frac{\eta^2}{8}\left\|\nabla f(x_0)\right\|_2^2 - \frac{\eta^2}{8}\left\|\nabla f(x_K)\right\|_2^2.
\end{aligned}$$

Proof. Recall $\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) = f(x_0) - f(x_K) + \frac{1}{2}\|v_0\|_2^2 - \frac{1}{2}\|v_K\|_2^2$. We begin by expanding

$$\begin{aligned}
\frac{1}{2}\|v_0\|_2^2 - \frac{1}{2}\|v_K\|_2^2 &= \frac{1}{2}\|v_0\|_2^2 - \frac{1}{2}\left\|v_0 - \frac{\eta}{2}\nabla f(x_0) - \eta\sum_{j\in[K-1]}\nabla f(x_j) - \frac{\eta}{2}\nabla f(x_K)\right\|_2^2 \\
&= \eta\left\langle v_0, \frac{1}{2}\nabla f(x_0) + \sum_{j\in[K-1]}\nabla f(x_j) + \frac{1}{2}\nabla f(x_K)\right\rangle \\
&\quad - \frac{\eta^2}{2}\left\|\frac{1}{2}\nabla f(x_0) + \sum_{j\in[K-1]}\nabla f(x_j) + \frac{1}{2}\nabla f(x_K)\right\|_2^2 \\
&= \frac{\eta}{2}\sum_{0\le k\le K-1}\langle v_0, \nabla f(x_k) + \nabla f(x_{k+1})\rangle \\
&\quad - \frac{\eta^2}{2}\sum_{0\le k\le K-1}\left\langle \nabla f(x_k) + \nabla f(x_{k+1}), \frac{1}{2}\nabla f(x_0) + \sum_{j\in[k]}\nabla f(x_j)\right\rangle \\
&\quad + \frac{\eta^2}{8}\langle \nabla f(x_0) - \nabla f(x_K), \nabla f(x_0) + \nabla f(x_K)\rangle\,.
\end{aligned}$$

Here the first equality used Fact 34. Moreover, for each $0 \le k \le K-1$, by Fact 34

$$\begin{aligned}
\frac{1}{2}\langle \nabla f(x_k) + \nabla f(x_{k+1}), x_{k+1} - x_k\rangle &= \frac{\eta}{2}\langle \nabla f(x_k) + \nabla f(x_{k+1}), v_0\rangle \\
&\quad - \frac{\eta^2}{2}\left\langle \nabla f(x_k) + \nabla f(x_{k+1}), \frac{1}{2}\nabla f(x_0) + \sum_{j\in[k]}\nabla f(x_j)\right\rangle.
\end{aligned}$$

Combining yields the result. □

We state a simple corollary of Lemma 203 in the case of quadratics.

Corollary 53. *For $f(x) = \frac{1}{2}x^\top \mathbf{A}x$, the iterates given by Fact 34 satisfy*

$$\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) = \frac{\eta^2}{8}\|\nabla f(x_0)\|_2^2 - \frac{\eta^2}{8}\|\nabla f(x_K)\|_2^2\,.$$

Proof. It suffices to observe that for any two points $x, y \in \mathbb{R}^d$,

$$f(x) - f(y) + \frac{1}{2}\langle \nabla f(x) + \nabla f(y), y - x\rangle = \frac{1}{2}x^\top \mathbf{A}x - \frac{1}{2}y^\top \mathbf{A}y + \frac{1}{2}\langle \mathbf{A}(x+y), y - x\rangle = 0.$$

□

Finally, it will be convenient to have a more explicit form of iterates in the case of quadratics, which follows directly from examining the recursion in Fact 34.

Lemma 204. *For $f(x) = \frac{1}{2}x^\top \mathbf{A}x$, the iterates $\{x_k\}_{0\le k\le K}$ given by Fact 34 satisfy*

$$\begin{aligned} x_k &= \left(\sum_{0\le j\le k} D_{j,k}(\eta^2\mathbf{A})^j\right)x_0 + \left(\eta\sum_{0\le j\le k-1} E_{j,k}(\eta^2\mathbf{A})^j\right)v_0, \\ \text{where } D_{j,k} &:= (-1)^j\cdot\frac{k}{k+j}\cdot\binom{k+j}{2j},\ E_{j,k} := (-1)^j\cdot\binom{k+j}{2j+1}. \end{aligned} \tag{10.46}$$

Proof. This formula can be verified to match the recursions of Fact 34 by checking the base cases $D_{0,k} = 1$, $D_{1,k} = -\frac{k^2}{2}$, $E_{0,k} = k$, and (where $D_{j,k} := 0$ for $j > k$ and $E_{j,k} := 0$ for $j \ge k$)

$$D_{j,k} = -\sum_{i\in[k-1]}(k-i)D_{j-1,i},\ E_{j,k} = -\sum_{i\in[k-1]}(k-i)E_{j-1,i}.$$

In particular, by using the third displayed line of Fact 34, the coefficient of $(\eta^2\mathbf{A})^j x_0$ in x_k for $j \ge 2$ is the negated sum of the coefficients of $(\eta^2\mathbf{A})^{j-1}$ in all $(k-i)x_i$. Similarly, the coefficient of $\eta(\eta^2\mathbf{A})^j v_0$ in x_k for $j \ge 1$ is the negated sum of the coefficients of $\eta(\eta^2\mathbf{A})^{j-1}$ in all $(k-i)x_i$. The displayed coefficient identities follow from the binomial coefficient identities

$$\frac{k}{k+j}\binom{k+j}{2j} = \sum_{j-1\le i\le k-1}\frac{(k-i)i}{i+j-1}\binom{i+j-1}{2j-2},\ \binom{k+j}{2j+1} = \sum_{j\le i\le k-1}(k-i)\binom{i+j-1}{2j-1}.$$

□

Lemma 204 motivates the definition of the polynomials

$$p_k(z) := \sum_{0\le j\le k} D_{j,k}z^j,\ q_k(z) := \sum_{0\le j\le k-1} E_{j,k}z^j. \tag{10.47}$$

In this way, at least in the case when $\mathbf{A} = \mathbf{diag}(\lambda)$ for a vector of eigenvalues $\lambda \in \mathbb{R}^d$, we can concisely express the coordinates of iterates in (10.46) by

$$[x_k]_i = p_k(\eta^2\lambda_i)[x_0]_i + \eta q_k(\eta^2\lambda_i)[v_0]_i. \tag{10.48}$$

Interestingly, the polynomial p_k turns out to have a close relationship with the k^{th} *Chebyshev polynomial* (of the first kind), which we denote by T_k. Similarly, the polynomial q_k is closely related to the $(k-1)^{\text{th}}$ *Chebyshev polynomial* of the second kind, denoted U_{k-1}. The relationship between the Chebyshev polynomials and the phenomenon of *acceleration* for optimizing quadratics via first-order methods has been known for some time (see e.g. [266, 49] for discussions), and we find it interesting to further explore this relationship. Concretely, the following identities hold.

Lemma 205. *Following definitions* (10.46), (10.47),

$$p_k(z) = T_k\left(1 - \frac{z}{2}\right),\ q_k(z) = U_{k-1}\left(1 - \frac{z}{2}\right).$$

Proof. It is easy to check $p_0(z) = 1$ and $p_1(z) = 1 - \frac{z}{2}$, so the former conclusion would follow from

$$p_{k+1}(z) = (2-z)p_k(z) - p_{k-1}(z) \iff D_{j,k+1} = 2D_{j,k} - D_{j-1,k} - D_{j,k-1},$$

following well-known recursions defining the Chebyshev polynomials of the first kind. This identity can be verified by direct expansion. Moreover, for the latter conclusion, recalling the definition of Morgan-Voyce polynomials of the first kind $B_k(z)$, we can directly match $q_k(z) = B_{k-1}(-z)$. The conclusion follows from Section 4 of [30], which shows $B_{k-1}(-z) = U_{k-1}(1 - \frac{z}{2})$ as desired (note that in the work [30], the indexing of Chebyshev polynomials is off by one from ours). □

Now for $z = \eta^2\lambda_i$, we have from (10.48) and Lemma 205 that $[x_k]_i = \pm[x_0]_i$ precisely when

$$p_k(z) = T_k\left(1 - \frac{z}{2}\right) = \pm 1,\ q_k(z) = U_{k-1}\left(1 - \frac{z}{2}\right) = 0.$$

Hence, this occurs whenever $1 - \frac{z}{2}$ is both an *extremal point* of T_k in the range $[-1, 1]$ and a root of U_{k-1}. Both of these occur exactly at the points $\cos(\frac{j}{k}\pi)$, for $0 \le j \le k$.

Proposition 52. *For $\kappa \ge \pi^2$ and $K \ge 2$, no K-step HMC Markov chain with step size $1 \ge \eta^2 \ge \frac{\pi^2}{\kappa K^2}$ can mix in finite time for all densities on $\mathbb{R}^d$ whose negative log-density's Hessian has eigenvalues between 1 and κ for all points $x \in \mathbb{R}^d$, initialized at a constant-warm start.*

Proof. Fix a value of $1 \ge \eta \ge \sqrt{\frac{\pi^2}{\kappa K^2}}$. We claim there exists a $1 \le j \le K-1$ such that for

$$\lambda := \frac{2\left(1 - \cos\left(\frac{j\pi}{K}\right)\right)}{\eta^2},\ 1 \le \lambda \le \kappa.$$

Since λ is a monotone function of η, it suffices to check the endpoints of the interval $[\frac{\pi^2}{\kappa K^2}, 1]$. For $\eta^2 = 1$, we choose $j = K - 1$, which using $\frac{2x^2}{\pi^2} \le 1 - \cos(x) \le \frac{x^2}{2}$ for all $-\pi \le x \le \pi$, yields

$$1 \le \frac{4(K-1)^2}{K^2} \le \lambda \le \frac{(K-1)^2\pi^2}{K^2} \le \pi^2 \le \kappa.$$

Similarly, for $\eta^2 = \frac{\pi^2}{\kappa K^2}$, we choose $j = 1$, which yields

$$1 \le \frac{4}{\eta^2 K^2} \le \lambda \le \frac{\pi^2}{\eta^2 K^2} \le \kappa.$$

Now, consider the quadratic $f(x) = \frac{1}{2}x^\top \mathbf{A}x$ where $\mathbf{A} \in \mathbb{R}^{d\times d}$ is a diagonal matrix, $\mathbf{A}_{11} = 1$, $\mathbf{A}_{ii} = \kappa$ for all $3 \le i \le d$, and $\mathbf{A}_{22} = \lambda := \frac{2\left(1-\cos\left(\frac{j\pi}{K}\right)\right)}{\eta^2}$ for the choice of j which makes $1 \le \lambda \le \kappa$. For

any symmetric starting set capturing a constant amount of measure along the second coordinate, by Lemma 204 and the following exposition, $x_K = \pm x_0$ along the second coordinate regardless of the random choice of velocity and thus the chain cannot leave the starting set. □

10.10.2 HMC lower bound for all K

We now give our HMC lower bound, via improving Proposition 52 by a dimension dependence. We begin by giving a stronger upper bound on η in the range $\eta^2 \le \frac{1}{\kappa K^2}$. Noting that there is a constant-sized gap between this range and the bound in Proposition 52, we next rule out this gap. Finally, we handle the case of extremely large $\eta^2 \ge 1$. We put these pieces together to prove Theorem 74.

Upper bounding $\eta = O(K^{-1}\kappa^{-\frac{1}{2}})$ under a constant gap

For this section, we let $\mathbf{A}$ be the $d \times d$ diagonal matrix which induces the hard quadratic function f_{hq}, defined in (10.31) and reproduced here for convenience:

$$f_{\mathrm{hq}}(x) := \sum_{i \in [d]} f_i(x_i), \text{ where } f_i(c) = \begin{cases} \frac{1}{2}c^2 & i = 1 \\ \frac{\kappa}{2}c^2 & 2 \le i \le d \end{cases}.$$

We also let $h := \frac{\eta^2}{2}$, $x := x_0$, $g := v_0$, and $y := x_K$ throughout for analogy to Section 10.7, so that we can apply Proposition 49. Next, note that by the closed-form expression given by Lemma 204, we can write the iterates of the HMC chain in the form (10.32), reproduced here:

$$y = \begin{pmatrix} y_1 \\ y_{-1} \end{pmatrix}, \text{ where } y_1 = (1-\alpha_1)x_1 + \beta_1 g_1$$
$$\text{and } y_{-1} = (1-\alpha_{-1})x_{-1} + \beta_{-1} g_{-1}, \text{ for } g \sim \mathcal{N}(0, \mathbf{I}).$$

Concretely, we have by Lemma 204 that

$$\begin{aligned} \alpha_1 &= -\sum_{1 \le j \le K} (-1)^j (2h)^j \left(\frac{K}{K+j}\right)\binom{K+j}{2j}, \\ \alpha_{-1} &= -\sum_{1 \le j \le K} (-1)^j (2h\kappa)^j \left(\frac{K}{K+j}\right)\binom{K+j}{2j}, \\ \beta_1 &= \sqrt{2h} \sum_{0 \le j \le K-1} (-1)^j (2h)^j \binom{K+j}{2j+1}, \\ \beta_{-1} &= \sqrt{2h} \sum_{0 \le j \le K-1} (-1)^j (2h\kappa)^j \binom{K+j}{2j+1}. \end{aligned} \tag{10.49}$$

By a straightforward computation, the parameters in (10.49) satisfy the conditions of Proposition 49.

Lemma 206. *Supposing* $\eta^2 \le \frac{1}{\kappa K^2}$, α_1, α_{-1}, β_1, β_{-1} *defined in* (10.49) *satisfy*

$$|\alpha_{-1}| \le \frac{3}{5}\beta_{-1}^2\kappa,\ |\alpha_1| = O(|\alpha_{-1}|),\ \beta_1 = O(\beta_{-1}).$$

Proof. The proof follows since under $\eta^2 \le \frac{1}{10\kappa K^2}$, all of the parameters in (10.49) are dominated by their first summand. We will argue this for α_{-1} and β_{-1}; the corresponding conclusions for α_1 and β_1 follow analogously since $\kappa \ge 1$. Define the summands of α_{-1} and β_{-1} by

$$c_j := (-1)^{j+1}(2h\kappa)^j \left(\frac{K}{K+j}\right)\binom{K+j}{2j},\ 1 \le j \le K,$$
$$d_j := \sqrt{2h}(-1)^j(2h\kappa)^j \binom{K+j}{2j+1},\ 0 \le j \le K-1.$$

Then, we compute that for all $1 \le j \le K-1$, assuming $2h\kappa K^2 \le 1$,

$$0 \ge \frac{c_{j+1}}{c_j} = (-2h\kappa)\frac{(K+j)(K-j)}{(2j+2)(2j+1)} \ge -\frac{2h\kappa K^2}{12} \ge -0.1. \tag{10.50}$$

Similarly, for all $0 \le j \le K-2$,

$$0 \ge \frac{d_{j+1}}{d_j} = (-2h\kappa)\,\frac{(K+j+1)(K-j-1)}{(2j+3)(2j+2)} \ge -\frac{2h\kappa K^2}{6} \ge -0.2. \tag{10.51}$$

By repeating these calculations for α_1 and β_1, we see that all parameters are given by rapidly decaying geometric sequences, and thus the conclusion follows by examination from

$$\alpha_1 \in \left[0.8hK^2, hK^2\right],\ \alpha_{-1} \in \left[0.8h\kappa K^2, h\kappa K^2\right],$$
$$\beta_1 \in \left[0.8\sqrt{2h}K, \sqrt{2h}K\right],\ \beta_{-1} \in \left[0.8\sqrt{2h}K, \sqrt{2h}K\right].$$

□

We obtain the following corollary by combining Lemma 206, Corollary 53, and Proposition 49.

Corollary 54. *Let* $x \in \mathbb{R}^d$ *satisfy* $\|x_{-1}\|_2 \le \sqrt{\frac{2d}{3\kappa}}$ *and* $|x_1| \le 5\sqrt{\log d}$, *let* (x_K, v_K) *be the result of the* K*-step HMC Markov chain with step size* $\eta = \sqrt{2h}$ *with* $\eta^2 \le \frac{1}{\kappa K^2}$ *from* $x_0 = x$, *and let* $\mathbf{A}$ *be as in* (10.31). *Then with probability at least* $1 - d^{-5}$ *over the randomness of* $v_0 \sim \mathcal{N}(0, \mathbf{I})$, *we have*

$$\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) = -\Omega\left(h^2\kappa^2K^2d\right).$$

Proof. It suffices to use the bounds on $\beta_{-1} = \Theta(\sqrt{h}K)$ shown in the proof of Lemma 206 and the conclusions of Corollary 53 and Proposition 49. □

Removing the constant gap

We show how to improve the bound in Corollary 54 to only require $\eta^2 \le \frac{\pi^2}{\kappa K^2}$, which removes the constant gap between the requirement of Corollary 54 and the bound in Proposition 52. First, let $\mathbf{A}_c$ be the $D \times d$ diagonal matrix which induces the following hard quadratic function f_{hqc}:

$$f_{\mathrm{hqc}}(x) := \sum_{i \in [d]} f_i(x_i), \text{ where } f_i(c) = \begin{cases} \frac{1}{2}c^2 & i = 1 \\ \frac{\kappa}{2\pi^2}c^2 & 2 \le i \le d-1 \\ \frac{\kappa}{2}c^2 & i = d \end{cases}. \tag{10.52}$$

In other words, along the first $d-1$ coordinates, f_{hqc} is the same as a $d-1$-dimensional variant of f_{hq} with condition number $\frac{\kappa}{\pi^2}$. We define a coordinate partition of x and g into x_1, x_{-1d}, x_d, and g_1, g_{-1d}, g_d, and we define α_1, α_{-1d}, α_d, β_1, β_{-1d}, β_d in analogy with (10.32).

We first note that because of separability of f_{hqc}, and since the assumption of Corollary 54 holds on the first $d-1$ coordinates for $\eta^2 \le \frac{\pi^2}{\kappa K^2}$, we can immediately obtain a bound on the change in the Hamiltonian along these coordinates.

Corollary 55. *Let $x \in \mathbb{R}^d$ satisfy $\|x_{-1}\|_2 \le \sqrt{\frac{2\pi^2 d}{3\kappa}}$ and $|x_1| \le 5\sqrt{\log d}$, let (x_K, v_K) be the result of the K-step HMC Markov chain with step size $\eta = \sqrt{2h}$ where $\eta^2 \le \frac{\pi^2}{\kappa K^2}$ from $x_0 = x$, and let $\mathbf{A}_c$ be as in* (10.52)*. Then with probability at least $1 - 2d^{-5}$ over the randomness of $v_0 \sim \mathcal{N}(0, \mathbf{I})$, we have*

$$\mathcal{H}\left([x_0]_{[d-1]}, [v_0]_{[d-1]}\right) - \mathcal{H}\left([x_K]_{[d-1]}, [v_K]_{[d-1]}\right) = -\Omega\left(h^2\kappa^2 K^2 d\right).$$

We now move to bounding the contribution of the last coordinate.

Lemma 207. *Let (y, v_K) be the result of the K-step HMC Markov chain with step size $\eta = \sqrt{2h}$ where $\eta^2 \le \frac{\pi^2}{\kappa K^2}$, and write $y_d = (1 - \alpha_d)x_d + \beta_d g_d$, for*

$$\alpha_d = -\sum_{1 \le j \le K} (-1)^j (2h\kappa)^j \left(\frac{K}{K+j}\right)\binom{K+j}{2j}, \ \beta_d = \sqrt{2h}\sum_{0 \le j \le K-1} (-1)^j (2h\kappa)^j \binom{K+j}{2j+1}.$$

Then, we have $|\alpha_d| = O(h\kappa K^2)$, $|\beta_d| = O(\sqrt{h}K)$.

Proof. After the index j is a sufficiently large constant, the geometric argument sequence of Lemma 206 applies (since the denominators of the ratios (10.50) and (10.51) grow with the index j); before then, each coefficient is within a constant factor of the first in absolute value. Thus, the coefficients can be at most a constant factor larger than the first in absolute value. □

Lemma 208. *Let $|[x_0]_d| \le \frac{\log d}{\sqrt{\kappa}}$, $|[v_0]_d| \le \log d$, and let (x_K, v_K) be the result of the K-step HMC Markov chain with step size $\eta = \sqrt{2h}$ where $\eta^2 \le \frac{\pi^2}{\kappa K^2}$. Then with probability at least $1 - d^{-5}$ over*

the randomness of $v_0 \sim \mathcal{N}(0, \mathbf{I})$*, we have*

$$\mathcal{H}\left([x_0]_d, [v_0]_d\right) - \mathcal{H}\left([x_K]_d, [v_K]_d\right) = o\left(h^2\kappa^2 K^2 d\right).$$

Proof. We can assume $\operatorname{abs}[v_0]_d = \operatorname{abs} g_d \le \log d$, which passes the high probability bound. By Corollary 53 and Lemma 193, we wish to bound

$$\frac{h\kappa^2}{4}\left(\left(2\alpha_d - \alpha_d^2\right)x_d^2 - \beta_d^2 g_d^2 - 2(1-\alpha_d)\beta_d x_d g_d\right) = o\left(h^2\kappa^2 K^2 d\right).$$

Dropping all clearly negative terms, and since $|\alpha_d| = O(1)$ by Lemma 207, it is enough to show

$$\operatorname{abs} h\kappa^2\alpha_d x_d^2 = o\left(h^2\kappa^2 K^2 d\right),\ \operatorname{abs} h\kappa^2\beta_d x_d g_d = o\left(h^2\kappa^2 K^2 d\right).$$

The first bound is immediate from assumptions. The second follows from assumptions as well since $\sqrt{h\kappa K^2}$ is at most a constant, so $\operatorname{abs} h\kappa^2\beta_d x_d g_d = O(h^{1.5}\kappa^{1.5} K \log^2 d) = O(h^2\kappa^2 K^2 \log^2 d)$. □

By combining Lemma 208 and Corollary 55, we obtain the following strengthening of Corollary 54.

Corollary 56. *Let* $x \in \mathbb{R}^d$ *satisfy* $\|x_{-1d}\|_2 \le \sqrt{\frac{2d}{3\kappa}}$, $|x_1| \le 5\sqrt{\log d}$, *and* $|x_d| \le \frac{\log d}{\sqrt{\kappa}}$, *let* (x_K, v_K) *be the result of the* K*-step HMC Markov chain with step size* $\eta = \sqrt{2h}$ *with* $\eta^2 \le \frac{\pi^2}{\kappa K^2}$ *from* $x_0 = x$*, and let* $\mathbf{A}_c$ *be as in* (10.52)*. Then with probability at least* $1 - d^{-5}$ *over the randomness of* $v_0 \sim \mathcal{N}(0, \mathbf{I})$*, we have*

$$\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) = -\Omega\left(h^2\kappa^2 K^2 d\right).$$

Ruling out $\eta \ge 1$

Finally, we give a short argument ruling out the case $\eta \ge 1$ not covered by Proposition 52. In this section, let $\pi^* = \mathcal{N}(0, \kappa^{-1}\mathbf{I})$, with negative log-density $f(x) = \frac{\kappa}{2}\|x\|_2^2$. For $\eta \ge 1$ and $\kappa \ge 10$, (10.48) and straightforward lower bounds on Chebyshev polynomials outside the range $[-1, 1]$ demonstrate the proposal distribution is of the form (from starting point $x_0 \in \mathbb{R}^d$)

$$x_K \leftarrow \alpha x_0 + \beta v_0,\ v_0 \sim \mathcal{N}(0,1),\ |\alpha| \ge 10,\ |\beta| \ge 1. \tag{10.53}$$

Lemma 209. *Letting* (x_K, v_K) *be the result of* K*-step HMC from any* x_0*, and* $f(x) = \frac{\kappa}{2}\|x\|_2^2$*, for* $\eta \ge 1$*, with probability at least* $1 - d^{-5}$ *over the randomness of* $v_0 \sim \mathcal{N}(0, \mathbf{I})$*, we have*

$$\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) = -\Omega(d).$$

Proof. Following notation (10.53) and applying Corollary 53, it suffices to show

$$\|x_0\|_2^2 - \|\alpha x_0 + \beta v_0\|_2^2 = -\Omega(d).$$

Expanding, it suffices to upper bound

$$\left(1-\alpha^2\right)\|x_0\|_2^2 - 2\alpha\beta\left\langle x_0, v_0\right\rangle - \beta^2\|v_0\|_2^2.$$

With probability at least $1 - d^{-5}$, Fact 32 shows $\|v_0\|_2^2 \geq \frac{1}{2}d$ and $\langle x_0, v_0\rangle \geq -4\sqrt{\log d}\,\|x_0\|_2$. Hence,

$$\begin{aligned}\left(1-\alpha^2\right)\|x_0\|_2^2 - 2\alpha\beta\left\langle x_0, v_0\right\rangle - \beta^2\|v_0\|_2^2 &\leq -0.99\alpha^2\|x_0\|_2^2 + 8\alpha\beta\sqrt{\log d}\,\|x_0\|_2 - \frac{\beta^2}{2}d \\ &\leq 20\beta^2\log d - \frac{\beta^2}{2}d = -\Omega(d).\end{aligned}$$

Here, we used that $\alpha^2 \geq 100$ and took d larger than a sufficiently large constant. □

Proof of Theorem 74

A consequence of Corollary 54 is that if the step size $h = \omega(\frac{\sqrt{\log d}}{\kappa K\sqrt{d}})$, initializing the chain from any x_0 in the set Ω defined in (10.36) leads to a polynomially bad mixing time. We further relate the step size to the spectral gap of the HMC Markov chain in the following.

Lemma 210. *The spectral gap of the K-step HMC Markov chain for sampling from the density proportional to* $\exp(-f_{\mathrm{hq}})$*, where* f_{hq} *is defined in* (10.31)*, is* $O(hK^2 + h^2K^4)$*.*

Proof. We follow the proof of Lemma 192; again let $g(x) = x_1$, and π^* be the stationary distribution. For our function f, it is clear again that $\mathrm{Var}_{\pi^*}[g] = \Theta(1)$. Thus it suffices to upper bound $\mathcal{E}(g,g)$: letting $\mathcal{P}_x(y)$ be the proposal distribution of K-step HMC, and α_1, β_1 be as in (10.49),

$$\begin{aligned}\mathcal{E}(g,g) &\leq \frac{1}{2}\iint (x_1 - y_1)^2\mathcal{P}_x(y)d\pi^*(x)dy \\ &\leq \mathbb{E}_{x\sim\pi^*}\left[\alpha_1^2 x_1^2\right] + \mathbb{E}_{\xi\sim\mathcal{N}(0,1)}\left[\beta_1^2\xi^2\right] \\ &= \alpha_1^2 + \beta_1^2 = O\left(hK^2 + h^2K^4\right).\end{aligned}$$

□

Finally, by combining Lemma 210 and Corollary 56, we arrive at the main result of this section.

Theorem 74. *For every step size and count, there is a target Gaussian on $\mathbb{R}^d$ whose negative log-density always has Hessian eigenvalues in $[1,\kappa]$, such that the relaxation time of HMC is $\Omega(\frac{\kappa\sqrt{d}}{K\sqrt{\log d}})$.*

Proof. For $1 \geq \eta^2 \geq \frac{\pi^2}{\kappa K^2}$ it suffices to apply Proposition 52. For $\eta^2 \geq 1$, we apply Lemma 209. Otherwise, in the relevant range of $h = 2\eta^2$, the dominant term in Lemma 210 is $O(hK^2)$. Applying

Corollary 56 with the hard quadratic function f_{hqc}, the remainder of the proof follows analogously to that of Theorem 71. □

We remark that as in Theorem 71, it is straightforward to see that the measure of the bad region $\|x_{-1d}\|_2 \le \sqrt{\frac{2d}{3\kappa}}$, $|x_1| \le 5\sqrt{\log d}$, and $|x_d| \le \frac{\log d}{\sqrt{\kappa}}$ used in the proof is at least $\exp(-d)$.

10.11 Conclusion

In this work, we presented relaxation time lower bounds for the MALA and HMC Markov chains at every step size and scale, as well as a mixing time bound for MALA from an exponentially warm start. We highlight in this section a number of unexplored directions left open by our work, beyond direct strengthenings of our results, which we find interesting and defer to a future exploration.

Variable or random step sizes. Our lower bounds were for MALA and HMC Markov chains with a *fixed step size*. For variable step sizes which take e.g. values in a bounded multiplicative range, we believe our arguments can be modified to give relaxation time lower bounds for the resulting Markov chains. However, the arguments of Section 10.10 (our HMC lower bound) are particularly brittle to large multiplicative ranges of candidate step sizes, because they rely on the locations of Chebyshev polynomial zeroes, which only occur in a bounded range. From an algorithm design perspective, this suggests that adaptively or randomly choosing step size ranges may be effective in improving the performance of HMC. Such a result would also give theoretical justification to the No-U-Turn sampler of [273], a common HMC alternative in practice. We state as an explicit open problem: can one obtain improved upper bounds, such as a $\sqrt{\kappa}$ dependence or a dimension-independent rate, for example by using variations of these strategies (variable step sizes)?

Necessity of κ lower bound. All of our witness sets throughout the paper are $\exp(-d)$ sized. It was observed in [210] that it is possible to construct a starting distribution with warmness arbitrarily close to $\sqrt{\kappa}^d$; the marginal restriction of our witness set falls under this warmness bound for all $\kappa \ge e^2 \approx 8$. However, recently [346] proposed a *proximal point reduction* approach to sampling, which (for mixing bounds scaling at least linearly in κ) shows that it suffices to sample a small number of regularized distributions, whose condition numbers are arbitrarily close to 1.

By adjusting constants, we can modify the proof of the Gaussian lower bounds (Theorems 71 and 74) to have witness sets with measure c^d for a constant c arbitrarily close to 1 (the bottleneck being Lemma 195). However, our witness set for the family of hard distributions in Section 10.8 encounters a natural barrier at measure 2^d, since the set is sign-restricted by the cosine function (and hence can only contain roughly every other period). This bottleneck is encountered in the proof of Lemma 200. We find it interesting to see if a stronger construction rules out the existing warm starts for all $\kappa \ge 1$, or if an upper bound can take advantage of the reduction of [346] to obtain improved dependences on dimension assuming $\kappa \approx 1$.

Appendix A

Deferred proofs from Chapter 2

A.1 Convex analysis

We use the standard notion of a convex conjugate of a convex function, also known as the Fenchel dual, which we will refer to as a "dual function" throughout.

Definition 47 (Fenchel dual)**.** *For convex function f, its* Fenchel dual *is defined by*

$$f^*(y) := \min_x \{\langle y, x\rangle - f(x)\}$$

We state several key facts about dual functions that we will frequently make use of. These are well-known and we defer proofs to standard texts, e.g. [458].

Lemma 211 (Convex conjugate, Fenchel duality)**.** *For any convex f, its Fenchel dual f^*, and x, y in their respective domains, $f(x) + f^*(y) \geq \langle y, x\rangle$.*

Lemma 212 (Conjugate of a conjugate, maximizing argument)**.** *For convex f, it holds that $f^{**}(x) = f(x)$, and if $x = \operatorname{argmax}_{x'}\{\langle y, x\rangle - f(x')\}$, then $x \in \partial f^*(y)$ where ∂ is the subgradient operator. Consequently, if f and f^* are both differentiable, ∇f and ∇f^* are inverse functions.*

Lemma 213 (Divergence of f^*)**.** *The Bregman divergences of dual functions are related by*

$$V^{f^*}_{\nabla f(x)}(\nabla f(x')) = V^f_{x'}(x).$$

We prove two statements regarding strong convexity of the dual of a smooth function. The first has been previously been observed by e.g. [303], but we include a proof for completeness as a precursor to Lemma 215, a coordinate smoothness generalization.

Lemma 214 (Strong convexity of the dual)**.** *Suppose f is convex and L-smooth with respect to $\|\cdot\|$. Then, f^* is $\frac{1}{L}$-strongly convex with respect to $\|\cdot\|_*$.*

Proof. This is equivalent to showing that, for two dual points ξ_1, ξ_2, we have

$$f^*(\xi_2) - f^*(\xi_1) - \langle \nabla f^*(\xi_1), \xi_1 - \xi_2 \rangle \geq \frac{1}{2L} \|\xi_1 - \xi_2\|_*^2 .$$

Writing $y = \nabla f^*(\xi_1), x = \nabla f^*(\xi_2)$, we also have $\xi_1 = \nabla f(y), \xi_2 = \nabla f(x)$ by the earlier analysis of maximizing arguments. Furthermore, by Lemma 213 relating the Bregman divergences of conjugate functions, it suffices to show that

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle \geq \frac{1}{2L} \|\nabla f(y) - \nabla f(x)\|_*^2 .$$

Let

$$z = \operatorname{argmin}_w \langle \nabla f(y) - \nabla f(x), w - y \rangle + \frac{L}{2} \|w - y\|^2 ,$$

where we note that by the equality case of Cauchy-Schwarz, we have

$$\langle \nabla f(y) - \nabla f(x), z - y \rangle + \frac{L}{2} \|z - y\|^2 = -\frac{1}{2L} \|\nabla f(y) - \nabla f(x)\|_*^2 .$$

Then, the proof follows directly by using convexity and smoothness. Indeed,

$$\begin{aligned}
0 &\leq f(z) - f(x) - \langle \nabla f(x), z - x \rangle \\
&\leq f(y) + \langle \nabla f(y), z - y \rangle + \frac{L}{2} \|z - y\|^2 - f(x) - \langle \nabla f(x), z - x \rangle \\
&= f(y) - f(x) - \langle \nabla f(x), y - x \rangle + \langle \nabla f(y) - \nabla f(x), z - y \rangle + \frac{L}{2} \|z - y\|^2 \\
&= f(y) - f(x) - \langle \nabla f(x), y - x \rangle - \frac{1}{2L} \|\nabla f(y) - \nabla f(x)\|_*^2 ,
\end{aligned}$$

where in the last line, we used the definition of z. This proves the desired claim. □

The following Lemma 215 can be thought of as a limit of Lemma 214 in diagonal quadratic norms $\|\cdot\|$ where all but one coordinate tends to ∞ (and in the dual, only one coordinate is nonzero), which captures coordinate smoothness. We provide a more direct proof for completeness.

Lemma 215 (Coordinate strong convexity of the dual)**.** *For a convex function f which is L_i-smooth in the i^{th} coordinate, we have*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L_i} |\nabla_i f(y) - \nabla_i f(x)|^2 .$$

Proof. The proof is essentially the same as that of Lemma 214, with a tighter guarantee given by

coordinate smoothness. Let $\|\cdot\|$ be the ℓ_2 norm, and

$$z = \text{argmin}_{w:w=y+te_i} \langle \nabla f(y) - \nabla f(x), w - y\rangle + \frac{L_i}{2}\|w-y\|^2$$

where we note that by the equality case of Cauchy-Schwarz, and as y and z only differ in the i^{th} coordinate, we have

$$\langle \nabla f(y) - \nabla f(x), z - y\rangle + \frac{L_i}{2}\|z-y\|^2 = -\frac{1}{2L_i}|\nabla_i f(y) - \nabla_i f(x)|^2.$$

Then, the proof follows:

$$\begin{aligned}
0 &\le f(z) - f(x) - \langle \nabla f(x), z - x\rangle \\
&\le f(y) + \langle \nabla f(y), z - y\rangle + \frac{L_i}{2}\|z-y\|^2 - f(x) - \langle \nabla f(x), z - x\rangle \\
&= f(y) - f(x) - \langle \nabla f(x), y - x\rangle + \langle \nabla f(y) - \nabla f(x), z - y\rangle + \frac{L_i}{2}\|z-y\|^2 \\
&= f(y) - f(x) - \langle \nabla f(x), y - x\rangle - \frac{1}{2L_i}|\nabla_i f(y) - \nabla_i f(x)|^2
\end{aligned}$$

where in the last line, we used the definition of z. This proves the desired claim. □

As a corollary of (2.10), we obtain the following.

Lemma 216. *For $w = \text{Prox}_z^r(g)$, $\forall u$, $\langle g, w - u\rangle \le V_z^r(u) - V_w^r(u) - V_z^r(w)$.*

Proof. By first order optimality of the function $\langle g, w\rangle + V_z^r(w)$ as a function of w, we have

$$\langle g + \nabla V_z^r(w), u - w\rangle \ge 0 \implies \langle g, w - u\rangle \le \langle -\nabla V_z^r(w), w - u\rangle = V_z^r(u) - V_w^r(u) - V_z^r(w),$$

where we used (2.10). □

Lemma 2. *If g is L-Lipschitz and r is μ-strongly convex in $\|\cdot\|$, g is L/μ-relatively Lipschitz with respect to r.*

Proof. By Cauchy-Schwarz, Lipschitzness of g, and strong convexity of r,

$$\begin{aligned}
\langle g(w) - g(z), w - u\rangle &\le \|g(w) - g(z)\|_* \|w - u\| \le L\|w - z\|\|w - u\| \\
&\le \frac{L}{2}\left(\|w-z\|^2 + \|w-u\|^2\right) \le \frac{L}{\mu}\left(V_z^r(w) + V_w^r(u)\right).
\end{aligned}$$

□

Lemma 3. *If f is L-relatively smooth with respect to r, i.e. $V_x^f(y) \le LV_x^r(y)$ for all x and y, then g, defined by $g(x) := \nabla f(x)$ for all x, is L-relatively Lipschitz with respect to r.*

Proof. By assumption of relative smoothness of f and the definition of divergence,

$$\begin{aligned} L\left(V_z^r(w) + V_w^r(u)\right) &\geq V_z^f(w) + V_w^f(u) \\ &= f(w) - \left[f(z) + \nabla f(z)^\top (w - z)\right] + f(u) - \left[f(w) + \nabla f(w)^\top (u - w)\right] \\ &= V_z^f(u) - \nabla f(z)^\top (z - u) - \nabla f(z)^\top (w - z) + \nabla f(w)^\top (w - u) \\ &= V_z^f(u) + \langle g(w) - g(z), u - z \rangle \ . \end{aligned}$$

The result follows from the fact that $V_z^f(u) \geq 0$ by convexity of f. □

A.2 Unaccelerated smooth convex optimization via mirror prox

For completeness, we provide here the proof that the $1/T$ rate of extragradient methods for solving Lipschitz monotone VIs also yields a $1/T$ rate for smooth function minimization. Given a smooth, convex f this rate is achieved by applying these methods to solve the VI induced by $g(x) := \nabla f(x)$. Consequently, this section highlights that the accelerated rates we achieve for minimizing f are via working in the expanded primal-dual space of the VI induced by the Fenchel game, as well as our fine-grained notion of relative Lipschitzness, as considered in Section 2.4.

Lemma 217. *For L-smooth, convex $f : \mathbb{R}^d \to \mathbb{R}$ define $g : \mathbb{R}^d \to \mathbb{R}$ by $g(x) := \nabla f(x)$, and for arbitrary x_0 let $r(x) := \frac{1}{2}\|x_0 - x\|_2^2$. For all $T \geq 0$, both* Mirror-Prox(x_0, T) *(Algorithm 1) and* Dual-Ex(x_0, T) *(Algorithm 65) produce $\{w_t\}_{0 \leq t < T}$ such that for any x^* minimizing f,*

$$f\left(\frac{1}{T}\sum_{0 \leq t < T} w_t\right) - f(x^*) \leq \frac{L\|x_0 - x^*\|_2^2}{2T}.$$

Proof. Since $x_0 = \operatorname{argmin}_{x \in \mathbb{R}^d} r(x)$, and since (by Lemma 2 and smoothness) g is L-relatively Lipschitz with respect to r, Proposition 1 and Proposition 53 imply

$$\sum_{0 \leq t < T} \langle g(w_t), w_t - u \rangle \leq L V_{x_0}^r(u).$$

Consequently, applying convexity and letting $u = x^*$ above yields

$$\begin{aligned} f\left(\frac{1}{T}\sum_{0 \leq t < T} w_t\right) - f(x_*) \leq \frac{1}{T}\sum_{0 \leq t < T}\left(f(w_t) - f(x^*)\right) \leq \frac{1}{T}\sum_{0 \leq t < T}\langle g(w_t), w_t - x^* \rangle \leq \frac{L V_{x_0}^r(x^*)}{T} \\ = \frac{L\|x_0 - x^*\|_2^2}{2T}. \end{aligned}$$

□

We remark that for μ-strongly convex functions, it is well-known that this also implies a $O(\frac{L}{\mu})$ rate of linear convergence to an approximate minimizer by converting function error back to a bound on the squared distance to x^* via applying strong convexity.

A.3 Minimax optimization

In this section, we give a new fine-grained complexity bound of minimax optimization under block-wise strong convexity and smoothnesses of the problem. Specifically, consider the problem

$$\min_{x\in\mathbb{R}^d}\max_{y\in\mathbb{R}^d} f(x,y), \tag{A.1}$$

and assume that the objective satisfies the following bounds[1]:

1. $\left\|\nabla^2_{xx} f(x,y)\right\|_{\text{op}} \le L_{xx}$, $\left\|\nabla^2_{xy} f(x,y)\right\|_{\text{op}} \le L_{xy}$, $\left\|\nabla^2_{yy} f(x,y)\right\|_{\text{op}} \le L_{yy}$ everywhere
2. $f(x,\cdot)$ is μ_y-strongly concave for each x, and $f(\cdot,y)$ is μ_x-strongly convex for each y

We show an upper bound of

$$O\left(\left(\frac{L_{xx}}{\mu_x} + \sqrt{\frac{L_{xy}^2}{\mu_x\mu_y}} + \frac{L_{yy}}{\mu_y}\right)\cdot \log\frac{1}{\epsilon}\right)$$

queries via a simple application of our relative Lipschitzness framework, and a known strongly monotone variant of Algorithm 1, for obtaining ϵ Euclidean distance to the saddle point of (A.1), improving upon the bound of [366] in some cases. This can be converted into a duality gap bound while only increasing the logarithmic term by problem parameters; see the end of this section for a discussion. We will use Algorithm 3 and a tightening of its guarantees following Definition 1, in Proposition 3, a strengthening of a known derivation (see e.g. Proposition 5 of [111]) under relative Lipschitzness. We defer its proof to Appendix A.4. Our claimed upper bound follows by combining Proposition 3 with the following relative Lipschitzness bound. We note that the same rate can be obtained by directly combining Proposition 3 with a rescaling of the space to make the objective 1-strongly convex in each variable, but we include a proof via relative Lipschitzness because it makes the calculation more mechanical (and also originally motivated this observation).

Lemma 218. *Let $r(x,y) = \frac{\mu_x}{2}\|x\|_2^2 + \frac{\mu_y}{2}\|y\|_2^2$ and $g(x,y) = (\nabla_x f(x,y), -\nabla_y f(x,y))$. Then with respect to r, g is 1-strongly monotone and*

$$\frac{L_{xx}}{\mu_x} + \sqrt{\frac{L_{xy}^2}{\mu_x\mu_y}} + \frac{L_{yy}}{\mu_y}\text{-relatively Lipschitz.}$$

[1]Here, $\|\cdot\|_{\text{op}}$ is the ℓ_2 operator norm.

Proof. First, we prove strong monotonicity: let $z = (z^{\mathsf{x}}, z^{\mathsf{y}})$ and $w = (w^{\mathsf{x}}, w^{\mathsf{y}})$. Moreover, let $\mathbf{J}$ be the Jacobian of the operator g, and note that

$$g(w) - g(z) = \int_0^1 \mathbf{J}(z_t)(w - z)dt, \text{ where } z_t := (1-t)z + tw.$$

Thus, integrating and using antisymmetry of the off-diagonal blocks of $\mathbf{J}$,

$$\begin{aligned}
\langle g(w) - g(z), w - z\rangle &= \int_0^1 (w-z)^\top \mathbf{J}(z_t)(w-z)dt \\
&= \int_0^1 \left((w^{\mathsf{x}} - z^{\mathsf{x}})^\top \nabla^2 f_{xx}(z_t)(w^{\mathsf{x}} - z^{\mathsf{x}}) - (w^{\mathsf{y}} - z^{\mathsf{y}})^\top \nabla^2 f_{yy}(z_t)(w^{\mathsf{y}} - z^{\mathsf{y}})\right) dt \\
&\geq \int_0^1 \left(\mu_x \left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2^2 + \mu_y \left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2^2\right) dt = V_z^r(w) + V_w^r(z).
\end{aligned}$$

In the only inequality, we used our strong convexity and strong concavity assumptions. Next, we prove the relative Lipschitz bound: consider three points z, w, and $u = (u^{\mathsf{x}}, u^{\mathsf{y}})$. We have by triangle inequality and the assumed operator norm bounds

$$\begin{aligned}
g^{\mathsf{x}}(w) - g^{\mathsf{x}}(z) &= \int_0^1 \left(\nabla^2_{xx} f(z_t)(w^{\mathsf{x}} - z^{\mathsf{x}}) + \nabla^2_{xy} f(z_t)(w^{\mathsf{y}} - z^{\mathsf{y}})\right) dt \\
&\implies \left\|g^{\mathsf{x}}(w) - g^{\mathsf{x}}(z)\right\|_2 \leq L_{xx}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 + L_{xy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2, \\
g^{\mathsf{y}}(w) - g^{\mathsf{y}}(z) &= -\int_0^1 \left(\nabla^2_{yx} f(z_t)(w^{\mathsf{x}} - z^{\mathsf{x}}) + \nabla^2_{yy} f(z_t)(w^{\mathsf{y}} - z^{\mathsf{y}})\right) dt \\
&\implies \left\|g^{\mathsf{y}}(w) - g^{\mathsf{y}}(z)\right\|_2 \leq L_{xy}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 + L_{yy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2.
\end{aligned}$$

Therefore, it follows from Cauchy-Schwarz that

$$\begin{aligned}
\langle g(w) - g(z), w - u\rangle \leq\ & L_{xx}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 \left\|w^{\mathsf{x}} - u^{\mathsf{x}}\right\|_2 + L_{xy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2 \left\|w^{\mathsf{x}} - u^{\mathsf{x}}\right\|_2 \\
& + L_{xy}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 \left\|w^{\mathsf{y}} - u^{\mathsf{y}}\right\|_2 + L_{yy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2 \left\|w^{\mathsf{y}} - u^{\mathsf{y}}\right\|_2.
\end{aligned}$$

Finally, denoting $r^{\mathsf{x}}(x) := \frac{\mu_x}{2}\|x\|_2^2$, $r^{\mathsf{y}}(y) := \frac{\mu_y}{2}\|y\|_2^2$, the conclusion of relative Lipschitzness follows from nonnegativity of divergences and combining the three bounds

$$\begin{aligned}
L_{xx}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 \left\|w^{\mathsf{x}} - u^{\mathsf{x}}\right\|_2 &\leq \frac{L_{xx}}{\mu_x}\left(V_{z^{\mathsf{x}}}^{r^{\mathsf{x}}}(w^{\mathsf{x}}) + V_{w^{\mathsf{x}}}^{r^{\mathsf{x}}}(u^{\mathsf{x}})\right), \\
L_{yy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2 \left\|w^{\mathsf{y}} - u^{\mathsf{y}}\right\|_2 &\leq \frac{L_{yy}}{\mu_y}\left(V_{z^{\mathsf{y}}}^{r^{\mathsf{y}}}(w^{\mathsf{y}}) + V_{w^{\mathsf{y}}}^{r^{\mathsf{y}}}(u^{\mathsf{y}})\right), \\
L_{xy}\left\|w^{\mathsf{y}} - z^{\mathsf{y}}\right\|_2 \left\|w^{\mathsf{x}} - u^{\mathsf{x}}\right\|_2 + L_{xy}\left\|w^{\mathsf{x}} - z^{\mathsf{x}}\right\|_2 \left\|w^{\mathsf{y}} - u^{\mathsf{y}}\right\|_2 &\leq \frac{L_{xy}}{\sqrt{\mu_x\mu_y}}\left(V_z^r(w) + V_w^r(u)\right).
\end{aligned}$$

We prove the last bound here; the other two follow similarly. By expanding the definition of r,

$$\begin{aligned}\frac{L_{xy}}{\sqrt{\mu_x\mu_y}}\left(V_z^r(w)+V_w^r(u)\right) &= \frac{L_{xy}}{\sqrt{\mu_x\mu_y}}\Big(\frac{\mu_x}{2}\left\|w^{\mathsf{x}}-z^{\mathsf{x}}\right\|_2^2+\frac{\mu_y}{2}\left\|w^{\mathsf{y}}-z^{\mathsf{y}}\right\|_2^2 \\ &\quad +\frac{\mu_x}{2}\left\|z_+^{\mathsf{x}}-w^{\mathsf{x}}\right\|_2^2+\frac{\mu_y}{2}\left\|z_+^{\mathsf{y}}-w^{\mathsf{y}}\right\|_2^2\Big) \\ &\geq L_{xy}\left(\left\|w^{\mathsf{x}}-z^{\mathsf{x}}\right\|_2\left\|w^{\mathsf{y}}-u^{\mathsf{y}}\right\|_2+\left\|w^{\mathsf{x}}-u^{\mathsf{x}}\right\|_2\left\|w^{\mathsf{y}}-z^{\mathsf{y}}\right\|_2\right).\end{aligned}$$

□

We give a brief discussion regarding converting a distance bound from a pair (x, y) to the saddle point (x^*, y^*) of (A.1) into a duality gap bound. Specifically, let y' be the best response to x, and x' be the best response to y. Further, denote

$$h_1(x) := \max_{y\in\mathbb{R}^d} f(x,y),\ h_2(y) := \min_{x\in\mathbb{R}^d} f(x,y).$$

Note the duality gap of the pair (x, y) is precisely

$$\begin{aligned}f(x,y')-f(x',y) &= \left(f(x,y')-f(x^*,y^*)\right)+\left(f(x^*,y^*)-f(x',y)\right) \\ &= \left(h_1(x)-h_1(x^*)\right)+\left(h_2(y^*)-h_2(y)\right).\end{aligned}$$

Denote $L = \max(L_{xx}, L_{xy}, L_{yy})$. Under the setting of this section, it was shown in Lemma B.2 of [366] that h_1 is $\frac{2L^2}{\mu_x}$ smooth, which implies the bound (since x^* minimizes h_1 by definition)

$$h_1(x)-h_1(x^*) \leq \frac{L^2}{\mu_x}\left\|x-x^*\right\|_2^2.$$

Consequently, we can convert a distance bound into a duality gap bound, while only affecting the logarithmic runtime term. A similar argument holds for terms corresponding to h_2.

A.4 Additional extragradient methods

A.4.1 Strongly monotone mirror prox

We give a proof of Proposition 3, restated here for convenience.

Proposition 3. *If Φ is λ-relatively Lipschitz with respect to r over $\mathcal{Z}_{\text{alg}}$ containing all iterates of Algorithm 3, and its VI is solved by $z_\star$, the iterates of Algorithm 3 satisfy*

$$V_{z_t}^r(z_\star) \leq \left(1+\frac{m}{\lambda}\right)^{-t} V_{z_0}^r(z_\star),\ \textit{for all } t\in[T].$$

Proof. As in (2.11), first-order optimality with respect to z^* and nonnegativity of $V_{w_t}(z_{t+1})$ implies

$$\frac{1}{\lambda}\langle g(z_t), w_t - z_{t+1}\rangle \le V_{z_t}^r(z_{t+1}) - V_{w_t}^r(z_{t+1}) - V_{z_t}^r(w_t),$$
$$\frac{1}{\lambda}\langle g(w_t), z_{t+1} - z^*\rangle \le V_{z_t}^r(z^*) - V_{z_{t+1}}^r(z^*) - V_{z_t}^r(z_{t+1}) + \frac{m}{\lambda}\left(V_{w_t}^r(z^*) - V_{z_{t+1}}^r(z^*)\right).$$

Rearranging and applying relative Lipschitzness (Definition 1) as in (2.12) yields

$$\left(1+\frac{m}{\lambda}\right)V_{z_{t+1}}(z^*) \le \frac{1}{\lambda}\left(\langle g(w_t), w_t - z^*\rangle - mV_{w_t}(z^*)\right) + \left(1+\frac{m}{\lambda}\right)V_{z_{t+1}}(z^*) \le V_{z_t}(z^*). \tag{A.2}$$

Here, we used the fact that by the definition of z^* and strong monotonicity,

$$\langle g(w_t), w_t - z^*\rangle - mV_{w_t}(z^*) \ge \langle g(w_t) - g(z^*), w_t - z^*\rangle - mV_{w_t}(z^*) \ge 0.$$

Thus, iterating the inequality (A.2) yields the conclusion. □

A.4.2 Dual extrapolation

In this section, we give a simplified presentation of the dual extrapolation algorithm of [420] for approximately solving a variational inequality in a monotone operator g, using a regularizer r. It obtains the same rate of convergence as the mirror prox algorithm, under relative Lipschitzness (Definition 1). Specifically, it is the following algorithm which updates a dual variable s_t iteratively.

Algorithm 65: DUAL-EX$(\bar{z}, T)$: Dual extrapolation [420]

Input: Distance generating r, λ-relatively Lipschitz monotone $g : \mathcal{Z} \to \mathcal{Z}^*$, initial point $\bar{z} \in \mathcal{Z}$;
$s_0 \leftarrow 0$;
for $0 \le t < T$ **do**
  $z_t \leftarrow \mathrm{Prox}_{\bar{z}}^r(s_t)$;
  $w_t \leftarrow \mathrm{Prox}_{z_t}^r(\frac{1}{\lambda}g(z_t))$;
  $s_{t+1} \leftarrow s_t + \frac{1}{\lambda}g(w_t)$;

Proposition 53. *The iterates $\{w_t\}$ of Algorithm 65 satisfy for all $u \in \mathcal{Z}$,*

$$\sum_{0\le t<T}\langle g(w_t), w_t - u\rangle \le \lambda V_{\bar{z}}^r(u).$$

Proposition 53 requires the following helper lemma, the main tool in its proof.

Lemma 219. *For every iteration t,*

$$\frac{1}{\lambda}\langle g(w_t), w_t - \bar{z}\rangle \le \langle s_{t+1}, z_{t+1} - \bar{z}\rangle + V_{\bar{z}}^r(z_{t+1}) - \langle s_t, z_t - \bar{z}\rangle - V_{\bar{z}}^r(z_t).$$

Proof. We again apply Lemma 216 on the two steps with respect to z_{t+1}, yielding

$$\langle s_t, z_t - z_{t+1}\rangle \le V_{\bar{z}}^r(z_{t+1}) - V_{z_t}^r(z_{t+1}) - V_{\bar{z}}^r(z_t),$$
$$\frac{1}{\lambda}\langle g(z_t), w_t - z_{t+1}\rangle \le V_{z_t}^r(z_{t+1}) - V_{w_t}^r(z_{t+1}) - V_{z_t}^r(w_t).$$

Furthermore, note that by relative Lipschitzness, we have

$$\frac{1}{\lambda}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle \le V_{w_t}^r(z_{t+1}) + V_{z_t}^r(w_t).$$

Combining these three inequalities and rearranging terms appropriately yields the conclusion:

$$\langle s_t, z_t - z_{t+1}\rangle + \frac{1}{\lambda}\langle g(w_t), w_t - z_{t+1}\rangle \le V_{\bar{z}}^r(z_{t+1}) - V_{\bar{z}}^r(z_t)$$
$$\implies \frac{1}{\lambda}\langle g(w_t), w_t - \bar{z}\rangle \le \langle s_{t+1}, z_{t+1} - \bar{z}\rangle + V_{\bar{z}}^r(z_{t+1}) - \langle s_t, z_t - \bar{z}\rangle - V_{\bar{z}}^r(z_t).$$

□

This immediately yields the following:

Corollary 57. $\Phi_t = \frac{1}{\lambda}\sum_{k=0}^{t-1}\langle g(w_k), w_k - \bar{z}\rangle - \langle s_t, z_t - \bar{z}\rangle - V_{\bar{z}}(z_t)$ *is nonincreasing in* t.

Proof of Proposition 53. Note that for any u,

$$\begin{aligned}\sum_{t=0}^{T-1}\langle g(w_t), w_t - u\rangle &= \sum_{t=0}^{T-1}\langle g(w_t), w_t - \bar{z}\rangle + \sum_{t=0}^{T-1}\langle g(w_t), \bar{z} - u\rangle + (\lambda V_{\bar{z}}(u) - \lambda V_{\bar{z}}(u))\\ &\le \sum_{t=0}^{T-1}\langle g(w_t), w_t - \bar{z}\rangle + \sum_{t=0}^{T-1}\langle g(w_t), \bar{z} - z_T\rangle + (\lambda V_{\bar{z}}(u) - \lambda V_{\bar{z}}(z_T))\\ &= \lambda\Phi_T + \lambda V_{\bar{z}}(u) \le \lambda\Phi_0 + \lambda V_{\bar{z}}(u) = \lambda V_{\bar{z}}(u).\end{aligned}$$

The first inequality used the definition of z_T, and the second inequality used Corollary 57. □

We note that all our acceleration results are also implementable with dual extrapolation as the base method, by analogous arguments as in Sections 2.4 and 2.6.

A.5 Extragradient acceleration in non-Euclidean norms

In this section, we generalize the developments of Section 2.4 to general norms, where f is L-smooth in some $\|\cdot\|$. The notion of strong convexity in general norms is slightly different; for acceleration to be achievable, f must be μ-strongly convex with respect to a regularizer ω, where ω is 1-strongly convex in $\|\cdot\|$; see e.g. [18] for a discussion.

We first state our general strategy. Let $h(x) := f(x) - \mu\omega(x)$, for all $x \in \mathbb{R}^d$; it is immediate that h is convex and L-smooth in $\|\cdot\|$. To solve the problem $\min_{x\in\mathbb{R}^d} f(x) = \min_{x\in\mathbb{R}^d} h(x) + \mu\omega(x)$, it instead suffices to solve the equivalent saddle point problem

$$\min_{x\in\mathbb{R}^d}\max_{y\in\mathbb{R}^d} \mu\omega(x) + \langle y, x\rangle - h^*(y). \tag{A.3}$$

It is immediate that the saddle point of (A.3) is $z^* := (x^*, \nabla h(x^*))$. The key observation is that (A.3) is strongly monotone and relatively Lipschitz (with an accelerated parameter) with respect to the natural choice of regularizer,

$$r(x, y) := \mu\omega(x) + h^*(y). \tag{A.4}$$

From this point, it suffices to apply the strongly monotone extragradient framework of Proposition 3. We now make this formal by giving the following two helper lemmata.

Lemma 220. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth in norm $\|\cdot\|$, and μ-strongly convex with respect to ω, a 1-strongly convex function in $\|\cdot\|$. Let $h(x) := f(x) - \mu\omega(x)$ $\forall x \in \mathbb{R}^d$, and let $g(x, y) = (y + \mu\nabla\omega(x), \nabla h^*(y) - x)$ be the gradient operator of the objective* (A.3)*. Then, g is 1-strongly monotone with respect to the distance generating function r defined in* (A.4)*.*

Proof. By definition, it suffices to show that for all w, $z \in \mathbb{R}^d \times \mathbb{R}^d$,

$$\langle g(w) - g(z), w - z\rangle \geq V_w^r(z) + V_z^r(w) = \langle \nabla r(w) - \nabla r(z), w - z\rangle .$$

By direct expansion, this is an equality. In particular, for $w = (w^{\mathsf{x}}, w^{\mathsf{y}})$ and $z = (z^{\mathsf{x}}, z^{\mathsf{y}})$,

$$\begin{aligned}
\langle g(w) - g(z), w - z\rangle &= \langle (w^{\mathsf{y}} - z^{\mathsf{y}}) + \mu\left(\nabla\omega(w^{\mathsf{x}}) - \nabla\omega(z^{\mathsf{x}})\right), w^{\mathsf{x}} - z^{\mathsf{x}}\rangle \\
&\quad + \langle (\nabla h^*(w^{\mathsf{y}}) - \nabla h^*(z^{\mathsf{y}})) - (w^{\mathsf{x}} - z^{\mathsf{x}}), w^{\mathsf{y}} - z^{\mathsf{y}}\rangle \\
&= \mu\langle \nabla\omega(w^{\mathsf{x}}) - \nabla\omega(z^{\mathsf{x}}), w^{\mathsf{x}} - z^{\mathsf{x}}\rangle + \langle \nabla h^*(w^{\mathsf{y}}) - \nabla h^*(z^{\mathsf{y}}), w^{\mathsf{y}} - z^{\mathsf{y}}\rangle \\
&= \langle \nabla r(w) - \nabla r(z), w - z\rangle .
\end{aligned}$$

□

Lemma 221. *In the setting of Lemma 220, g is $1 + \sqrt{\frac{L}{\mu}}$-relatively Lipschitz with respect to r.*

Proof. The proof is patterned off of Lemma 4. Consider three points $z = (z^{\mathsf{x}}, z^{\mathsf{y}})$, $w = (w^{\mathsf{x}}, w^{\mathsf{y}})$, $u = (u^{\mathsf{x}}, u^{\mathsf{y}})$. By direct calculation,

$$\begin{aligned}
\langle g(w) - g(z), w - u\rangle &= \langle w^{\mathsf{y}} - z^{\mathsf{y}}, w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle -w^{\mathsf{x}} + z^{\mathsf{x}}, w^{\mathsf{y}} - u^{\mathsf{y}}\rangle \\
&\quad + \mu\langle \nabla\omega(w^{\mathsf{x}}) - \nabla\omega(z^{\mathsf{x}}), w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle \nabla h^*(w^{\mathsf{y}}) - \nabla h^*(z^{\mathsf{y}}), w^{\mathsf{y}} - u^{\mathsf{y}}\rangle .
\end{aligned} \tag{A.5}$$

To bound the first line of (A.5), we have the following analog to (2.14), where we use that h^* is

$\frac{1}{L}$-strongly convex in $\|\cdot\|_*$ by Lemma 214:

$$\begin{aligned}\langle w^{\mathsf{y}} - z^{\mathsf{y}}, w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle z^{\mathsf{x}} - w^{\mathsf{x}}, w^{\mathsf{y}} - u^{\mathsf{y}}\rangle &\le \|w^{\mathsf{y}} - z^{\mathsf{y}}\|_* \|w^{\mathsf{x}} - u^{\mathsf{x}}\| + \|z^{\mathsf{x}} - w^{\mathsf{x}}\| \|w^{\mathsf{y}} - u^{\mathsf{y}}\|_* \\ &\le \sqrt{\frac{L}{\mu}} \left(\frac{\mu}{2} \|w^{\mathsf{x}} - z^{\mathsf{x}}\|^2 + \frac{\mu}{2} \|w^{\mathsf{x}} - u^{\mathsf{x}}\|^2 + \frac{1}{2L} \|w^{\mathsf{y}} - z^{\mathsf{y}}\|_*^2 + \frac{1}{2L} \|w^{\mathsf{y}} - u^{\mathsf{y}}\|_*^2 \right) \\ &\le \sqrt{\frac{L}{\mu}} \left(V_z^r(w) + V_w^r(u) \right).\end{aligned}$$

To bound the second line of (A.5), we use that the calculation (2.15) implies that

$$\begin{aligned}\mu \langle \nabla\omega(w^{\mathsf{x}}) - \nabla\omega(z^{\mathsf{x}}), w^{\mathsf{x}} - u^{\mathsf{x}}\rangle + \langle \nabla h^*(w^{\mathsf{y}}) - \nabla h^*(z^{\mathsf{y}}), w^{\mathsf{y}} - u^{\mathsf{y}}\rangle &= \langle \nabla r(w) - \nabla r(z), w - u\rangle \\ &\le V_z^r(w) + V_w^r(u).\end{aligned}$$

Combining the above two calculations in the context of (A.5) yields the desired claim. □

Theorem 75. *In the setting of Lemma 220, consider running Algorithm 3 on the monotone operator g and the distance generating function r initialized at $z_0 := (x_0, \nabla h(x_0))$, for T iterations. Every iteration consists of solving a constant number of proximal problems in the function ω, and a constant number of d-dimensional vector operations; moreover, we can always maintain each z_t in the form $(x_t, \nabla h(v_t))$ for some explicitly computed $v_t \in \mathbb{R}^d$. Finally, we have*

$$T \ge 4\sqrt{\frac{L}{\mu}} \log\left(\frac{2L}{\mu} \cdot \frac{f(x_0) - f(x^*)}{\epsilon} \right) \implies f(x_T) - f(x^*) \le \epsilon.$$

Proof. We first demonstrate the claimed implementability of steps of Algorithm 3. Note that from the form of iterates in Algorithm 3, each x block of variables indeed results from solving proximal problems in ω. On the y block, the claim of the invariant (that it can be represented as some $\nabla h(v)$ for explicit v) follows identically to the arguments in Lemma 5, where we inductively show that the y block of each z_t, w_t can be maintained as a gradient of h.

Next, we prove the desired error bound. We first compute, for $z^* := (x^*, \nabla h(x^*))$,

$$\begin{aligned}V_{z_0}^r(z^*) &= \mu V_{x_0}^{\omega}(x^*) + V_{\nabla h(x_0)}^{h^*}(\nabla h(x^*)) = \mu V_{x_0}^{\omega}(x^*) + V_{x^*}^h(x_0) \\ &\le V_{x_0}^f(x^*) + V_{x^*}^f(x_0) = \langle \nabla f(x_0) - \nabla f(x^*), x_0 - x^*\rangle \\ &\le L \|x_0 - x^*\|^2 \le \frac{2L}{\mu} \left(f(x_0) - f(x^*) \right).\end{aligned}$$

By applying Proposition 3 with the bounds from Lemmas 220 and 221, we have

$$V_{z_T}^r(z^*) \le \frac{\mu\epsilon}{L}.$$

The conclusion follows from smoothness of f and strong convexity of ω, i.e.

$$\frac{L}{\mu}V^r_{z_T}(z^*) \geq \frac{L}{\mu}V^{\mu\omega}_{x_T}(x^*) \geq \frac{L}{2}\left\|x_T - x^*\right\|^2 \geq f(x_T) - f(x^*).$$

□

A.6 Missing proofs from Section 2.6

Lemma 7. *Let $\bar{w}_t := (x_t + \sum_{i\in[d]} \Delta_t^{(i)}, \nabla f(v_{t+\frac{1}{2}}))$. Then $\forall u$, taking expectations over iteration t,*

$$\mathbb{E}\left[\left\langle g_i(w_t^{(i)}), w_t^{(i)} - u\right\rangle\right] = \left\langle g(\bar{w}_t), \bar{w}_t - u\right\rangle.$$

Proof. Note $v_{t+\frac{1}{2}}$ is deterministic regardless of the sampled $i \in [d]$. Expanding for $u = (u^{\mathsf{x}}, u^{\mathsf{y}})$,

$$\begin{aligned}\mathbb{E}\left[\left\langle g_i(w_t^{(i)}), w_t^{(i)} - u\right\rangle\right] &= \sum_{i\in[d]} p_i \left(\left\langle \frac{1}{p_i}\nabla_i f(v_{t+\frac{1}{2}}), x^{(i)}_{t+\frac{1}{2}} - u^{\mathsf{x}}\right\rangle\right. \\ &\quad \left. + \left\langle v_{t+\frac{1}{2}} - \left(x_t + \frac{1}{p_i}\Delta_t^{(i)}\right), y_{t+\frac{1}{2}} - u^{\mathsf{y}}\right\rangle\right) \\ &= \left\langle g(\bar{w}_t), \bar{w}_t - u\right\rangle.\end{aligned}$$

Here, we used the fact that $\nabla_i f(v_{t+\frac{1}{2}})$ is 1-sparse. □

Lemma 8 (Expected relative Lipschitzness)**.** *Let $\lambda = 1 + S_{1/2}/\sqrt{\mu}$, where $S_{1/2} := \sum_{i\in[d]}\sqrt{L_i}$. Then, for the iterates* (2.21) *with $p_i = \sqrt{L_i}/S_{1/2}$, taking expectations over iteration t,*

$$\mathbb{E}\left[\left\langle g_i(w_t^{(i)}) - g_i\left(z_t\right), w_t^{(i)} - z_{t+1}^{(i)}\right\rangle\right] \leq \lambda\mathbb{E}\left[V^r_{z_t}(w_t^{(i)}) + V^r_{w_t^{(i)}}(z_{t+1}^{(i)})\right].$$

Proof. Equivalently, we wish to show that

$$\mathbb{E}\left[\left\langle g_i(w_t^{(i)}) - g_i\left(z_t\right), w_t^{(i)} - z_{t+1}^{(i)}\right\rangle\right] \leq \left(1 + \frac{S_{1/2}}{\sqrt{\mu}}\right)\mathbb{E}\left[V^r_{z_t}(w_t^{(i)}) + V^r_{w_t^{(i)}}(z_{t+1}^{(i)})\right].$$

The proof is patterned from Lemma 4. By direct calculation, the left hand side is

$$\begin{aligned}&\mathbb{E}\left[\left\langle g_i(w_t^{(i)}) - g_i\left(z_t\right), w_t^{(i)} - z_{t+1}^{(i)}\right\rangle\right] \\ &= \sum_{i\in[d]} p_i\left(\frac{1}{p_i}\left\langle \nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f\left(v_t\right), x^{(i)}_{t+\frac{1}{2}} - x^{(i)}_{t+1}\right\rangle\right. \\ &\left. + \frac{1}{p_i}\left\langle x_t - x^{(i)}_{t+\frac{1}{2}}, \nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f(v^{(i)}_{t+1})\right\rangle\right) \\ &+ \sum_{i\in[d]} p_i\left\langle v_{t+\frac{1}{2}} - v_t, \nabla f(v_{t+\frac{1}{2}}) - \nabla f(v^{(i)}_{t+1})\right\rangle.\end{aligned} \tag{A.6}$$

We first bound the second and third lines of (A.6):

$$\begin{aligned}\frac{1}{p_i}\left(\left\langle \nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f\left(v_t\right), x^{(i)}_{t+\frac{1}{2}} - x^{(i)}_{t+1}\right\rangle + \left\langle x_t - x^{(i)}_{t+\frac{1}{2}}, \nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f(v^{(i)}_{t+1})\right\rangle\right)\\ \leq \frac{S_{1/2}}{\sqrt{\mu}}\left(\frac{\mu}{2}\left\|x^{(i)}_{t+\frac{1}{2}} - x^{(i)}_{t+1}\right\|_2^2 + \frac{1}{2L_i}\left\|\nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f(v^{(i)}_{t+1})\right\|_2^2\right.\\ \left.+\frac{\mu}{2}\left\|x_t - x^{(i)}_{t+\frac{1}{2}}\right\|_2^2 + \frac{1}{2L_i}\left\|\nabla_i f(v_{t+\frac{1}{2}}) - \nabla_i f\left(v_t\right)\right\|_2^2\right)\\ \leq \frac{S_{1/2}}{\sqrt{\mu}}\left(V^r_{z_t}(w^{(i)}_t) + V^r_{w^{(i)}_t}(z^{(i)}_{t+1})\right).\end{aligned} \tag{A.7}$$

The first inequality used the definition $p_i = \sqrt{L_i/S_{1/2}}$ and Cauchy-Schwarz, and the second used strong convexity and Lemma 215. Next, we bound the fourth line of (A.6):

$$\begin{aligned}&\left\langle v_{t+\frac{1}{2}} - v_t, \nabla f(v_{t+\frac{1}{2}}) - \mathbb{E}\left[\nabla f(v^{(i)}_{t+1})\right]\right\rangle\\ &\leq V^{f^*}_{\nabla f(v_t)}\left(\nabla f(v_{t+\frac{1}{2}})\right) + V^{f^*}_{\nabla f(v_{t+\frac{1}{2}})}\left(\mathbb{E}\left[\nabla f(v^{(i)}_{t+1})\right]\right)\\ &\leq V^{f^*}_{\nabla f(v_t)}\left(\nabla f(v_{t+\frac{1}{2}})\right) + \mathbb{E}\left[V^{f^*}_{\nabla f(v_{t+\frac{1}{2}})}\left(\nabla f(v^{(i)}_{t+1})\right)\right]\\ &\leq \mathbb{E}\left[V^r_{z_t}(w^{(i)}_t) + V^r_{w^{(i)}_t}(z^{(i)}_{t+1})\right].\end{aligned}$$

The first inequality is (2.15), the second is convexity of Bregman divergence, and the third used nonnegativity of $\frac{\mu}{2}\|\cdot\|_2^2$. Combining with an expectation over (A.7) yields the claim. □

Lemma 222. *Suppose at step t, for $\mathbf{B_t} \in \mathbb{R}^{2\times 2}$, $p_t, q_t \in \mathbb{R}^d$, we maintain*

$$\begin{pmatrix} x_t & v_t\end{pmatrix} = \begin{pmatrix} p_t & q_t\end{pmatrix}\mathbf{B_t}.$$

Then, for any sampled i, we can compute $\mathbf{B_{t+1}} \in \mathbb{R}^{2\times 2}$, $p_{t+1}, q_{t+1} \in \mathbb{R}^d$ such that

$$\begin{pmatrix} x_{t+1} & v_{t+1}\end{pmatrix} = \begin{pmatrix} p_{t+1} & q_{t+1}\end{pmatrix}\mathbf{B_{t+1}},$$

using two generalized partial derivative oracle queries and constant additional work.

Proof. Suppose on iteration t that coordinate i was sampled. In closed form, we have by the definition of gradient estimators (2.21) the updates to the x component

$$x_{t+\frac{1}{2}} = x_t - \frac{1}{\mu\lambda p_i}\nabla_i f(v_t),\ x_{t+1} = x_t - \frac{1}{\mu\lambda p_i}\nabla_i f(v_{t+\frac{1}{2}}).$$

Thus, if we can maintain the implicit representation of v_t and $v_{t+\frac{1}{2}}$, we can compute these updates

Algorithm 1 EG-COORD-ACCEL(x_0, ϵ): Extragradient accelerated coordinate-smooth minimization

Input: $x_0 \in \mathbb{R}^d$, f $\{L_i\}_{i\in[d]}$-coordinate smooth and μ-s.c. in $\|\cdot\|_2$, and $\epsilon_0 \geq f(x_0) - f(x^*)$
$\lambda \leftarrow 1 + \sum_{i\in[d]} \sqrt{L_i/\mu}$, $T \leftarrow 4\lceil\lambda\rceil$, $K \leftarrow \lceil \log_2 \frac{\epsilon_0}{\epsilon}\rceil$, $\mathbf{A} \leftarrow \begin{pmatrix} 1 & \frac{1}{\kappa} - \frac{1}{\kappa^2} \\ 0 & 1 - \frac{1}{\kappa} + \frac{1}{\kappa^2} \end{pmatrix}$
$p_0 \leftarrow x_0$, $q_0 \leftarrow x_0$, $\mathbf{B_0} \leftarrow \mathbf{I}_{2\times 2}$
for $0 \leq k < K$ **do**
  Sample τ uniformly in $[0, T-1]$
  for $0 \leq t < \tau$ **do**
    Sample $i \propto \sqrt{L_i}$
    Compute $\nabla_i f(v_t)$, $\nabla_i f((1-\lambda^{-1})v_t + \lambda^{-1}x_t)$ via generalized partial derivative oracle
    $s_t \leftarrow \begin{pmatrix} \frac{1}{\mu\lambda p_i}\nabla_i f((1-\lambda^{-1})v_t + \lambda^{-1}x_t) & \frac{1}{\mu\lambda^2 p_i^2}\nabla_i f(v_t) \end{pmatrix}$
    $\mathbf{B_{t+1}} \leftarrow \mathbf{B_t}\mathbf{A}$, $\begin{pmatrix} p_{t+1} & q_{t+1} \end{pmatrix} \leftarrow \begin{pmatrix} p_t & q_t \end{pmatrix} - s_t \mathbf{B_{t+1}}^{-1}$
  end for
  $\mathbf{B_0} \leftarrow \begin{pmatrix} [\mathbf{B}_\tau]_{12} & [\mathbf{B}_\tau]_{12} \\ [\mathbf{B}_\tau]_{22} & [\mathbf{B}_\tau]_{22} \end{pmatrix}$, $p_0 \leftarrow p_\tau$, $q_0 \leftarrow q_\tau$
end for
return $[\mathbf{B}_\tau]_{12} p_\tau + [\mathbf{B}_\tau]_{22} q_\tau$

via a partial derivative oracle. Next, the proof of Lemma 5 implies in closed form

$$v_{t+\frac{1}{2}} = \left(1 - \frac{1}{\lambda}\right) v_t + \frac{1}{\lambda} x_t,$$

$$v_{t+1} = v_t + \frac{1}{\lambda}\left(x_t + \frac{1}{p_i}\Delta_t^{(i)} - v_{t+\frac{1}{2}}\right) = \left(1 - \frac{1}{\lambda} + \frac{1}{\lambda^2}\right) v_t + \left(\frac{1}{\lambda} - \frac{1}{\lambda^2}\right) x_t - \frac{1}{\mu\lambda^2 p_i^2}\nabla_i f(v_t),$$

where we have already computed $\nabla_i f(v_t)$. Therefore, the update can be written as, for 2-sparse s_t,

$$\begin{pmatrix} x_{t+1} & v_{t+1} \end{pmatrix} = \begin{pmatrix} x_t & v_t \end{pmatrix} \mathbf{A} - s_t$$

$$\text{where } \mathbf{A} := \begin{pmatrix} 1 & \frac{1}{\kappa} - \frac{1}{\kappa^2} \\ 0 & 1 - \frac{1}{\kappa} + \frac{1}{\kappa^2} \end{pmatrix}, \; s_t := \begin{pmatrix} \frac{1}{\mu\lambda p_i}\nabla_i f(v_{t+\frac{1}{2}}) & \frac{1}{\mu\lambda^2 p_i^2}\nabla_i f(v_t) \end{pmatrix}.$$

We can therefore maintain this implicitly via the equivalent step (doing constant extra work),

$$\mathbf{B_{t+1}} = \mathbf{B_t}\mathbf{A}, \; \begin{pmatrix} p_{t+1} & q_{t+1} \end{pmatrix} = \begin{pmatrix} p_t & q_t \end{pmatrix} - s_t \mathbf{B_{t+1}}^{-1}.$$

□

A.7 Optimism

In this section, we discuss the relationship of our condition (relative Lipschitzness) with the condition of "optimism" proposed by [452], which is known to recover the mirror prox algorithm for Lipschitz operators. Specifically, the optimistic mirror descent procedure of [452] iterates the following steps (see Equation 1 of their paper), for strongly convex r:

$$w_t \leftarrow \mathrm{Prox}^r_{z_t}\left(\eta M_t\right), \; z_{t+1} \leftarrow \mathrm{Prox}^r_{z_t}\left(\eta g(w_t)\right). \tag{A.8}$$

Here, M_t is a vector which ideally "resembles" the point $\eta g(w_t)$, capturing a notion of "predictability." Under Lipschitzness of g, [452] notes that simply choosing $M_t = g(z_t)$ and $\eta = \frac{1}{L}$ yields an algorithm identical to Algorithm 1, where the notion of predictable sequences comes from stability of g. Rakhlin and Sridharan prove the following claim about their procedure (Lemma 1, [452]):

Proposition 54 (Optimistic mirror descent)**.** *For $g : \mathcal{Z} \to \mathcal{Z}^*$ and any $u \in \mathcal{Z}$,*

$$\sum_{0 \le t < T} \langle g(w_t), w_t - u \rangle \le \frac{V^r_{z_0}(u)}{\eta} + \sum_{0 \le t < T} \|M_t - g(w_t)\|_* \|w_t - z_{t+1}\| - \frac{1}{2\eta}\left(\|w_t - z_t\|^2 + \|w_t - z_{t+1}\|^2\right).$$

Indeed, Proposition 54 follows by the proofs we give of Proposition 1 and Lemma 2, where we apply Cauchy-Schwarz and strong convexity of r. However, crucially our proof in Section 2.4 of the accelerated rate bypasses this direct application of Cauchy-Schwarz to the quantity $M_t - g(w_t)$, and couples terms in a way which can yield a tighter relative Lipschitz parameter (see Lemma 4). In this sense, relative Lipschitzness can sharpen the rate of convergence for optimistic mirror descent achieved by [452], which was crucial for our improved guarantees.

A.8 Reducing strongly monotone problems to regularized subproblems

A.8.1 Convex optimization

We give the following generic reduction for strongly convex optimization in the form of an algorithm. Similar reductions are standard in the literature [227], but we include the algorithm and full analysis here for completeness.

Lemma 223. *In Algorithm 66, letting $x_\star$ minimize f, we have for every $k \in [K]$:*

$$\mathbb{E} V_{x_k}(x_\star) \le \frac{1}{2^k} V_{x_0}(x_\star).$$

Algorithm 66: REDX-CONVEX: Strongly convex optimization reduction

Input: μ-strongly convex $f : \mathcal{X} \to \mathbb{R}$, $x_0 \in \mathcal{X}$
Parameter(s): $K \in \mathbb{N}$
for $0 \le k < K$ **do**
  $x_{k+1} \leftarrow$ any (possibly random) point satisfying

$$\mathbb{E}V_{x_{k+1}}(x^\star_{k+1}) \le \frac{1}{4}V_{x_k}(x^\star_{k+1}), \text{ where } x^\star_{k+1} := \operatorname{argmin}_{x\in\mathcal{X}} f(x) + \frac{\mu}{4}V_{x_k}(x)$$

Proof. By applying the optimality condition on $x^\star_{k+1}$, strong convexity of f, and (2.10),

$$\begin{aligned}
\left\langle \nabla f(x^\star_{k+1}), x^\star_{k+1} - x_\star \right\rangle &\le \frac{\mu}{4}\left\langle x_k - x^\star_{k+1}, x^\star_{k+1} - x_\star \right\rangle \\
\implies \mu V_{x^\star_{k+1}}(x_\star) &\le f(x^\star_{k+1}) - f(x_\star) \\
&\le \left\langle \nabla f(x^\star_{k+1}), x^\star_{k+1} - x_\star \right\rangle \\
&\le \frac{\mu}{4}V_{x_k}(x_\star) - \frac{\mu}{4}V_{x^\star_{k+1}}(x_\star) - \frac{\mu}{4}V_{x_k}(x^\star_{k+1}).
\end{aligned}$$

Further by the triangle inequality and $(a+b)^2 \le 2a^2 + 2b^2$, we have

$$V_{x_{k+1}}(x_\star) \le 2V_{x_{k+1}}(x^\star_{k+1}) + 2V_{x^\star_{k+1}}(x_\star).$$

Hence, combining these pieces,

$$\begin{aligned}
\mathbb{E}V_{x_{k+1}}(x_\star) &\le 2V_{x^\star_{k+1}}(x_\star) + 2\mathbb{E}V_{x_{k+1}}(x^\star_{k+1}) \\
&\le 2V_{x^\star_{k+1}}(x_\star) + \frac{1}{2}V_{x_k}(x^\star_{k+1}) \\
&\le \frac{1}{2}V_{x_k}(x_\star) - \frac{1}{2}V_{x^\star_{k+1}}(x_\star) \le \frac{1}{2}V_{x_k}(x_\star).
\end{aligned}$$

□

We apply this reduction in order to prove Corollary 4.

Corollary 4. *Suppose the summands $\{f_i\}_{i\in[n]}$ in (2.36) satisfy Assumption 2, and F_{fs} is μ-strongly convex with minimizer $x_\star$. Further, suppose we have $x_0 \in \mathcal{X}$ such that $F_{\text{fs}}(x_0) - F_{\text{fs}}(x_\star) \le \epsilon_0$. Algorithm 66 using Algorithm 6 to implement steps returns $x \in \mathcal{X}$ with $\mathbb{E}F_{\text{fs}}(x) - F_{\text{fs}}(x_\star) \le \epsilon$ in N_{tot} iterations, using a total of $O(N_{\text{tot}})$ gradient calls each to some f_i for $i \in [n]$, where*

$$N_{\text{tot}} = O\left(\kappa_{\text{fs}} \log\left(\frac{\kappa_{\text{fs}}\epsilon_0}{\epsilon}\right)\right), \text{ for } \kappa_{\text{fs}} := n + \sum_{i\in[n]} \frac{\sqrt{L_i}}{\sqrt{n\mu}}.$$

Proof. The overhead K is asymptotically the same here as the parameter T in Theorem 8, by

analogous smoothness and strong convexity arguments. Moreover, we use Theorem 8 to solve each subproblem required by Algorithm 66; in particular, the subproblem is equivalent to approximately minimizing $F_{\text{fs}} + \frac{\mu}{8}\|\cdot\|^2$, up to a linear shift which does not affect any smoothness bounds, and a constant in the strong convexity. We note that we will initialize the subproblem solver in iteration k with x_k. We hence can set $T = 2$ and $S = O(\kappa_{\text{fs}})$, yielding the desired iteration bound. □

A.8.2 Convex-concave optimization

We give the following generic reduction for strongly convex-concave optimization in the form of an algorithm. For simplicity in this section, we define for $z = (z^{\mathsf{x}}, z^{\mathsf{y}}) \in \mathcal{X} \times \mathcal{Y}$,

$$\omega(z) := \frac{\mu^{\mathsf{x}}}{2}\|z^{\mathsf{x}}\|^2 + \frac{\mu^{\mathsf{y}}}{2}\|z^{\mathsf{y}}\|^2.$$

Algorithm 67: REDX-MINIMAX: Reduction for minimax

Input: $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that $F(\cdot, y)$ is μ^{x}-strongly convex for all $y \in \mathcal{Y}$ and $F(x, \cdot)$ μ^{y}-strongly concave for all $x \in \mathcal{X}$, $z_0 \in \mathcal{X} \times \mathcal{Y}$

Parameter(s): $K \in \mathbb{N}$

for $0 \le k < K$ **do**

$z_{k+1} \leftarrow$ any (possibly random) point satisfying

$$\mathbb{E}\left[V^{\omega}_{z_{k+1}}\left(z^\star_{k+1}\right)\right] \le \frac{1}{4}\left(V^{\omega}_{z_k}\left(z^\star_{k+1}\right)\right),$$

$$\text{where } z^\star_{k+1} := \operatorname{argmin}_{z^{\mathsf{x}} \in \mathcal{X}} \operatorname{argmax}_{z^{\mathsf{y}} \in \mathcal{Y}} F(z^{\mathsf{x}}, z^{\mathsf{y}}) + \frac{\mu^{\mathsf{x}}}{4} V_{z^{\mathsf{x}}_k}(z^{\mathsf{x}}) - \frac{\mu^{\mathsf{y}}}{4} V_{z^{\mathsf{y}}_k}(z^{\mathsf{y}})$$

Lemma 224. *In Algorithm 67, letting $(x_\star, y_\star)$ be the saddle point of F, we have for every $k \in [K]$:*

$$\mathbb{E}\left[V^{\omega}_{z_k}(z_\star)\right] \le \frac{1}{2^k} V^{\omega}_{z_0}(z_\star).$$

Proof. By applying the optimality conditions on $z^\star_{k+1}$, strong convexity-concavity of F, and (2.10), and letting Φ^F be the gradient operator of F,

$$\begin{aligned}
\left\langle \Phi^F(z^\star_{k+1}), z^\star_{k+1} - z_\star \right\rangle &\le \frac{\mu^{\mathsf{x}}}{4}\left\langle z^{\mathsf{x}}_k - [z^\star_{k+1}]^{\mathsf{x}}, [z^\star_{k+1}]^{\mathsf{x}} - z^{\mathsf{x}}_\star \right\rangle \\
&\quad + \frac{\mu^{\mathsf{y}}}{4}\left\langle z^{\mathsf{y}}_k - [z^\star_{k+1}]^{\mathsf{y}}, [z^\star_{k+1}]^{\mathsf{y}} - z^{\mathsf{y}}_\star \right\rangle \\
\implies V^{\omega}_{z^\star_{k+1}}(z_\star) &\le \left\langle \Phi^F(z^\star_{k+1}), z^\star_{k+1} - z_\star \right\rangle \\
&\le \frac{1}{4} V^{\omega}_{z_k}(z_\star) - \frac{1}{4} V^{\omega}_{z^\star_{k+1}}(z_\star) - \frac{1}{4} V_{z_k}(z^\star_{k+1}).
\end{aligned}$$

Further by the triangle inequality and $(a+b)^2 \leq 2a^2 + 2b^2$, we have

$$V^{\omega}_{z_{k+1}}(z_\star) \leq 2V^{\omega}_{z_{k+1}}(z^\star_{k+1}) + 2V^{\omega}_{z^\star_{k+1}}(z_\star).$$

Hence, combining these pieces,

$$\begin{aligned}\mathbb{E}V^{\omega}_{z_{k+1}}(z_\star) &\leq 2V^{\omega}_{z^\star_{k+1}}(z_\star) + 2\mathbb{E}V^{\omega}_{z_{k+1}}(z^\star_{k+1}) \\ &\leq 2V^{\omega}_{z^\star_{k+1}}(z_\star) + \frac{1}{2}V^{\omega}_{z_k}(z^\star_{k+1}) \\ &\leq \frac{1}{2}V^{\omega}_{z_k}(z_\star) - \frac{1}{2}V^{\omega}_{z^\star_{k+1}}(z_\star) \leq \frac{1}{2}V^{\omega}_{z_k}(z_\star).\end{aligned}$$

□

We apply this reduction in order to prove Corollary 5.

Corollary 5. *Suppose the summands $\{f_i, g_i, h_i\}_{i\in[n]}$ in (2.49) satisfy Assumption 4, and F_{mmfs} is μ^{x}-strongly convex in x, μ^{y}-strongly convex in y, with saddle point $(x_\star, y_\star)$. Further, suppose we have $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ such that $\text{Gap}_{F_{\text{mmfs}}}(x_0, y_0) \leq \epsilon_0$. Algorithm 66 using Algorithm 8 and 9 to implement steps returns $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathbb{E}\text{Gap}(x, y) \leq \epsilon$ in N_{tot} iterations, using a total of $O(N_{\text{tot}})$ gradient calls each to some f_i, g_i, or h_i for $i \in [n]$, where*

$$N_{\text{tot}} = O\left(\kappa_{\text{mmfs}} \log(\kappa_{\text{mmfs}}) \log\left(\frac{\kappa_{\text{mmfs}}\epsilon_0}{\epsilon}\right)\right),$$
$$\text{for } \kappa_{\text{mmfs}} := n + \frac{1}{\sqrt{n}} \sum_{i\in[n]} \left(\sqrt{\frac{L^{\text{x}}_i}{\mu^{\text{x}}}} + \sqrt{\frac{L^{\text{y}}_i}{\mu^{\text{y}}}} + \frac{\Lambda^{\text{xx}}_i}{\mu^{\text{x}}} + \frac{\Lambda^{\text{xx}}_i}{\sqrt{\mu^{\text{x}}\mu^{\text{y}}}} + \frac{\Lambda^{\text{yy}}_i}{\mu^{\text{y}}}\right).$$

Proof. The overhead K is asymptotically the same here as the logarithmic term in the parameter T in Theorem 9, by analogous smoothness and strong convexity arguments. Moreover, we use Theorem 9 with μ^{x}, μ^{y} rescaled by constants to solve each subproblem required by Algorithm 67; in particular, the subproblem is equivalent to approximately finding a saddle point to $F_{\text{fs}}(z) + \frac{\mu^{\text{x}}}{8}\|z^{\text{x}}\|^2 - \frac{\mu^{\text{y}}}{8}\|z^{\text{y}}\|^2$, up to a linear shift which does not affect any smoothness bounds. We note that we will initialize the subproblem solver in iteration k with z_k. We hence can set $T = O(\gamma)$, yielding the desired iteration bound. □

A.9 Helper facts

Here we state two helper facts that are used throughout the analysis, for completeness of the paper. The first gives a few properties on monotone operators. We first recall by definition, an operator

$\Phi : \mathcal{Z} \to \mathcal{Z}^*$ is monotone if

$$\langle \Phi(z) - \Phi(z'), z - z' \rangle \geq 0, \quad \text{for all } z', z' \in \mathcal{Z}.$$

An operator Φ is m-strongly monotone with respect to convex $r : \mathcal{Z} \to \mathbb{R}$ if for all $z, z' \in \mathcal{Z}$,

$$\langle \Phi(z) - \Phi(z'), z - z' \rangle \geq m \langle \nabla r(z) - \nabla r(z'), z - z' \rangle, \quad \text{for all } z', z' \in \mathcal{Z}.$$

We state the following standard facts about monotone operators and their specialization to convex-concave functions, and include references or proofs for completeness.

Fact 35. *The following facts about monotone operators hold true:*

1. *Given a convex function $f(x) : \mathcal{X} \to \mathbb{R}$, its induced operator $\Phi = \nabla f : \mathcal{X} \to \mathcal{X}^*$ is monotone.*
2. *Given a convex-concave function $h(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $\Phi(x, y) = (\nabla_x h(x, y), -\nabla_y h(x, y)) : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}^* \times \mathcal{Y}^*$ is monotone.*
3. *Given a convex function f, its induced operator $\Phi = \nabla f$ is 1-strongly monotone with respect to itself.*
4. *Monotonicity is preserved under addition: For any $m, m' \geq 0$, if Φ is m-strongly monotone and Ψ is m'-strongly monotone with respect to convex r, then $\Phi + \Psi$ is $(m + m')$-strongly monotone with respect to r.*

Proof. The first two items are basic fact of convexity and minimax optimization [460]. For the third item, we note that for any $x, x' \in \mathcal{X}$

$$\langle \Phi(x) - \Phi(x'), x - x' \rangle = \langle \nabla f(x) - \nabla f(x'), x - x' \rangle,$$

which satisfies 1-strong monotonicity with respect to f by definition.

For the fourth item, we note that for any $m, m' \geq 0$ and assumed Φ, Ψ,

$$\begin{aligned} \langle \Phi(z) - \Phi(z'), z - z' \rangle &\geq m \langle \nabla r(z) - \nabla r(z'), z - z' \rangle, \\ \langle \Psi(z) - \Psi(z'), z - z' \rangle &\geq m' \langle \nabla r(z) - \nabla r(z'), z - z' \rangle, \\ \implies \langle \Phi(z) + \Psi(z) - (\Phi(z') + \Psi(z')), z - z' \rangle &\geq (m + m') \langle \nabla r(z) - \nabla r(z'), z - z' \rangle. \end{aligned}$$

□

These facts about monotone operators find usage in proving (relative) strong monotonicity of our operators; see Lemma 11, 18 and 27 in the main paper.

The second fact bounds the smoothness of best-response function of some given convex-concave function $h; \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. We refer readers to Fact 1 of [533] for a complete proof.

Fact 36 (Fact 1, [533]). *Suppose h satisfies the blockwise-smoothness properties: for all $u, v \in \mathcal{X} \times \mathcal{Y}$,*

$$\begin{aligned} \|\nabla_x h(u) - \nabla_x h(v)\| &\le \Lambda^{\mathsf{xx}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda^{\mathsf{xx}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\|, \\ \|\nabla_y h(u) - \nabla_y h(v)\| &\le \Lambda^{\mathsf{xx}} \|u^{\mathsf{x}} - v^{\mathsf{x}}\| + \Lambda^{\mathsf{yy}} \|u^{\mathsf{y}} - v^{\mathsf{y}}\|, \end{aligned} \tag{A.9}$$

and suppose h is μ^{x}-strongly convex in x and μ^{y}-strongly concave in y. The best response function $h^{\mathsf{y}}(x) := \max_{y \in \mathcal{Y}} h(x, y)$ is μ^{x}-strongly convex and $\Lambda^{\mathsf{xx}} + \frac{(\Lambda^{\mathsf{xx}})^2}{\mu^{\mathsf{y}}}$-smooth, and $h^{\mathsf{x}}(y) := \min_{x \in \mathcal{Y}} h(x, y)$ is μ^{y}-strongly concave and $\Lambda^{\mathsf{yy}} + \frac{(\Lambda^{\mathsf{xx}})^2}{\mu^{\mathsf{x}}}$-smooth.

We use this fact when converting radius bounds to duality gap bounds in Lemma 15 and 16.

A.10 Proofs for Section 2.9

A.10.1 Proofs for Section 2.9.2

Proposition 5 (Partial variance analysis of randomized mirror prox). *Suppose (possibly random) $\widetilde{\Phi}$ is defined so that in each iteration s, for all $u \in \mathcal{Z}$ and all $\rho > 0$, there exists a (possibly random) point $\bar{w}_s \in \mathcal{Z}$ and a γ-strongly monotone operator $\Phi : \mathcal{Z} \to \mathcal{Z}^*$ (with respect to r) such that*

$$\begin{aligned} \mathbb{E}\left[\left\langle \widetilde{\Phi}(w_{s+1/2}), w_{s+1/2} - w_\star \right\rangle\right] &= \mathbb{E}\left[\langle \Phi(\bar{w}_s), \bar{w}_s - w_\star \rangle\right], \\ \mathbb{E}\left[\left\langle \widetilde{\Phi}(w_{s+1/2}) - \widetilde{\Phi}(w_s), w_{s+1/2} - w_{s+1} \right\rangle\right] &\le \left(\lambda_0 + \frac{1}{\rho}\right) \mathbb{E}\left[V^r_{w_s}(w_{s+1/2}) + V^r_{w_{s+1/2}}(w_{s+1})\right] \\ &\quad + \rho\lambda_1 \mathbb{E}\left[V^r_{w_0}(w_\star) + V^r_{\bar{w}_s}(w_\star)\right], \end{aligned} \tag{2.56}$$

where $w_\star$ solves the VI in Φ. Then by setting

$$\rho \leftarrow \frac{\gamma}{5\lambda_1},\ \lambda \leftarrow \lambda_0 + \frac{1}{\rho},\ T \leftarrow \frac{5\lambda}{\gamma} = \frac{5\lambda_0}{\gamma} + \frac{25\lambda_1}{\gamma^2},$$

in Algorithm 5, and returning $\bar{w}_\sigma$ for $0 \le \sigma < S$ sampled uniformly at random,

$$\mathbb{E}\left[V^r_{\bar{w}_\sigma}(w_\star)\right] \le \frac{1}{2} V^r_{w_0}(w_\star).$$

Proof. First, consider a single iteration $0 \le s < S$, and fix the point w_s in Algorithm 5. By the optimality conditions on $w_{s+1/2}$ and w_{s+1}, we have

$$\begin{aligned} \frac{1}{\lambda}\left\langle \widetilde{\Phi}(w_s), w_{s+1/2} - w_{s+1} \right\rangle &\le V^r_{w_s}(w_{s+1}) - V^r_{w_{s+1/2}}(w_{s+1}) - V^r_{w_s}(w_{s+1/2}), \\ \frac{1}{\lambda}\left\langle \widetilde{\Phi}(w_{s+1/2}), w_{s+1} - w_\star \right\rangle &\le V^r_{w_s}(w_\star) - V^r_{w_{s+1}}(w_\star) - V^r_{w_s}(w_{s+1}). \end{aligned}$$

Summing the above, rearranging, and taking expectations yields

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{\lambda}\left\langle\Phi(\bar{w}_s), \bar{w}_s - w_\star\right\rangle\right] &= \mathbb{E}\left[\frac{1}{\lambda}\left\langle\widetilde{\Phi}(w_{s+1/2}), w_{s+1/2} - w_\star\right\rangle\right] \\
&\leq \mathbb{E}\left[V^r_{w_s}(w_\star) - V^r_{w_{s+1}}(w_\star)\right] \\
&+ \mathbb{E}\left[\frac{1}{\lambda}\left\langle\widetilde{\Phi}(w_{s+1/2}) - \widetilde{\Phi}(w_s), w_{s+1/2} - w_{s+1}\right\rangle - V^r_{w_s}(w_{s+1/2}) + V^r_{w_{s+1/2}}(w_{s+1})\right] \\
&\leq \mathbb{E}\left[V^r_{w_s}(w_\star) - V^r_{w_{s+1}}(w_\star)\right] + \frac{\rho\lambda_1}{\lambda}\mathbb{E}\left[V^r_{w_0}(w_\star) + V^r_{\bar{w}_s}(w_\star)\right].
\end{aligned}$$

In the last line we used the assumption (2.56). Since $w_\star$ solves the VI in Φ, adding $\mathbb{E}\frac{1}{\lambda}\left\langle\Phi(w_\star), w_\star - \bar{w}_s\right\rangle$ to the left-hand side above and applying strong monotonicity of g in r yields

$$\mathbb{E}\left[\frac{1}{\lambda}V^r_{\bar{w}_s}(w_\star)\right] \leq \mathbb{E}\left[V^r_{w_s}(w_\star) - V^r_{w_{s+1}}(w_\star)\right] + \frac{\rho\lambda_1}{\lambda}\mathbb{E}\left[V^r_{w_0}(w_\star) + V^r_{\bar{w}_s}(w_\star)\right].$$

Telescoping the above for $0 \leq s < S$ and using nonnegativity of Bregman divergences yields

$$(\gamma - \rho\lambda_1)\,\mathbb{E}\left[\frac{1}{T}\sum_{0\leq t<T} V^r_{\bar{w}_s}(w_\star)\right] \leq \left(\frac{\lambda}{T} + \rho\lambda_1\right) V^r_{w_0}(w_\star).$$

Substituting our choices of $\bar{w}_s$, ρ, λ, and T yields the claim. □

Lemma 25. *Define* $\{\Phi_{jk\ell}, \Phi_{jk\ell'}\} : \mathcal{Z} \to \mathcal{Z}^*$ *as in* (2.58), (2.59), *and the random "aggregate point"* $\bar{w}(\ell)$ *as in* (2.60). *Then, for all* $u \in \mathcal{Z}$, *recalling the definition of* $\Phi = \Phi^{\text{mmfs-pd}} + \gamma(\nabla r - \nabla r(\bar{z}))$ *from* (2.55),

$$\mathbb{E}\left[\left\langle\Phi_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)), w_{\mathsf{aux}}(jk\ell) - u\right\rangle\right] = \mathbb{E}_{\ell\sim r}\left[\left\langle\Phi(\bar{w}(\ell)), \bar{w}(\ell) - u\right\rangle\right].$$

Proof. We demonstrate this equality for the $\mathcal{X}$ and $(\mathcal{X}^*)^n$ blocks; the others (the $\mathcal{Y}$ and $(\mathcal{Y}^*)^n$ blocks) follow symmetrically. We will use the definitions of Φ^h and Φ^{bilin} from (2.54).

$\mathcal{X}$ block. Fix $\ell \in [n]$. We first observe that

$$\begin{aligned}
\mathbb{E}_{\ell'\sim r}\left[\left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{x}}\right] &= \left[\Phi^h(\bar{w}(\ell))\right]^{\mathsf{x}}, \\
\mathbb{E}_{\ell'\sim r}\left[\left[\Phi^{\text{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{x}}\right] &= (1+\gamma)\left[\nabla r(\bar{w}(\ell))\right]^{\mathsf{x}} - \gamma\left[\nabla r(\bar{z})\right]^{\mathsf{x}}.
\end{aligned}$$

Moreover, by expanding the expectation over $j \sim p$,

$$\begin{aligned}
\mathbb{E}_{j\sim p}\left[\left\langle\left[\Phi^{\text{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{x}}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - u^{\mathsf{x}}\right\rangle\right] &= \left\langle\frac{1}{n}\sum_{j\in[n]}(w^{\mathsf{f}^*_j} + \Delta^{\mathsf{x}}(j)), w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - u^{\mathsf{x}}\right\rangle \\
&= \left\langle\left[\Phi^{\text{bilin}}(\bar{w}(\ell))\right]^{\mathsf{x}}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - u^{\mathsf{x}}\right\rangle.
\end{aligned}$$

Summing, we conclude that for fixed ℓ and taking expectations over j, k, ℓ',

$$\mathbb{E}\left[\left\langle [\Phi_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))]^{\mathsf{x}}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - u^{\mathsf{x}}\right\rangle\right] = \left\langle [\Phi(\bar{w}(\ell))]^{\mathsf{x}}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - u^{\mathsf{x}}\right\rangle.$$

The conclusion for the $\mathcal{X}$ block follows by taking expectations over ℓ.

$\mathcal{X}^*$ blocks. Note that the $[\Phi^h_{jk\ell'}]^{\mathsf{f}^*}$ blocks are always zero. Next, for the $[\Phi^{\mathrm{sep}}_{jk\ell'}]^{\mathsf{f}^*}$ component, by expanding the expectation over $j \sim p$ and taking advantage of sparsity, for any $\ell \in [n]$,

$$\begin{aligned}
&\mathbb{E}_{j\sim p}\left[\left\langle \left[\Phi^{\mathrm{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{f}^*}, w^{\mathsf{f}^*}_{\mathsf{aux}}(jk\ell) - u^{\mathsf{f}^*}\right\rangle\right] \\
&\quad = (1+\gamma)\sum_{j\in[n]}\left\langle \frac{1}{n}\nabla f^*_j\left(w^{\mathsf{f}^*_{\mathsf{j}}}_{\mathsf{aux}}\right), w^{\mathsf{f}^*_{\mathsf{j}}}_{\mathsf{aux}} - u^{\mathsf{f}^*_{\mathsf{j}}}\right\rangle \\
&\qquad -\gamma\sum_{j\in[n]}\left\langle \frac{1}{n}\nabla f^*_j\left(\bar{z}^{\mathsf{f}^*_{\mathsf{j}}}\right), w^{\mathsf{f}^*_{\mathsf{j}}}_{\mathsf{aux}} - u^{\mathsf{a}_{\mathsf{j}}}\right\rangle \\
&= \left\langle (1+\gamma)\left[\nabla r(\bar{w}(\ell))\right]^{\mathsf{f}^*} - \gamma\left[\nabla r(\bar{z})\right]^{\mathsf{f}^*}, \bar{w}^{\mathsf{f}^*}(\ell) - u^{\mathsf{f}^*}\right\rangle.
\end{aligned}$$

Here, we recall $\mathsf{f}^*_{\mathsf{j}}$ denotes the block corresponding to the j^{th} copy of $\mathcal{X}^*$. Finally, for the $[\Phi^{\mathrm{bilin}}_{jk\ell'}]^{\mathsf{f}^*}$ component, fix $\ell \in [n]$. Expanding the expectation over $j \sim p$ and taking advantage of sparsity,

$$\mathbb{E}_{j\sim p}\left[\left\langle \left[\Phi^{\mathrm{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))\right]^{\mathsf{f}^*}, [w_{\mathsf{aux}}(jk\ell)]^{\mathsf{f}^*} - u^{\mathsf{f}^*}\right\rangle\right] = \left\langle \left[\Phi^{\mathrm{bilin}}(\bar{w}(\ell))\right]^{\mathsf{f}^*}, \bar{w}^{\mathsf{f}^*}(\ell) - u^{\mathsf{f}^*}\right\rangle.$$

Summing, we conclude that for fixed ℓ and taking expectations over j, k, ℓ',

$$\mathbb{E}\left\langle [g_{jk\ell'}(w_{\mathsf{aux}}(jk\ell))]^{\mathsf{f}^*}, [w_{\mathsf{aux}}(jk\ell)]^{\mathsf{f}^*} - u^{\mathsf{f}^*}\right\rangle = \left\langle [g_{\mathrm{tot}}(\bar{w}(\ell))]^{\mathsf{f}^*}, \bar{w}^{\mathsf{f}^*}(\ell) - u^{\mathsf{f}^*}\right\rangle.$$

The conclusion for the $\mathcal{X}^*$ blocks follows by taking expectations over ℓ. □

Lemma 26. *Lines 6 to 18 of Algorithm 9 implement Algorithm 5 on $(\{\widetilde{\Phi}\}, r)$ defined in (2.58), (2.59), (2.53), for σ iterations, and returns $\bar{w}_\sigma$, following the definition (2.60). Each iteration $s > 0$ is implementable in $O(1)$ gradient calls to some $\{f_j, g_k, h_l\}$, and $O(1)$ vector operations on $\mathcal{X}$ and $\mathcal{Y}$.*

Proof. Let $\{w_s, w_{s+1/2}\}_{0\le s\le\sigma}$ be the iterates of Algorithm 5. We will inductively show that some run of Lines 6 to 18 in Algorithm 9 preserves the invariants

$$\begin{aligned}
w_s &= \left(w^{\mathsf{x}}_s, w^{\mathsf{y}}_s, \left\{\nabla f_i(w^{\mathsf{f}_{\mathsf{i}}}_s)\right\}_{i\in[n]}, \left\{\nabla f_i(w^{\mathsf{g}_{\mathsf{i}}}_s)\right\}_{i\in[n]}\right), \\
w_{s+1/2} &= \left(w^{\mathsf{x}}_{s+1/2}, w^{\mathsf{y}}_{s+1/2}, \left\{\nabla f_i(w^{\mathsf{f}_{\mathsf{i}}}_{s+1/2})\right\}_{i\in[n]}, \left\{\nabla f_i(w^{\mathsf{g}_{\mathsf{i}}}_{s+1/2})\right\}_{i\in[n]}\right)
\end{aligned}$$

for all $0 \leq s \leq \sigma$. Once we prove this claim, it is clear that Lines 6 to 18 in Algorithm 9 implements Algorithm 5 and returns $\bar{w}_\sigma$, upon recalling the definitions (2.58), (2.59), (2.53), and (2.60).

The base case of our induction follows from the way w_0 is initialized in Line 18. Next, suppose for some $0 \leq s \leq \sigma$, our inductive claim holds. By the update in Line 9 of Algorithm 9, if $j \in [n]$ was sampled in iteration s, using the first item in Fact 1,

$$\begin{aligned} w_{s+1/2}^{\mathsf{f}_\mathsf{j}^*} &\leftarrow \operatorname{argmin}_{w^{\mathsf{f}_\mathsf{j}^*} \in \mathcal{X}^*} \left\{ \left\langle \frac{1}{n\lambda p_j} \left((1+\gamma) w_s^{\mathsf{f}_\mathsf{j}} - \gamma \bar{z}^{\mathsf{f}_\mathsf{j}} - w_s^{\mathsf{x}} \right), w^{\mathsf{f}_\mathsf{j}^*} \right\rangle + V_{w_s^{\mathsf{f}_\mathsf{j}^*}}^{f_j^*} \left(w^{\mathsf{f}_\mathsf{j}^*} \right) \right\} \\ &= \nabla f_j \left(w_s^{\mathsf{f}_\mathsf{j}} - \frac{1}{n\lambda p_j} \left((1+\gamma) w_s^{\mathsf{f}_\mathsf{j}} - \gamma \bar{z}^{\mathsf{f}_\mathsf{j}} - w_s^{\mathsf{x}} \right) \right). \end{aligned}$$

Similarly, by the update in Line 10, if $k \in [n]$ was sampled in iteration s,

$$\begin{aligned} w_{s+1/2}^{\mathsf{g}_\mathsf{k}^*} &\leftarrow \operatorname{argmin}_{w^{\mathsf{g}_\mathsf{k}^*}} \left\langle \frac{1}{n\lambda q_k} \left((1+\gamma) w_s^{\mathsf{g}_\mathsf{k}} - \gamma \bar{z}^{\mathsf{g}_\mathsf{k}} - w_s^{\mathsf{y}} \right), w^{\mathsf{g}_\mathsf{k}^*} \right\rangle + V_{w_s^{\mathsf{g}_\mathsf{k}^*}}^{g_k^*} \left(w^{\mathsf{g}_\mathsf{k}^*} \right). \\ &= \nabla g_k \left(w_s^{\mathsf{g}_\mathsf{k}} - \frac{1}{n\lambda q_k} \left((1+\gamma) w_s^{\mathsf{g}_\mathsf{k}} - \gamma \bar{z}^{\mathsf{g}_\mathsf{k}} - w_s^{\mathsf{y}} \right) \right). \end{aligned}$$

Hence, the updates to $w_{s+1/2}^{\mathsf{f}_\mathsf{j}^*}$ and $w_{s+1/2}^{\mathsf{g}_\mathsf{k}^*}$ preserve our invariant, and all other $w_{s+1/2}^{\mathsf{f}_\mathsf{i}^*}$, $i \neq j$ and $w_{s+1/2}^{\mathsf{g}_\mathsf{i}^*}$, $i \neq k$ do not change by sparsity of $\Phi_{jk\ell}$. Analogously the updates to each $w_{s+1}^{\mathsf{f}_\mathsf{i}^*}$ and $w_{s+1}^{\mathsf{g}_\mathsf{i}^*}$ preserve our invariant. Finally, in every iteration $s > 0$, the updates to $w_{s+1/2}^{\mathsf{xx}}$ and w_{s+1}^{xx} only require evaluating $O(1)$ new gradients each, by 1-sparsity of the dual block updates. □

A.10.2 Proofs for Section 2.9.3

Lemma 28. *Define $\{\Phi_{jk\ell}, \Phi_{jk\ell'}\} : \mathcal{Z} \to \mathcal{Z}^*$ as in (2.58), (2.59), and define $r : \mathcal{Z} \to \mathbb{R}$ as in (2.53). Letting $w_+(jk\ell\ell')$ be w_{s+1} in Algorithm 9 if j, k, ℓ, ℓ' were sampled in iteration s, defining*

$$\begin{aligned} \Phi_{jk\ell}^{fg}(w) &:= \Phi_{jk\ell}^{\text{sep}}(w) + \Phi_{jk\ell}^{\text{bilin}}(w), \\ \Phi_{jk\ell'}^{fg}(w_{\mathsf{aux}}(jk\ell)) &:= \Phi_{jk\ell'}^{\text{sep}}(w_{\mathsf{aux}}(jk\ell)) + \Phi_{jk\ell'}^{\text{bilin}}(w_{\mathsf{aux}}(jk\ell)), \end{aligned}$$

we have

$$\begin{aligned} &\mathbb{E}\left[\left\langle \Phi_{jk\ell'}^{fg}(w_{\mathsf{aux}}(jk\ell)) - \Phi_{jk\ell}^{fg}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell') \right\rangle \right] \\ &\qquad \leq \lambda^{fg} \mathbb{E}\left[V_w^r \left(w_{\mathsf{aux}}(jk\ell) \right) + V_{w_{\mathsf{aux}}(jk\ell)}^r \left(w_+(jk\ell\ell') \right) \right], \end{aligned}$$

for

$$\lambda^{fg} = 2n(1+\gamma) + \frac{\sum_{i \in [n]} \sqrt{L_i^{\mathsf{x}}}}{\sqrt{n\mu^{\mathsf{x}}}} + \frac{\sum_{i \in [n]} \sqrt{L_i^{\mathsf{y}}}}{\sqrt{n\mu^{\mathsf{y}}}}.$$

Proof. This is immediate upon combining the following Lemmas 225 and 226. □

Lemma 225. *Following notation of Lemma 28, for* $\lambda^{\mathrm{sep}} := 2n(1+\gamma)$*, we have*

$$\begin{aligned}\mathbb{E}\left[\left\langle \Phi^{\mathrm{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^{\mathrm{sep}}_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle\right] \\ \le \lambda^{\mathrm{sep}}\mathbb{E}\left[V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right].\end{aligned}$$

Proof. The proof is similar to (part of) the proof of Lemma 21. We claim that for any j,k,ℓ,ℓ',

$$\left\langle \Phi^{\mathrm{sep}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^{\mathrm{sep}}_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle \le \lambda^{\mathrm{sep}}\left(V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right).$$

Fix j,k,ℓ,ℓ'. Since all p_j and q_k are lower bounded by $\frac{1}{2n}$ by assumption, applying Lemma 1 to the relevant blocks of r and nonnegativity of Bregman divergences proves the above display. □

Lemma 226. *Following notation of Lemma 28, for*

$$\lambda^{\mathrm{cross}} := \frac{2\sum_{i\in[n]}\sqrt{L^{\mathsf{x}}_i}}{\sqrt{n\mu^{\mathsf{x}}}} + \frac{2\sum_{i\in[n]}\sqrt{L^{\mathsf{y}}_i}}{\sqrt{n\mu^{\mathsf{y}}}},$$

we have

$$\begin{aligned}\mathbb{E}\left[\left\langle \Phi^{\mathrm{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^{\mathrm{bilin}}_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle\right] \\ \le \lambda^{\mathrm{cross}}\mathbb{E}\left[V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right].\end{aligned}$$

Proof. The proof is similar to (part of) the proof of Lemma 21. We claim that for any j,k,ℓ,ℓ',

$$\left\langle \Phi^{\mathrm{bilin}}_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^{\mathrm{bilin}}_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle \le \lambda^{\mathrm{cross}}\left(V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right).$$

Fix j,k,ℓ,ℓ'. By applying Item 1 in Lemma 13 with $f = f_j$, $\alpha = (L^{\mathsf{x}}_j\mu^{\mathsf{x}})^{-\frac{1}{2}}$,

$$\begin{aligned}\mathbb{E}_j\left[\frac{1}{np_j}\left\langle w^{\mathsf{f}^*_j}_{\mathsf{aux}} - w^{\mathsf{f}^*_j}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\rangle + \frac{1}{np_j}\left\langle w^{\mathsf{x}} - w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{f}^*_j}_{\mathsf{aux}} - w^{\mathsf{f}^*_j}_+(jk\ell\ell')\right\rangle\right] \\ \le \frac{2\sum_{i\in[n]}\sqrt{L^{\mathsf{x}}_i}}{\sqrt{n\mu^{\mathsf{x}}}}\left(V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right).\end{aligned}$$

Similarly, by applying Item 1 in Lemma 13 with $f = g_k$, $\alpha = (L^{\mathsf{y}}_k\mu^{\mathsf{y}})^{-\frac{1}{2}}$,

$$\begin{aligned}\mathbb{E}_j\left[\frac{1}{nq_k}\left\langle w^{\mathsf{g}^*_k}_{\mathsf{aux}} - w^{\mathsf{g}^*_k}, w^{\mathsf{y}}_{\mathsf{aux}}(\ell) - w^{\mathsf{y}}_+(jk\ell\ell')\right\rangle + \frac{1}{nq_k}\left\langle w^{\mathsf{y}} - w^{\mathsf{y}}_{\mathsf{aux}}(\ell), w^{\mathsf{g}^*_k}_{\mathsf{aux}} - w^{\mathsf{g}^*_k}_+(jk\ell\ell')\right\rangle\right] \\ \le \frac{2\sum_{i\in[n]}\sqrt{L^{\mathsf{y}}_i}}{\sqrt{n\mu^{\mathsf{y}}}}\left(V^r_w\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right).\end{aligned}$$

Summing the above displays yields the desired claim. □

Lemma 29. *Following notation of Lemma 28, and recalling the definition* (2.61)*, for*

$$\lambda_1 := 32(\lambda^h)^2,$$

where we define

$$\lambda^h := \frac{1}{n}\sum_{i\in[n]}\left(\frac{\Lambda_i^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda_i^{\mathsf{xx}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda_i^{\mathsf{yy}}}{\mu^{\mathsf{y}}}\right). \tag{2.61}$$

we have for any $\rho > 0$,

$$\begin{aligned}
&\mathbb{E}\left[\left\langle \Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^h_{jk\ell}(w), w_{\mathsf{aux}}(jk\ell) - w_+(jk\ell\ell')\right\rangle\right] \\
&\quad\le \left(2\lambda^h + \frac{1}{\rho}\right)\mathbb{E}\left[V_w^r\left(w_{\mathsf{aux}}(jk\ell)\right) + V^r_{w_{\mathsf{aux}}(jk\ell)}\left(w_+(jk\ell\ell')\right)\right] + \rho\lambda_1\mathbb{E}\left[V^r_{w_0}(w^\star) + V^r_{\bar w(\ell)}(w_\star)\right].
\end{aligned} \tag{2.62}$$

Proof. The proof is similar to (part of) the proof of Lemma 14. Fix j, k, ℓ, ℓ'. By definition,

$$\begin{aligned}
&\left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^h_{jk\ell}(w)\right]^{\mathsf{xx}} \\
&= \frac{1}{nr_{\ell'}}\left(\nabla_x h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_x h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}), \nabla_y h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}) - \nabla_y h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell))\right) \\
&\qquad -\frac{1}{nr_\ell}\left(\nabla_x h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}}) - \nabla_x h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}), \nabla_y h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}) - \nabla_y h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}})\right).
\end{aligned}$$

We decompose the x blocks of the left-hand side of (2.62) as

$$\begin{aligned}
\left\langle \left[\Phi^h_{jk\ell'}(w_{\mathsf{aux}}(jk\ell)) - \Phi^h_{jk\ell}(w)\right]^{\mathsf{x}}, w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\rangle &= \textcircled{1} + \textcircled{2} + \textcircled{3}, \\
\textcircled{1} &:= \frac{1}{nr_{\ell'}}\left\langle \nabla_x h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_x h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}), w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\rangle, \\
\textcircled{2} &:= \frac{1}{nr_\ell}\left\langle \nabla_x h_\ell(w_0^{\mathsf{x}}, w_0^{\mathsf{y}}) - \nabla_x h_\ell(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)), w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\rangle, \\
\textcircled{3} &:= \frac{1}{nr_\ell}\left\langle \nabla_x h_\ell(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_x h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}}), w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\rangle.
\end{aligned}$$

By the Lipschitzness bounds in (2.51) and Young's inequality,

$$\begin{aligned}
\textcircled{1} &\le \frac{1}{nr_{\ell'}}\left\|\nabla_x h_{\ell'}(w^{\mathsf{x}}_{\mathsf{aux}}(\ell), w^{\mathsf{y}}_{\mathsf{aux}}(\ell)) - \nabla_x h_{\ell'}(w_0^{\mathsf{x}}, w_0^{\mathsf{y}})\right\| \left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\| \\
&\le \frac{1}{nr_{\ell'}}\left(\Lambda^{\mathsf{xx}}_{\ell'}\left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w_0^{\mathsf{x}}\right\|\left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\| + \Lambda^{\mathsf{xx}}_{\ell'}\left\|w^{\mathsf{y}}_{\mathsf{aux}}(\ell) - w_0^{\mathsf{y}}\right\|\left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\|\right) \\
&\le \frac{2\rho(\Lambda^{\mathsf{xx}}_{\ell'})^2}{\mu^{\mathsf{x}}n^2r^2_{\ell'}}\left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w_0^{\mathsf{x}}\right\|^2 + \frac{2\rho(\Lambda^{\mathsf{xx}}_{\ell'})^2}{\mu^{\mathsf{x}}n^2r^2_{\ell'}}\left\|w^{\mathsf{y}}_{\mathsf{aux}}(\ell) - w_0^{\mathsf{y}}\right\|^2 + \frac{\mu^{\mathsf{x}}}{4\rho}\left\|w^{\mathsf{x}}_{\mathsf{aux}}(\ell) - w^{\mathsf{x}}_+(jk\ell\ell')\right\|^2.
\end{aligned}$$

Symmetrically, we bound

$$\textcircled{2} \le \frac{2\rho(\Lambda_\ell^{\mathsf{xx}})^2}{\mu^{\mathsf{x}} n^2 r_\ell^2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_0^{\mathsf{x}}\right\|^2 + \frac{2\rho(\Lambda_\ell^{\mathsf{xx}})^2}{\mu^{\mathsf{x}} n^2 r_\ell^2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_0^{\mathsf{y}}\right\|^2 + \frac{\mu^{\mathsf{x}}}{4\rho} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\|^2.$$

Finally, we have

$$\begin{aligned}
\textcircled{3} &\le \frac{1}{nr_\ell} \left\|\nabla_x h_\ell(w_{\mathsf{aux}}^{\mathsf{x}}(\ell), w_{\mathsf{aux}}^{\mathsf{y}}(\ell)) - \nabla_x h_\ell(w^{\mathsf{x}}, w^{\mathsf{y}})\right\| \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\| \\
&\le \frac{1}{nr_\ell} \left(\Lambda_\ell^{\mathsf{xx}} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w^{\mathsf{x}}\right\| \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\| + \Lambda_\ell^{\mathsf{xx}} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w^{\mathsf{y}}\right\| \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\|\right) \\
&\le \frac{1}{nr_\ell} \left(\frac{\Lambda_\ell^{\mathsf{xx}}}{\mu^{\mathsf{x}}} \left(\frac{\mu^{\mathsf{x}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w^{\mathsf{x}}\right\|^2 + \frac{\mu^{\mathsf{x}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\|^2\right)\right) \\
&+ \frac{1}{nr_\ell} \left(\frac{\Lambda_\ell^{\mathsf{xx}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} \left(\frac{\mu^{\mathsf{y}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w^{\mathsf{y}}\right\|^2 + \frac{\mu^{\mathsf{x}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_+^{\mathsf{x}}(jk\ell\ell')\right\|^2\right)\right).
\end{aligned}$$

We may similarly decompose the y blocks of the left-hand side of (2.62) as $\textcircled{4} + \textcircled{5} + \textcircled{6}$, where symmetrically, we have

$$\begin{aligned}
\textcircled{4} &\le \frac{2\rho(\Lambda_{\ell'}^{\mathsf{yy}})^2}{\mu^{\mathsf{y}} n^2 r_{\ell'}^2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_0^{\mathsf{y}}\right\|^2 + \frac{2\rho(\Lambda_{\ell'}^{\mathsf{xx}})^2}{\mu^{\mathsf{y}} n^2 r_{\ell'}^2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_0^{\mathsf{x}}\right\|^2 + \frac{\mu^{\mathsf{y}}}{4\rho} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_+^{\mathsf{y}}(jk\ell\ell')\right\|^2, \\
\textcircled{5} &\le \frac{2\rho(\Lambda_\ell^{\mathsf{yy}})^2}{\mu^{\mathsf{y}} n^2 r_\ell^2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_0^{\mathsf{y}}\right\|^2 + \frac{2\rho(\Lambda_\ell^{\mathsf{xx}})^2}{\mu^{\mathsf{y}} n^2 r_\ell^2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_0^{\mathsf{x}}\right\|^2 + \frac{\mu^{\mathsf{y}}}{4\rho} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_+^{\mathsf{y}}(jk\ell\ell')\right\|^2, \\
\textcircled{6} &\le \frac{1}{nr_\ell} \left(\frac{\Lambda_\ell^{\mathsf{yy}}}{\mu^{\mathsf{y}}} \left(\frac{\mu^{\mathsf{y}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w^{\mathsf{y}}\right\|^2 + \frac{\mu^{\mathsf{y}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_+^{\mathsf{y}}(jk\ell\ell')\right\|^2\right)\right) \\
&+ \frac{1}{nr_\ell} \left(\frac{\Lambda_\ell^{\mathsf{xx}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} \left(\frac{\mu^{\mathsf{x}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w^{\mathsf{x}}\right\|^2 + \frac{\mu^{\mathsf{y}}}{2} \left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_+^{\mathsf{y}}(jk\ell\ell')\right\|^2\right)\right).
\end{aligned}$$

We first observe that by definition of r and nonnegativity of Bregman divergences,

$$\begin{aligned}
\textcircled{3} + \textcircled{6} &\le \frac{1}{nr_\ell} \left(\frac{\Lambda_\ell^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{\Lambda_\ell^{\mathsf{xx}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}} + \frac{\Lambda_\ell^{\mathsf{yy}}}{\mu^{\mathsf{y}}}\right) \left(V_w^r(w_{\mathsf{aux}}(jk\ell)) + V_{w_{\mathsf{aux}}(jk\ell)}^r(w_+(jk\ell\ell'))\right) \\
&\le 2\lambda^h \left(V_w^r(w_{\mathsf{aux}}(jk\ell)) + V_{w_{\mathsf{aux}}(jk\ell)}^r(w_+(jk\ell\ell'))\right).
\end{aligned}$$

Moreover, since by the triangle inequality and $(a+b)^2 \le 2a^2 + 2b^2$,

$$\begin{aligned}
\left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_0^{\mathsf{x}}\right\|^2 &\le 2\left\|w_{\mathsf{aux}}^{\mathsf{x}}(\ell) - w_\star^{\mathsf{x}}\right\|^2 + 2\left\|w_0^{\mathsf{x}} - w_\star^{\mathsf{x}}\right\|^2, \\
\left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_0^{\mathsf{y}}\right\|^2 &\le 2\left\|w_{\mathsf{aux}}^{\mathsf{y}}(\ell) - w_\star^{\mathsf{y}}\right\|^2 + 2\left\|w_0^{\mathsf{y}} - w_\star^{\mathsf{y}}\right\|^2,
\end{aligned}$$

we have by definition of r and λ_1,

$$\textcircled{1}+\textcircled{2}+\textcircled{4}+\textcircled{5} \leq \frac{1}{\rho}\left(V_w^r(w_{\mathsf{aux}}(jk\ell)) + V_{w_{\mathsf{aux}}(jk\ell)}^r(w_+(jk\ell\ell'))\right) + \rho\lambda_1\left(V_{w_0}^r(w_\star) + V_{\bar{w}(\ell)}^r(w_\star)\right).$$

Summing the above displays and taking expectations yields the claim. □

A.10.3 Proofs for Section 2.9.4

Proposition 7. *Consider a single iteration $0 \leq t < T$ of Algorithm 8, and let $z_\star$ is the saddle point to $F_{\text{mmfs-pd}}$ (defined in (2.52)). Setting S as in (2.63) and*

$$N := O\left(\log\left(\gamma\lambda\right)\right), \tag{2.64}$$

for an appropriately large constant in our implementation of Algorithm 9 and λ as in (2.63), we have

$$\mathbb{E}V_{z_{t+1}}^r(z_\star) \leq \frac{4\gamma}{1+4\gamma}V_{z_t}^r(z_\star).$$

Proof. Fix an iteration $t \in [T]$ of Algorithm 8, and let $z_{t+1}^\star$ be the exact solution to the VI in $\Phi^{\text{mmfs-pd}} + \gamma\nabla r - \nabla r(z_t)$. By the guarantee of Proposition 6, after the stated number of NS iterations in Algorithm 9 (for an appropriately large constant), we obtain a point z_{t+1} such that

$$\mathbb{E}\left[V_{z_{t+1}}^r(z_{t+1}^\star)\right] \leq \frac{1}{1+3\gamma\widetilde{\kappa}}V_{z_t}^r(\hat{z}_{t+1}), \text{ where } \widetilde{\kappa} := 10\sum_{i\in[n]}\left(\frac{L_i^{\mathsf{x}}+\Lambda_i^{\mathsf{xx}}}{\mu^{\mathsf{x}}} + \frac{L_i^{\mathsf{y}}+\Lambda_i^{\mathsf{yy}}}{\mu^{\mathsf{y}}} + \frac{\Lambda_i^{\mathsf{xx}}}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}\right)^2. \tag{A.10}$$

The optimality condition on $z_{t+1}^\star$ yields

$$\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}^\star\right), z_{t+1}^\star - z_\star\right\rangle \leq \gamma V_{z_t}^r\left(z_\star\right) - \gamma V_{z_{t+1}^\star}^r\left(z_\star\right) - \gamma V_{z_t}\left(z_{t+1}^\star\right).$$

Rearranging terms then gives:

$$\begin{aligned}
\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right), z_{t+1}-z_\star\right\rangle &\leq \gamma V_{z_t}^r\left(z_\star\right)-\gamma V_{z_{t+1}}^r\left(z_\star\right)-\gamma V_{z_t}^r\left(z_{t+1}^\star\right)\\
&+\gamma\left(V_{z_{t+1}}^r\left(z_\star\right)-V_{z_{t+1}^\star}^r\left(z_\star\right)\right)\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right)-\Phi^{\text{mmfs-pd}}\left(z_{t+1}^\star\right), z_{t+1}^\star-z_\star\right\rangle\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right), z_{t+1}-z_{t+1}^\star\right\rangle\\
&=\gamma V_{z_t}^r\left(z_\star\right)-\gamma V_{z_{t+1}}^r\left(z_\star\right)-\gamma V_{z_t}^r\left(z_{t+1}^\star\right)\\
&+\gamma V_{z_{t+1}}^r\left(z_{t+1}^\star\right)+\gamma\left\langle \nabla r\left(z_{t+1}\right)-\nabla r\left(z_{t+1}^\star\right), z_{t+1}^\star-z_\star\right\rangle\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right)-\Phi^{\text{mmfs-pd}}\left(z_{t+1}^\star\right), z_{t+1}^\star-z_\star\right\rangle\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right), z_{t+1}-z_{t+1}^\star\right\rangle\\
&\leq \gamma V_{z_t}^r\left(z_\star\right)-\gamma V_{z_{t+1}}^r\left(z_\star\right)-\gamma V_{z_t}^r\left(z_{t+1}^\star\right)+\gamma V_{z_{t+1}}^r\left(z_{t+1}^\star\right)\\
&+\gamma\left\langle \nabla r\left(z_{t+1}\right)-\nabla r\left(z_{t+1}^\star\right), z_{t+1}-z_\star\right\rangle\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right)-\Phi^{\text{mmfs-pd}}\left(z_{t+1}^\star\right), z_{t+1}-z_\star\right\rangle\\
&+\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right)-\Phi^{\text{mmfs-pd}}\left(z_\star\right), z_{t+1}-z_{t+1}^\star\right\rangle.
\end{aligned} \tag{A.11}$$

In the only equality, we used the identity (2.10). The last inequality used monotonicity of the operators $\gamma\nabla r$ and $\Phi^{\text{mmfs-pd}}$, as well as $\Phi^{\text{mmfs-pd}}(z_\star)=0$ because it is an unconstrained minimax optimization problem. In the remainder of the proof, we will bound the last three lines of (A.11).

First, for any $\alpha>0$, we bound:

$$\begin{aligned}
\left\langle \nabla r\left(z_{t+1}\right)-\nabla r\left(z_{t+1}^\star\right), z_{t+1}-z_\star\right\rangle &= \mu^{\mathsf{x}}\left\langle z_{t+1}^{\mathsf{x}}-(z_{t+1}^\star)^{\mathsf{x}}, z_{t+1}^{\mathsf{x}}-z_\star^{\mathsf{x}}\right\rangle+\mu^{\mathsf{y}}\left\langle z_{t+1}^{\mathsf{y}}-(z_{t+1}^\star)^{\mathsf{y}}, z_{t+1}^{\mathsf{y}}-z_\star^{\mathsf{y}}\right\rangle\\
&+\frac{1}{n}\sum_{i\in[n]}\left\langle \nabla f_i^*(z_{t+1}^{\mathsf{f}_\mathsf{i}^*})-\nabla f_i^*((z_{t+1}^\star)^{\mathsf{f}_\mathsf{i}^*}), z_{t+1}^{\mathsf{f}_\mathsf{i}^*}-z_\star^{\mathsf{f}_\mathsf{i}^*}\right\rangle\\
&+\frac{1}{n}\sum_{i\in[n]}\left\langle \nabla g_i^*(z_{t+1}^{\mathsf{g}_\mathsf{i}^*})-\nabla g_i^*((z_{t+1}^\star)^{\mathsf{g}_\mathsf{i}^*}), z_{t+1}^{\mathsf{g}_\mathsf{i}^*}-z_\star^{\mathsf{g}_\mathsf{i}^*}\right\rangle\\
&\leq 2\alpha\mu^{\mathsf{x}}\left\|z_{t+1}^{\mathsf{x}}-(z_{t+1}^\star)^{\mathsf{x}}\right\|^2+\frac{\mu^{\mathsf{x}}}{8\alpha}\left\|z_{t+1}^{\mathsf{x}}-z_\star^{\mathsf{x}}\right\|^2\\
&+2\alpha\mu^{\mathsf{y}}\left\|z_{t+1}^{\mathsf{y}}-(z_{t+1}^\star)^{\mathsf{y}}\right\|^2+\frac{\mu^{\mathsf{y}}}{8\alpha}\left\|z_{t+1}^{\mathsf{y}}-z_\star^{\mathsf{y}}\right\|^2\\
&+\frac{1}{n}\sum_{i\in[n]}\left(\frac{2\alpha L_i^{\mathsf{x}}}{(\mu^{\mathsf{x}})^2}\left\|z_{t+1}^{\mathsf{f}_\mathsf{i}^*}-(z_{t+1}^\star)^{\mathsf{f}_\mathsf{i}^*}\right\|^2+\frac{1}{8\alpha L_i^{\mathsf{x}}}\left\|z_{t+1}^{\mathsf{f}_\mathsf{i}^*}-z_\star^{\mathsf{f}_\mathsf{i}^*}\right\|^2\right)\\
&+\frac{1}{n}\sum_{i\in[n]}\left(\frac{2\alpha L_i^{\mathsf{y}}}{(\mu^{\mathsf{y}})^2}\left\|z_{t+1}^{\mathsf{g}_\mathsf{i}^*}-(z_{t+1}^\star)^{\mathsf{g}_\mathsf{i}^*}\right\|^2+\frac{1}{8\alpha L_i^{\mathsf{y}}}\left\|z_{t+1}^{\mathsf{g}_\mathsf{i}^*}-z_\star^{\mathsf{g}_\mathsf{i}^*}\right\|^2\right)\\
&\leq \frac{1}{4\alpha}V_{z_{t+1}}^r(z_\star)+\widetilde{\kappa}\alpha V_{z_{t+1}}^r(z_{t+1}^\star).
\end{aligned} \tag{A.12}$$

The equality used the definition of r in (2.53). The first inequality used Young's and Cauchy-Schwarz

on the $\mathcal{X} \times \mathcal{Y}$ blocks, as well as $\frac{1}{\mu_i^{\mathsf{x}}}$-smoothness of the f_i^* from Assumption 3 and Item 4 in Fact 1 (and similar bounds on each g_i^*). The last inequality used strong convexity of each piece of r.

Similarly, by definition of $\Phi^{\text{mmfs-pd}}$ (2.54) which we denote for Φ for brevity in the following:

$$\begin{aligned}
\left\langle \Phi\left(z_{t+1}\right) - \Phi\left(z_{t+1}^\star\right), z_{t+1} - z_\star \right\rangle \leq{}& \frac{1}{8} V_{z_{t+1}}^r(z_\star) + 2\widetilde{\kappa} V_{z_{t+1}}^r(z_{t+1}^\star) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left\langle \nabla_x h_i(z_{t+1}^{\mathsf{x}}, z_{t+1}^{\mathsf{y}}) - \nabla_x h_i((z_{t+1}^\star)^{\mathsf{x}}, (z_{t+1}^\star)^{\mathsf{y}}), z_{t+1}^{\mathsf{x}} - z_\star^{\mathsf{x}} \right\rangle \\
&+ \frac{1}{n} \sum_{i\in[n]} \left\langle \nabla_y h_i(z_{t+1}^{\mathsf{x}}, z_{t+1}^{\mathsf{y}}) - \nabla_y h_i((z_{t+1}^\star)^{\mathsf{x}}, (z_{t+1}^\star)^{\mathsf{y}}), z_{t+1}^{\mathsf{y}} - z_\star^{\mathsf{y}} \right\rangle \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(\left\langle z_{t+1}^{\mathsf{f}_i^*} - (z_{t+1}^\star)^{\mathsf{f}_i^*}, z_{t+1}^{\mathsf{x}} - z_\star^{\mathsf{x}} \right\rangle + \left\langle z_{t+1}^{\mathsf{g}_i^*} - (z_{t+1}^\star)^{\mathsf{g}_i^*}, z_{t+1}^{\mathsf{y}} - z_\star^{\mathsf{y}} \right\rangle \right) \\
&- \frac{1}{n} \sum_{i\in[n]} \left(\left\langle z_{t+1}^{\mathsf{x}} - (z_{t+1}^\star)^{\mathsf{x}}, z_{t+1}^{\mathsf{f}_i^*} - z_\star^{\mathsf{f}_i^*} \right\rangle + \left\langle z_{t+1}^{\mathsf{y}} - (z_{t+1}^\star)^{\mathsf{y}}, z_{t+1}^{\mathsf{g}_i^*} - z_\star^{\mathsf{g}_i^*} \right\rangle \right)
\end{aligned}$$

where we used (A.12) to bound the ∇r terms. Consequently,

$$\begin{aligned}
\left\langle \Phi\left(z_{t+1}\right) - \Phi\left(z_{t+1}^\star\right), z_{t+1} - z_\star \right\rangle \leq{}& \frac{1}{8} V_{z_{t+1}}^r(z_\star) + 2\widetilde{\kappa} V_{z_{t+1}}^r(z_{t+1}^\star) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(\frac{\mu^{\mathsf{x}}}{16} V_{z_{t+1}^{\mathsf{x}}}(z_\star^{\mathsf{x}}) + \frac{\mu^{\mathsf{y}}}{16} V_{z_{t+1}^{\mathsf{y}}}(z_\star^{\mathsf{y}}) \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(16 \left(\frac{(\Lambda_i^{\mathsf{xx}})^2}{\mu^{\mathsf{x}}} + \frac{(\Lambda_i^{\mathsf{xx}})^2}{\mu^{\mathsf{y}}} \right) V_{z_{t+1}^{\mathsf{x}}}((z_{t+1}^\star)^{\mathsf{x}}) \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(16 \left(\frac{(\Lambda_i^{\mathsf{xx}})^2}{\mu^{\mathsf{x}}} + \frac{(\Lambda_i^{\mathsf{yy}})^2}{\mu^{\mathsf{y}}} \right) V_{z_{t+1}^{\mathsf{y}}}((z_{t+1}^\star)^{\mathsf{y}}) \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(\frac{\mu^{\mathsf{x}}}{16} V_{z_{t+1}^{\mathsf{x}}}(z_\star^{\mathsf{x}}) + \frac{\mu^{\mathsf{y}}}{16} V_{z_{t+1}^{\mathsf{y}}}(z_\star^{\mathsf{y}}) \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(\frac{8}{\mu^{\mathsf{x}}} \left\| z_{t+1}^{\mathsf{f}_i^*} - (z_{t+1}^\star)^{\mathsf{f}_i^*} \right\|^2 + \frac{8}{\mu^{\mathsf{y}}} \left\| z_{t+1}^{\mathsf{g}_i^*} - (z_{t+1}^\star)^{\mathsf{g}_i^*} \right\|^2 \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(\frac{1}{8} V_{z_{t+1}^{\mathsf{f}_i^*}}^{f_i^*}\left(z_\star^{\mathsf{f}_i^*}\right) + \frac{1}{8} V_{z_{t+1}^{\mathsf{g}_i^*}}^{g_i^*}\left(z_\star^{\mathsf{g}_i^*}\right) \right) \\
&+ \frac{1}{n} \sum_{i\in[n]} \left(8L_i^{\mathsf{x}} V_{z_{t+1}^{\mathsf{x}}}((z_{t+1}^\star)^{\mathsf{x}}) + 8L_i^{\mathsf{y}} V_{z_{t+1}^{\mathsf{y}}}((z_{t+1}^\star)^{\mathsf{y}}) \right) \\
&\leq \frac{1}{4} V_{z_{t+1}}^r(z_\star) + \widetilde{\kappa} V_{z_{t+1}}^r(z_{t+1}^\star).
\end{aligned} \tag{A.13}$$

In the first inequality, we used Cauchy-Schwarz, Young's, and our various smoothness assumptions (as well as strong convexity of each f_i^* and g_i^*). The last inequality used strong convexity of each piece of r.

For the last term, by a similar argument as in the previous bounds, we have

$$\left\langle \Phi(z_{t+1}) - \Phi(z_\star), z_{t+1} - z^\star_{t+1} \right\rangle \leq \frac{1}{4} V^r_{z_{t+1}}(z_\star) + \widetilde{\kappa} V^r_{z_{t+1}}(z^\star_{t+1}). \tag{A.14}$$

Plugging the inequalities (A.12) with $\alpha = \gamma$, (A.13) and (A.14) back into (A.11), this implies

$$\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right), z_{t+1} - z_\star \right\rangle \leq \gamma V^r_{z_t}\left(z_\star\right) - \gamma V^r_{z_{t+1}}\left(z_\star\right) - \gamma V^r_{z_t}\left(z^\star_{t+1}\right) + \gamma V^r_{z_{t+1}}\left(z^\star_{t+1}\right) \tag{A.15}$$

$$+ \frac{3}{4} V^r_{z_{t+1}}(z_\star) + 3\widetilde{\kappa}\gamma^2 V^r_{z_{t+1}}(z^\star_{t+1}). \tag{A.16}$$

By strong monotonicity of $\Phi^{\text{mmfs-pd}}$ with respect to r, we also have

$$\left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right), z_{t+1} - z_\star \right\rangle \geq \left\langle \Phi^{\text{mmfs-pd}}\left(z_{t+1}\right) - \Phi^{\text{mmfs-pd}}\left(z_\star\right), z_{t+1} - z_\star \right\rangle \geq V^r_{z_{t+1}}\left(z_\star\right). \tag{A.17}$$

Combining (A.16) and (A.17) with the assumption (A.10), and taking expectations, we obtain

$$\left(\frac{1}{4} + \gamma\right) \mathbb{E} V^r_{z_{t+1}}(z_\star) \leq \gamma V^r_{z_t}(z_\star) \implies \mathbb{E} V^r_{z_{t+1}}(z_\star) \leq \frac{4\gamma}{1+4\gamma} V^r_{z_t}(z_\star).$$

□

Appendix B

Deferred proofs from Chapter 3

B.1 Deferred proofs from Section 3.2

Proof of Proposition 8. It is clear that our choices of $\mathcal{X}, \mathcal{Y}$ are compact and convex, and that the local norms we defined are indeed norms (in all cases, they are quadratic norms). Validity of our choices of Θ follow from the well-known facts that for $x \in \Delta^n$, the entropy function $\sum_{j\in[n]} x_j \log x_j$ is convex with range $\log n$, and that for $x \in \mathbb{B}^n$, $\frac{1}{2}\|x\|_2^2$ is convex with range $\frac{1}{2}$.

In the Euclidean case we have that $V_x(x') = \frac{1}{2}\|x - x'\|_2^2$ and (3.18) follows from the Cauchy-Schwarz and Young inequalities:

$$\langle \gamma, x' - x\rangle \le \frac{1}{2}\|\gamma\|_2^2 + \frac{1}{2}\|x - x'\|_2^2 .$$

Similarly, for the simplex we have that entropy is 1-strongly-convex with respect to $\|\cdot\|_1$ and therefore $V_y(y') \ge \frac{1}{2}\|y - y'\|_1^2$, and we obtain (3.18) from the Hölder and Young inequalities,

$$\langle \gamma, y' - y\rangle \le \frac{1}{2}\|\gamma\|_\infty^2 + \frac{1}{2}\|y - y'\|_1^2 ,$$

where we note that $\|\gamma\|_\infty = \|\gamma\|_*$ in this case.

Finally, $\mathrm{clip}(\cdot)$ is not the identity only when the corresponding domain is the simplex, in which case entropy satisfies the local norms bound (see the following Lemma 227),

$$\langle \gamma, y - y'\rangle - V_y(y') \le \sum_{i\in[m]} \gamma_i^2 y_i \quad \text{for all } y, y' \in \Delta^m \text{ and } \gamma \in \mathbb{R}^m \text{ such that } \|\gamma\|_\infty \le 1.$$

Noting that $\|\mathrm{clip}(\gamma)\|_\infty \le 1$ and that $\|\mathrm{clip}(\gamma)\|_y \le \|\gamma\|_y$ for all $y \in \Delta^m$, we have the desired local

bound (3.19). Finally, for every coordinate $i \in [m]$ we have

$$|\gamma_i - [\mathrm{clip}(\gamma)]_i| = ||\gamma_i| - 1| \, \mathbb{I}_{\{|\gamma_i|>1\}} \leq |\gamma_i| \, \mathbb{I}_{\{|\gamma_i|>1\}} \leq |\gamma_i|^2.$$

Consequently, $| \langle \gamma - \mathrm{clip}(\gamma), z \rangle | \leq \sum_{i\in[m]} \gamma_i^2 z_i$, giving the distortion bound (3.20). □

We now state and give a proof of Lemma 227, our local norm bound. For the remainder of this subsection, let $\mathcal{Y}$ be the m dimensional simplex Δ^m, and let $r(y) = \sum_{i=1}^m y_i \log y_i$ be the negative entropy distance generating function. The corresponding Bregman divergence is the KL divergence, which is well-defined for any $y, y' \in \mathbb{R}^m_{\geq 0}$ and has the form

$$V_y(y') = \sum_{i\in[m]} \left[y'_i \log \frac{y'_i}{y_i} + y_i - y'_i \right] = \int_0^1 dt \int_0^t \sum_{i\in[m]} \frac{(y_i - y'_i)^2}{(1-\tau)y_i + \tau y'_i} d\tau. \tag{B.1}$$

In the literature, "local norms" regret analysis [474] relies on the fact that $r^*(\gamma) = \log(\sum_i e^{\gamma_i})$ (the conjugate of negative entropy in the simplex) is locally smooth with respect to a Euclidean norm weighted by $\nabla r^*(\gamma) = \frac{e^\gamma}{\|e^\gamma\|_1}$. More precisely, the Bregman divergence $V^*_\gamma(\gamma') = r^*(\gamma') - r^*(\gamma) - \langle \nabla r^*(\gamma), \gamma' - \gamma \rangle$ satisfies

$$V^*_\gamma(\gamma + \delta) \leq \|\delta\|^2_{\nabla r^*(\gamma)} := \sum_i [\nabla r^*(\gamma)]_i \cdot \delta_i^2 \quad \text{whenever } \delta_i \leq 1.79 \;\; \forall i. \tag{B.2}$$

Below, we state this bound in a form that is directly applicable to our analysis.

Lemma 227. *Let $y, y' \in \Delta^m$ and $\delta \in \mathbb{R}^m$. If δ satisfies $\delta_i \leq 1.79$ for all $i \in [m]$ then the KL divergence $V_y(y')$ satisfies*

$$\langle \delta, y' - y \rangle - V_y(y') \leq \|\delta\|_y^2 := \sum_{i\in[m]} y_i \delta_i^2$$

Proof. It suffices to consider y in the relative interior of the simplex where r is differentiable; the final result will hold for any y in the simplex by continuity. Recall the following general facts about convex conjugates: $\langle \gamma', y' \rangle - r(y') \leq r^*(\gamma')$ for any $\gamma' \in \mathbb{R}^m$, $y = \nabla r^*(\nabla r(y))$ and $r^*(\nabla r(y)) = \langle \nabla r(y), y \rangle - r(y)$. Therefore, we have for all $y' \in \Delta^m$,

$$\begin{aligned}\langle \delta, y' - y \rangle - V_y(y') &= \langle \nabla r(y) + \delta, y' \rangle - r(y') - [\langle \nabla r(y), y \rangle - r(y)] - \langle y, \delta \rangle \\ &\leq r^*(\nabla r(y) + \delta) - r^*(\nabla r(y)) - \langle \nabla r^*(\nabla r(y)), \delta \rangle = V^*_{\nabla r(y)}\big(\nabla r(y) + \delta\big).\end{aligned}$$

The result follows from (B.2) with $\gamma = \nabla r(y)$, recalling again that $y = \nabla r^*(\nabla r(y))$. For completeness

we prove (B.2) below, following [474]. We have

$$r^*(\gamma+\delta)-r^*(\gamma)=\log\left(\frac{\sum_{i\in[m]}e^{\gamma_i+\delta_i}}{\sum_{i\in[m]}e^{\gamma_i}}\right)\overset{(i)}{\leq}\log\left(1+\frac{\sum_{i\in[m]}e^{\gamma_i}(\delta_i+\delta_i^2)}{\sum_{i\in[m]}e^{\gamma_i}}\right)$$
$$=\log(1+\left\langle\nabla r^*(\gamma),\delta+\delta^2\right\rangle)\overset{(ii)}{\leq}\left\langle\nabla r^*(\gamma),\delta\right\rangle+\left\langle\nabla r^*(\gamma),\delta^2\right\rangle,$$

where (i) follows from $e^x\leq 1+x+x^2$ for all $x\leq 1.79$ and (ii) follows from $\log(1+x)\leq x$ for all x. Therefore,

$$V_\gamma^*(\gamma+\delta)=r^*(\gamma+\delta)-r^*(\gamma)-\left\langle\nabla r^*(\gamma),\delta\right\rangle\leq\left\langle\nabla r^*(\gamma),\delta^2\right\rangle=\|\delta\|^2_{\nabla r^*(\gamma)},$$

completing the proof. □

B.2 Deferred proofs from Section 3.3

B.2.1 Proof of Proposition 9

In this section, we provide a convergence result for mirror descent under local norms. We require the following well-known regret bound for mirror descent.

Lemma 228. *Let $Q:\mathcal{Z}\to\mathbb{R}$ be convex, let $T\in\mathbb{N}$, $z_0\in\mathcal{Z}$ and $\gamma_0,\gamma_1,\ldots,\gamma_T\in\mathcal{Z}^*$. The sequence $z_1,\ldots,z_T$ defined by*

$$z_t=\operatorname{argmin}_{z\in\mathcal{Z}}\left\{\langle\gamma_{t-1},z\rangle+Q(z)+V_{z_{t-1}}(z)\right\}$$

satisfies for all $u\in\mathcal{Z}$ (denoting $z_{T+1}:=u$),

$$\sum_{t=0}^{T}\langle\gamma_t,z_t-u\rangle+\sum_{t=1}^{T}\langle\nabla Q(z_t),z_t-u\rangle\leq V_{z_0}(u)+\sum_{t=0}^{T}\{\langle\gamma_t,z_t-z_{t+1}\rangle-V_{z_t}(z_{t+1})\}.\tag{B.3}$$

Proof. Fix $u\equiv z_{T+1}\in\mathcal{Z}$. We note that by definition z_t is the solution of a convex optimization problem with (sub)gradient $\gamma_{t-1}+\nabla Q(\cdot)+\nabla V_{z_{t-1}}(\cdot)$, and therefore by the first-order optimality condition [271] satisfies

$$\left\langle\gamma_{t-1}+\nabla Q(z_t)+\nabla V_{z_{t-1}}(z_t),z_t-z_{T+1}\right\rangle\leq 0.$$

By the three-point equality of Bregman divergences we have $-\left\langle\nabla V_{z_{t-1}}(z_t),z_t-z_{T+1}\right\rangle=V_{z_{t-1}}(z_{T+1})-V_{z_t}(z_{T+1})-V_{z_{t-1}}(z_t)$. Substituting and summing over $t\in[T]$ gives

$$\sum_{t=1}^{T}\langle\gamma_{t-1}+\nabla Q(z_t),z_t-z_{T+1}\rangle\leq V_{z_0}(z_{T+1})-\sum_{t=0}^{T}V_{z_t}(z_{t+1}).$$

Rearranging the LHS and adding $\langle \gamma_T, z_T - z_{T+1} \rangle$ to both sides of the inequality gives

$$\sum_{t=1}^{T} \langle \gamma_t + \nabla Q(z_t), z_t - z_{T+1} \rangle \le V_{z_0}(z_{T+1}) + \sum_{t=0}^{T} \left\{ \langle \gamma_t, z_t - z_{t+1} \rangle - V_{z_t}(z_{t+1}) \right\},$$

which is the bound stated in the lemma. □

The proposition follows from this regret bound, the properties of the local norm setup, and the "ghost iterate" argument due to [416].

Proposition 9. *Let $(\mathcal{Z}, \|\cdot\|_\cdot, r, \Theta, \mathrm{clip})$ be a local norm setup, let $L, \epsilon > 0$, and let $\tilde{g}$ be an L-local estimator. Then, for $\eta \le \frac{\epsilon}{9L^2}$ and $T \ge \frac{6\Theta}{\eta\epsilon} \ge \frac{54L^2\Theta}{\epsilon^2}$, Algorithm 10 outputs a point $\bar{z}$ such that*

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \le \mathbb{E}\left[\sup_{u \in \mathcal{Z}} \frac{1}{T+1} \sum_{t=0}^{T} \langle g(z_t), z_t - u \rangle \right] \le \epsilon.$$

Proof. Defining

$$\tilde{\Delta}_t := g(z_t) - \frac{1}{\eta}\mathrm{clip}(\eta \tilde{g}(z_t))$$

and the ghost iterates

$$s_t = \mathrm{argmin}_{s \in \mathcal{Z}} \left\{ \left\langle \frac{1}{2}\eta\tilde{\Delta}_{t-1}, s \right\rangle + V_{s_{t-1}}(s) \right\} \quad \text{with} \quad s_0 = w_0,$$

we rearrange the regret as

$$\eta \sum_{t=0}^{T} \langle g(z_t), z_t - u \rangle \le \sum_{t=0}^{T} \langle \mathrm{clip}(\eta\tilde{g}(z_t)), z_t - u \rangle + \sum_{t=0}^{T} \left\langle \eta\tilde{\Delta}_t, s_t - u \right\rangle + \sum_{t=0}^{T} \left\langle \eta\tilde{\Delta}_t, z_t - s_t \right\rangle, \quad \text{(B.4)}$$

and bound each term in turn.

We first apply Lemma 228 with $Q = 0$ and $\gamma_t = \mathrm{clip}(\eta\tilde{g}(z_t))$, using (3.19) to conclude that

$$\sum_{t=0}^{T} \langle \mathrm{clip}(\eta\tilde{g}(z_t)), z_t - u \rangle \le V_{z_0}(u) + \eta^2 \sum_{t=0}^{T} \|\tilde{g}(z_t)\|_{z_t}^2, \text{ for all } u \in \mathcal{Z}. \quad \text{(B.5)}$$

Next, we apply Lemma 228 again, this time with $\gamma_t = \frac{1}{2}\eta\tilde{\Delta}_t$, to obtain the regret bound

$$\begin{aligned} \sum_{t=0}^{T} \left\langle \eta\tilde{\Delta}_t, s_t - u \right\rangle &\le 2V_{z_0}(u) + \sum_{t=0}^{T} \left\{ \left\langle \eta\tilde{\Delta}_t, s_t - s_{t+1} \right\rangle - 2V_{s_t}(s_{t+1}) \right\} \\ &\le 2V_{z_0}(u) + \eta^2 \sum_{t=0}^{T} \left\{ \|\tilde{g}(z_t)\|_{s_t}^2 + \frac{1}{2}\|g(z_t)\|_*^2 \right\}, \end{aligned} \quad \text{(B.6)}$$

for all $u \in \mathcal{Z}$, where we used

$$\left\langle \eta\tilde{\Delta}_t, s_t - s_{t+1} \right\rangle - 2V_{s_t}(s_{t+1}) \le \langle \eta g(z_t), s_t - s_{t+1}\rangle - V_{s_t}(s_{t+1}) + |\langle \mathrm{clip}(\eta\tilde{g}(z_t)), s_t - s_{t+1}\rangle| - V_{s_t}(s_{t+1}),$$

and then appealed to the bounds (3.18) and (3.19) in the definition of the local norm setup. Now, substituting (B.5) and (B.6) into (B.4), maximizing over u, and taking an expectation, we obtain

$$\begin{aligned}\mathbb{E}\sup_{u\in\mathcal{Z}}\sum_{t=0}^{T}\langle \eta g(z_t), z_t - u\rangle \le 3\Theta + \eta^2\mathbb{E}\sum_{t=0}^{T}\left\{\|\tilde{g}(z_t)\|_{z_t}^2 + \|\tilde{g}(z_t)\|_{s_t}^2 + \tfrac{1}{2}\|g(z_t)\|_*^2\right\} \\ + \mathbb{E}\sum_{t=0}^{T}\left\langle \eta\tilde{\Delta}_t, z_t - s_t\right\rangle.\end{aligned} \tag{B.7}$$

To bound the last term we use the fact that $g(z_t) = \mathbb{E}\left[\tilde{g}(z_t) \mid z_t, s_t\right]$ (which follows from the first part of Definition 4). We then write

$$\left|\mathbb{E}\left\langle \eta\tilde{\Delta}_t, z_t - s_t\right\rangle\right| \le \mathbb{E}\left|\langle \eta\tilde{g}(z_t) - \mathrm{clip}(\eta\tilde{g}(z_t)), z_t - s_t\rangle\right| \le \eta^2\|\tilde{g}(z_t)\|_{z_t}^2 + \eta^2\|\tilde{g}(z_t)\|_{s_t}^2, \tag{B.8}$$

where the first inequality is by Jensen's inequality, and the last is due to the property (3.20) of the local norm setup. Substituting (B.8) into (B.7), we obtain

$$\mathbb{E}\sup_{u\in\mathcal{Z}}\sum_{t=0}^{T}\langle \eta g(z_t), z_t - u\rangle \le 3\Theta + \eta^2\mathbb{E}\sum_{t=0}^{T}\left\{2\|\tilde{g}(z_t)\|_{z_t}^2 + 2\|\tilde{g}(z_t)\|_{s_t}^2 + \frac{1}{2}\|g(z_t)\|_*^2\right\}.$$

Finally, using the second moment bound of local gradient estimator (Definition 4) and its consequence Lemma 30, we may bound each of the expected squared norm terms by L^2. Dividing through by $\eta(T+1)$ gives

$$\mathbb{E}\sup_{u\in\mathcal{Z}}\left[\frac{1}{T+1}\sum_{t=0}^{T}\langle g(z_t), z_t - u\rangle\right] \le \frac{3\Theta}{\eta(T+1)} + \frac{9\eta L^2}{2}.$$

Our choices $\eta = \frac{\epsilon}{9L^2}$ and $T \ge \frac{6\Theta}{\eta\epsilon}$ imply that the right hand side is at most ϵ, as required. □

B.2.2 Proof of Proposition 10

Proposition 10. *Let $\mathcal{O}$ be an $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle with respect to gradient mapping g, distance-generating r with range at most Θ and some $\varepsilon_{\text{inner}} \le \varepsilon_{\text{outer}}$. Let $z_{1/2}, z_{3/2}, \ldots, z_{K-1/2}$ be iterates of Algorithm 11 and let $\bar{z}_K$ be its output. Then*

$$\mathbb{E}\,\mathrm{Gap}(\bar{z}_K) \le \mathbb{E}\max_{u\in\mathcal{Z}}\frac{1}{K}\sum_{k=1}^{K}\left\langle g(z_{k-1/2}), z_{k-1/2} - u\right\rangle \le \frac{\alpha\Theta}{K} + \varepsilon_{\text{outer}}.$$

Proof. For some iteration k, we have by the optimality conditions on $z_k^\star$ that

$$\left\langle g(z_{k-1/2}), z_k^\star - u \right\rangle \le \alpha \left(V_{z_{k-1}}(u) - V_{z_k^\star}(u) - V_{z_{k-1}}(z_k^\star) \right) \quad \forall u \in \mathcal{Z}.$$

Summing, writing $\left\langle g(z_{k-1/2}), z_k^\star - u \right\rangle = \left\langle g(z_{k-1/2}), z_{k-1/2} - u \right\rangle - \left\langle g(z_{k-1/2}), z_{k-1/2} - z_k^\star \right\rangle$, and rearranging yields

$$\begin{aligned} \sum_{k=1}^{K} \left\langle g(z_{k-1/2}), z_{k-1/2} - u \right\rangle \le & \alpha V_{z_0}(u) + \sum_{k=1}^{K} \alpha \left(V_{z_k}(u) - V_{z_k^\star}(u) \right) \\ & + \sum_{k=1}^{K} \left(\left\langle g(z_{k-1/2}), z_{k-1/2} - z_k^\star \right\rangle - \alpha V_{z_{k-1}}(z_k^\star) \right), \end{aligned} \tag{B.9}$$

for all $u \in \mathcal{Z}$. Since z_0 minimizes r, the first term is bounded by $V_{z_0}(u) \le r(u) - r(z_0) \le \Theta$. The second term is bounded by the definition of z_k in Algorithm 11:

$$\sum_{k=1}^{K} \alpha \left(V_{z_k}(u) - V_{z_k^\star}(u) \right) \le K(\varepsilon_{\text{outer}} - \varepsilon_{\text{inner}}).$$

Thus, maximizing (B.9) over u and then taking an expectation yields

$$\begin{aligned} \mathbb{E} \max_{u \in \mathcal{Z}} \sum_{k=1}^{K} \left\langle g(z_{k-1/2}), z_{k-1/2} - u \right\rangle \le & \alpha\Theta + K(\varepsilon_{\text{outer}} - \varepsilon_{\text{inner}}) \\ & + \sum_{k=1}^{K} \mathbb{E} \left[\left\langle g(z_{k-1/2}), z_{k-1/2} - z_k^\star \right\rangle - \alpha V_{z_{k-1}}(z_k^\star) \right]. \end{aligned}$$

Finally, by Definition 6, $\mathbb{E}\left[\left\langle g(z_{k-1/2}), z_{k-1/2} - z_k^\star \right\rangle - \alpha V_{z_{k-1}}(z_k^\star)\right] \le \varepsilon_{\text{inner}}$ for every k, and the result follows by dividing by K. □

B.2.3 Proof of Proposition 11

We provide a convergence result for the variance-reduced stochastic mirror descent scheme in Algorithm 12. We first state the following helper bound which is an application of Lemma 230. It is immediate from the variance bound of local-centered estimators (Property 2 of Definition 5) and the fact that all local norms (whether the domains are balls or simplices) are quadratic.

Lemma 229. *For any $w \in \mathcal{Z}$, (L, ϵ)-centered-local estimator $\tilde{g}_{w_0}$ satisfies*

$$\mathbb{E} \left\| \tilde{g}_{w_0}(z) - g(z) \right\|_w^2 \le L^2 V_{w_0}(z).$$

Lemma 230. *Let $\|\cdot\|_D$ be a quadratic norm in a diagonal matrix, e.g. for some $D = \operatorname{diag}(d)$ and*

$d \geq 0$ entrywise, let $\|x\|_D^2 = \sum d_i x_i^2$. Then, if X is a random vector, we have

$$\mathbb{E}\,\|X - \mathbb{E}[X]\|_D^2 \leq \mathbb{E}\,\|X\|_D^2\,.$$

Proof. This follows from the definition of variance:

$$\mathbb{E}\,\|X - \mathbb{E}[X]\|_D^2 = \mathbb{E}\,\|X\|_D^2 - \|\mathbb{E}[X]\|_D^2 \leq \mathbb{E}\,\|X\|_D^2\,.$$

□

Proposition 11. *Let $(\mathcal{Z},\ \|\cdot\|,\ r,\ \Theta,\ \mathrm{clip})$ be any local norm setup. Let $w_0 \in \mathcal{Z}$, $\alpha \geq \varepsilon_{\mathrm{inner}} > 0$, and $\tilde{g}_{w_0}$ be an L-centered-local estimator for some $L \geq \alpha$. Assume the domain is bounded by $\max_{z\in\mathcal{Z}} \|z\| \leq D$, that g is L-Lipschitz, i.e. $\|g(z) - g(z')\|_* \leq L\,\|z - z'\|$, that g is LD-bounded, i.e. $\max_{z\in\mathcal{Z}} \|g(z)\|_* \leq LD$, and that $\hat{w}_0 = w_0$. Then, for $\eta = \frac{\alpha}{10L^2}$, $T \geq \frac{6}{\eta\alpha} \geq \frac{60L^2}{\alpha^2}$, and $\varphi = \frac{\varepsilon_{\mathrm{inner}}}{6}$, Algorithm 12 outputs a point $\hat{w} \in \mathcal{Z}$ such that*

$$\mathbb{E} \max_{u\in\mathcal{Z}} \left[\langle g(\tilde{w}), \tilde{w} - u\rangle - \alpha V_{w_0}(u)\right] \leq \varepsilon_{\mathrm{inner}}, \tag{3.27}$$

i.e. Algorithm 12 is an $(\alpha, \varepsilon_{\mathrm{inner}})$-relaxed proximal oracle.

Proof. For any $u \in \mathcal{Z}$, and defining $\tilde{\Delta}_t := \tilde{g}(\hat{w}_t) - g(w_0)$ and $\Delta_t := g(\hat{w}_t) - g(w_0)$, we have

$$\begin{aligned}\sum_{t\in[T]} \langle \eta g(w_t), w_t - u\rangle = &\sum_{t\in[T]} \left\langle \mathrm{clip}(\eta\tilde{\Delta}_t) + \eta g(w_0), w_t - u\right\rangle + \sum_{t\in[T]} \left\langle \eta\Delta_t - \mathrm{clip}(\eta\tilde{\Delta}_t), w_t - u\right\rangle \\ &+ \sum_{t\in[T]} \langle \eta g(w_t) - \eta g(\hat{w}_t), w_t - u\rangle\,.\end{aligned} \tag{B.10}$$

We proceed to bound the three terms on the right hand side of (B.10) in turn. For the first term, recall the guarantees for the "ideal" iterates of Algorithm 12,

$$w_t^\star = \mathrm{argmin}_{w\in\mathcal{Z}} \left\{ \left\langle \mathrm{clip}(\eta\tilde{\Delta}_t) + \eta g(w_0), w\right\rangle + \frac{\alpha\eta}{2} V_{w_0}(w) + V_{w_{t-1}}(w)\right\}.$$

By using the optimality conditions of these iterates, defining $Q(z) := \langle \eta g(w_0), z\rangle + \frac{\alpha\eta}{2} V_{w_0}(z)$, $\gamma_t := \mathrm{clip}(\eta\tilde{\Delta}_t)$, and defining for notational convenience $w_{T+1}^\star := u$,

$$\begin{aligned}\sum_{t\in[T]} \langle \gamma_{t-1} + \nabla Q(w_t^\star), w_t^\star - u\rangle &\leq \sum_{t\in[T]} \left\langle -\nabla V_{w_{t-1}}(w_t^\star), w_t^\star - u\right\rangle \\ &= \sum_{t\in[T]} \left(V_{w_{t-1}}(u) - V_{w_t^\star}(u) - V_{w_{t-1}}(w_t^\star)\right) \\ &= V_{w_0}(u) + \sum_{t\in[T]} \left(V_{w_t}(u) - V_{w_t^\star}(u)\right) - \sum_{t=0}^{T} V_{w_t}(w_{t+1}^\star).\end{aligned} \tag{B.11}$$

We thus have the chain of inequalities, recalling $\gamma_0 = 0$,

$$\begin{aligned}
&\sum_{t\in[T]} \Big\langle \mathrm{clip}(\eta\tilde{\Delta}_t) + \eta g(w_0), w_t - u \Big\rangle + \frac{\alpha\eta}{2} \sum_{t\in[T]} \langle \nabla V_{w_0}(w_t^\star), w_t^\star - u\rangle \\
=&\sum_{t=0}^{T} \langle \gamma_t, w_t - u\rangle + \sum_{t\in[T]} \langle \nabla Q(w_t^\star), w_t^\star - u\rangle + \sum_{t\in[T]} \langle \eta g(w_0), w_t - w_t^\star\rangle \\
\overset{(i)}{\leq}& V_{w_0}(u) + \sum_{t\in[T]} \big(V_{w_t}(u) - V_{w_t^\star}(u)\big) + \sum_{t=0}^{T} \big(\langle \gamma_t, w_t - w_{t+1}^\star\rangle - V_{w_t}(w_{t+1}^\star)\big) + \sum_{t\in[T]} \langle \eta g(w_0), w_t - w_t^\star\rangle \\
\overset{(ii)}{\leq}& V_{w_0}(u) + 2\eta\varphi T + \sum_{t=0}^{T} \Big(\langle \mathrm{clip}(\eta\tilde{\Delta}_t), w_t - w_{t+1}^\star\rangle - V_{w_t}(w_{t+1}^\star)\Big) \overset{(iii)}{\leq} V_{w_0}(u) + 2\eta\varphi T + \sum_{t=0}^{T} \eta^2 \left\|\tilde{\Delta}_t\right\|_{w_t}^2 .
\end{aligned} \tag{B.12}$$

Here, (i) was by rearranging (B.11) via the equality

$$\sum_{t\in[T]} \langle \gamma_{t-1}, w_t^\star - u\rangle = \sum_{t=0}^{T} \langle \gamma_t, w_t - u\rangle - \sum_{t=0}^{T} \big\langle \gamma_t, w_t - w_{t+1}^\star \big\rangle ,$$

(ii) was by the conditions $\max_u \left[V_{w_t}(u) - V_{w_t^\star}(u)\right] \leq \eta\varphi$ and $\|w_t - w_t^\star\| \leq \frac{\varphi}{LD}$ satisfied by the iterates, and (iii) was by the property of clipping (3.18), as defined in the problem setup. Now by rearranging and using the three-point property of Bregman divergence (i.e. $\langle -\nabla V_{w'}(w), w - u\rangle = V_{w'}(u) - V_w(u) - V_{w'}(w)$), it holds that

$$\begin{aligned}
\sum_{t\in[T]} \Big\langle \mathrm{clip}(\eta\tilde{\Delta}_t) + \eta g(w_0), w_t - u \Big\rangle \leq& V_{w_0}(u) + 2\eta\varphi T + \eta^2 \sum_{t=0}^{T} \left\|\tilde{\Delta}_t\right\|_{w_t}^2 + \frac{\alpha\eta}{2} \sum_{t\in[T]} \left(V_{w_0}(u) - V_{w_0}(w_t^\star)\right) \\
\leq& V_{w_0}(u) + 3\eta\varphi T + \eta^2 \sum_{t=0}^{T} \left\|\tilde{\Delta}_t\right\|_{w_t}^2 + \frac{\alpha\eta}{2} \sum_{t\in[T]} \left(V_{w_0}(u) - V_{w_0}(\hat{w}_t)\right),
\end{aligned} \tag{B.13}$$

where the second inequality follows from the condition $V_{w_0}(\hat{w}_t) - V_{w_0}(w_t^\star) \leq \frac{2\varphi}{\alpha}$ satisfied by iterates of Algorithm 12. To bound the second term of (B.10), we define the ghost iterate sequnce $\{s_t\}$ by

$$s_t = \mathrm{argmin}_{s\in\mathcal{Z}} \left\{ \frac{1}{2}\big\langle \eta\Delta_{t-1} - \mathrm{clip}(\eta\tilde{\Delta}_{t-1}), s\big\rangle + V_{s_{t-1}}(s) \right\} \quad \text{with} \quad s_0 = w_0.$$

Applying Lemma 228 with $Q = 0$ and $\gamma_t = \frac{1}{2}(\eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t))$, and observing that again $\gamma_0 = 0$,

$$\begin{aligned}
&\sum_{t\in[T]} \left\langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), s_t - u \right\rangle \\
\leq& 2V_{w_0}(u) + \sum_{t=0}^{T} \left\{ \langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), s_t - s_{t+1}\rangle - 2V_{s_t}(s_{t+1}) \right\} \\
\leq& 2V_{w_0}(u) + \eta^2 \sum_{t=0}^{T} \left(\|\Delta_t\|_*^2 + \left\|\tilde{\Delta}_t\right\|_{s_t}^2 \right).
\end{aligned}$$

Here, we used properties (3.18) and (3.19). Consequently,

$$\begin{aligned}
&\sum_{t\in[T]} \left\langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), w_t - u \right\rangle \\
&= \sum_{t\in[T]} \left\langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), w_t - s_t \right\rangle + \sum_{t\in[T]} \left\langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), s_t - u \right\rangle \\
&\leq 2V_{w_0}(u) + \eta^2 \sum_{t=0}^{T} \left(\|\Delta_t\|_*^2 + \left\|\tilde{\Delta}_t\right\|_{s_t}^2 \right) + \sum_{t\in[T]} \left\langle \eta\Delta_t - \operatorname{clip}(\eta\tilde{\Delta}_t), w_t - s_t \right\rangle.
\end{aligned} \tag{B.14}$$

To bound the third term of (B.10), we use the condition $\|w_t - \hat{w}_t\| \leq \frac{\varphi}{LD}$ which implies

$$\sum_{t\in[T]} \langle \eta g(w_t) - \eta g(\hat{w}_t), w_t - u\rangle \leq \sum_{t\in[T]} \|\eta g(w_t) - \eta g(\hat{w}_t)\|_* \|w_t - u\| \leq 2\eta\varphi T. \tag{B.15}$$

Combining our three bounds (B.13), (B.14), and (B.15) in the context of (B.10), using $\hat{w}_0 = w_0$ and $\tilde{g}(w_0) = g(w_0)$, and finally dividing through by ηT, we obtain

$$\begin{aligned}
&\frac{1}{T}\sum_{t\in[T]} \langle g(w_t), w_t - u\rangle - \left(\frac{3}{\eta T} + \frac{\alpha}{2}\right) V_{w_0}(u) \\
&\leq 5\varphi + \frac{1}{T}\sum_{t\in[T]} \left(\eta \left\|\tilde{\Delta}_t\right\|_{w_t}^2 + \eta \left\|\tilde{\Delta}_t\right\|_{s_t}^2 + \eta \|\Delta_t\|_*^2 + \left\langle \Delta_t - \frac{1}{\eta}\operatorname{clip}(\eta\tilde{\Delta}_t), w_t - s_t \right\rangle - \frac{\alpha}{2} V_{w_0}(\hat{w}_t) \right).
\end{aligned} \tag{B.16}$$

Since $T \geq \frac{6}{\alpha\eta}$, taking a supremum over $u \in \mathcal{Z}$ in (B.16) and then an expectation yields

$$\begin{aligned}
&\mathbb{E}\sup_{u\in\mathcal{Z}} \left[\frac{1}{T}\sum_{t\in[T]} \langle g(w_t), w_t - u\rangle - \alpha V_{w_0}(u) \right] \leq 5\varphi \\
&+\frac{1}{T}\mathbb{E}\left[\sum_{t\in[T]} \eta \left\|\tilde{\Delta}_t\right\|_{w_t}^2 + \eta \left\|\tilde{\Delta}_t\right\|_{s_t}^2 + \eta \|\Delta_t\|_*^2 + \left\langle \Delta_t - \frac{1}{\eta}\operatorname{clip}(\eta\tilde{\Delta}_t), w_t - s_t \right\rangle - \frac{\alpha}{2} V_{w_0}(\hat{w}_t) \right].
\end{aligned} \tag{B.17}$$

We will show the second line of (B.17) is nonpositive. To do so, observe for each $t \in [T]$, by the

property (3.20) of clip$(\cdot)$, since conditional on w_t, s_t, $\tilde{\Delta}_t$ is unbiased for deterministic Δ_t,

$$\left|\mathbb{E}\left\langle \Delta_t - \frac{1}{\eta}\mathrm{clip}(\eta\tilde{\Delta}_t), w_t - s_t\right\rangle\right| = \left|\mathbb{E}\left\langle \tilde{\Delta}_t - \frac{1}{\eta}\mathrm{clip}(\eta\tilde{\Delta}_t), w_t - s_t\right\rangle\right| \le \eta\left\|\tilde{\Delta}_t\right\|_{w_t}^2 + \eta\left\|\tilde{\Delta}_t\right\|_{s_t}^2. \tag{B.18}$$

Finally, by using property 2 of the centered-local estimator $\tilde{\Delta}_t$, as well as Remark 1, we have for each $t \in [T]$,

$$\mathbb{E}\left[\eta\left\|\tilde{\Delta}_t\right\|_{w_t}^2\right] \le \eta L^2 V_{w_0}(\hat{w}_t),\ \mathbb{E}\left[\eta\left\|\tilde{\Delta}_t\right\|_{s_t}^2\right] \le \eta L^2 V_{w_0}(\hat{w}_t),\ \text{and}\ \eta\left\|\Delta_t\right\|_*^2 \le \eta L^2 V_{w_0}(\hat{w}_t). \tag{B.19}$$

Using bounds (B.18) and (B.19) in (B.17), as well as $\eta \le \frac{\alpha}{10L^2}$,

$$\mathbb{E}\sup_{u\in\mathcal{Z}}\left[\frac{1}{T}\sum_{t\in[T]}\langle g(w_t), w_t - u\rangle - \alpha V_{w_0}(u)\right] \le 5\varphi. \tag{B.20}$$

For the final claim, denote the true average iterate by $\bar{w} := \frac{1}{T}\sum_{t\in[T]} w_t$. We have $\forall u \in \mathcal{Z}$,

$$\begin{aligned}\langle g(\tilde{w}), \tilde{w} - u\rangle &\overset{(i)}{=} -\langle g(\tilde{w}), u\rangle = \langle g(\bar{w}) - g(\tilde{w}), u\rangle + \langle g(\bar{w}), \bar{w} - u\rangle \\ &\overset{(ii)}{\le} \varphi + \langle g(\bar{w}), \bar{w} - u\rangle \\ &= \frac{1}{T}\sum_{t\in[T]}\langle g(w_t), w_t - u\rangle + \varphi.\end{aligned}$$

Here, (i) used the fact that linearity of g gives $\langle g(z), z\rangle = 0$, $\forall z \in \mathcal{Z}$, and (ii) used Hölder's inequality $\langle g(\bar{w}) - g(\tilde{w}), u\rangle \le \|g(\bar{w}) - g(\tilde{w})\|_* \|u\| \le 2LD\|\tilde{w} - \bar{w}\| \le \varphi$ following from the approximation guarantee $\|\tilde{w} - \bar{w}\| \le \frac{\varphi}{2LD}$. Combining with (B.20) yields the conclusion, as $6\varphi = \varepsilon_{\text{inner}}$.

□

B.3 Deferred proofs for sublinear methods

B.3.1 ℓ_2-ℓ_2 sublinear coordinate method

Assumptions. The algorithm in this section will assume access to entry queries, ℓ_1 norms of rows and columns, and ℓ_1 sampling distributions for rows and columns. Further, it assumes the ability to sample a row or column proportional to its squared ℓ_1 norm; given access to all ℓ_1 norms, the algorithm may spend $O(m+n)$ constructing these sampling oracles in $O(m+n)$ time, which does not affect its asymptotic runtime. We use the ℓ_2-ℓ_2 local norm setup (Table 3.6). We define

$$L_{\mathsf{co}}^{2,2} := \sqrt{\sum_{i\in[m]}\|A_{i:}\|_1^2 + \sum_{j\in[n]}\|A_{:j}\|_1^2}. \tag{B.21}$$

Gradient estimator

For $z \in \mathbb{B}^n \times \mathbb{B}^m$, we specify two distinct choices of sampling distributions $p(z), q(z)$ which obtain the optimal Lipschitz constant. The first one is an oblivious distribution:

$$p_{ij}(z) := \frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{\|A_{:j}\|_1^2}{\sum_{k\in[n]} \|A_{:k}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}. \tag{B.22}$$

The second one is a dynamic distribution:

$$p_{ij}(z) := \frac{[z^{\mathsf{y}}]_i{}^2}{\|z^{\mathsf{y}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{[z^{\mathsf{x}}]_j{}^2}{\|z^{\mathsf{x}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}. \tag{B.23}$$

We now state the local properties of each estimator.

Lemma 231. *In the ℓ_2-ℓ_2 setup, estimator* (3.26) *using the sampling distribution in* (B.22) *or* (B.23) *is an $L_{\mathsf{co}}^{2,2}$-local estimator.*

Proof. For convenience, we restate the distributions here: they are respectively

$$p_{ij}(z) := \frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{\|A_{:j}\|_1^2}{\sum_{k\in[n]} \|A_{:k}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}$$

and

$$p_{ij}(z) := \frac{[z^{\mathsf{y}}]_i{}^2}{\|z^{\mathsf{y}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{[z^{\mathsf{x}}]_j{}^2}{\|z^{\mathsf{x}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}.$$

Unbiasedness holds by definition. We first show the variance bound on the x block for distribution (B.22):

$$\begin{aligned}
\mathbb{E}\left[\|\tilde{g}^{\mathsf{x}}(z)\|_2^2\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z) \cdot \left(\frac{A_{ij}[z^{\mathsf{y}}]_i}{p_{ij}(z)}\right)^2 = \sum_{i\in[m],j\in[n]} \frac{A_{ij}^2[z^{\mathsf{y}}]_i^2}{p_{ij}(z)} \\
&= \sum_{i\in[m],j\in[n]} \frac{|A_{ij}|}{\|A_{i:}\|_1}[z^{\mathsf{y}}]_i^2 \cdot \left(\sum_{i\in[m]} \|A_{i:}\|_1^2\right) = \sum_{i\in[m]} \|A_{i:}\|_1^2.
\end{aligned}$$

Similarly, we have

$$\mathbb{E}\left[\|\tilde{g}^{\mathsf{y}}(z)\|_2^2\right] \le \sum_{j\in[n]} \|A_{:j}\|_1^2.$$

Now, we show the variance bound on the x block for distribution (B.23):

$$\mathbb{E}\left[\|\tilde{g}^{\mathsf{x}}(z)\|_2^2\right] = \sum_{i\in[m],j\in[n]} p_{ij}(z)\cdot\left(\frac{A_{ij}[z^{\mathsf{y}}]_i}{p_{ij}(z)}\right)^2 = \sum_{i\in[m],j\in[n]}\frac{A_{ij}^2[z^{\mathsf{y}}]_i^2}{p_{ij}(z)}$$
$$= \sum_{i\in[m],j\in[n]} |A_{ij}|\|A_{i:}\|_1\,\|z^{\mathsf{y}}\|_2^2 \le \sum_{i\in[m]}\|A_{i:}\|_1^2,$$

and a similar bound holds on the y block. □

We remark that using the oblivious distribution (B.22) saves a logarithmic factor in the runtime compared to the dynamic distribution, so for the implementation of all of our ℓ_2-ℓ_2 algorithms we will use the oblivious distribution.

Implementation details

In this section, we discuss the details of how to leverage the $\texttt{IterateMaintainer}_2$ data structure to implement the iterations of our algorithm. The algorithm we analyze is Algorithm 10, using the local estimator defined in (3.26), and the distribution (B.22). We choose

$$\eta = \frac{\epsilon}{9\left(L_{\mathsf{co}}^{2,2}\right)^2} \text{ and } T = \left\lceil\frac{6\Theta}{\eta\epsilon}\right\rceil \ge \frac{54\left(L_{\mathsf{co}}^{2,2}\right)^2}{\epsilon^2}.$$

Lemma 231 implies that our estimator satisfies the remaining requirements for Proposition 9, giving the duality gap guarantee in T iterations. In order to give a runtime bound, we claim that each iteration can be implemented in constant time, with $O(m+n)$ additional runtime.

Data structure initializations and invariants. At the start of the algorithm, we spend $O(m+n)$ time initializing data structures via $\texttt{IM}_2^{\mathsf{x}}.\texttt{Init}(\mathbf{0}_n, b)$, $\texttt{IM}_2^{\mathsf{y}}.\texttt{Init}(\mathbf{0}_m, c)$, where $\texttt{IM}_2^{\mathsf{x}}, \texttt{IM}_2^{\mathsf{y}}$ are instantiations of $\texttt{IterateMaintainer}_2$ data structures. Throughout, we preserve the invariant that the points maintained by $\texttt{IM}_2^{\mathsf{x}}, \texttt{IM}_2^{\mathsf{y}}$ correspond to the x and y blocks of the current iterate z_t at iteration t of the algorithm. We note that we instantiate data structures which do not support $\texttt{Sample}()$.

Iterations. For simplicity, we only discuss the runtime of updating the x block as the y block follows symmetrically. We divide each iteration into the following substeps, each of which we show run in constant time. We refer to the current iterate by $z = (z^{\mathsf{x}}, z^{\mathsf{y}})$, and the next iterate by $w = (w^{\mathsf{x}}, w^{\mathsf{y}})$.

Sampling. Because the distribution is oblivious, sampling both i and $j \mid i$ using precomputed data structures takes constant time.

Computing the gradient estimator. To compute $c := A_{ij}[z^{\mathsf{y}}]_i/p_{ij}$, it suffices to compute A_{ij}, $[z^{\mathsf{y}}]_i$, and p_{ij}. Using an entry oracle for A obtains A_{ij} in constant time, and calling $\texttt{IM}_2^{\mathsf{y}}.\texttt{Get}(i)$ takes constant time. Computing p_{ij} using the precomputed row norms and the values of $A_{ij}, [z^{\mathsf{y}}]_i$ takes

constant time.

Performing the update. For the update corresponding to a proximal step, we have

$$w^{\mathsf{x}} \leftarrow \Pi_{\mathcal{X}}\left(z^{\mathsf{x}} - \eta\tilde{g}^{\mathsf{x}}(z)\right) = \frac{z^{\mathsf{x}} - \eta\tilde{g}^{\mathsf{x}}(z)}{\max\{\left\|z^{\mathsf{x}} - \eta\tilde{g}^{\mathsf{x}}(z)\right\|_2, 1\}}.$$

We have computed $\tilde{g}^{\mathsf{x}}(z)$, so to perform this update, we call

$$\begin{aligned}
&\texttt{IM}_2^{\mathsf{x}}.\texttt{AddSparse}(j, -\eta c);\\
&\texttt{IM}_2^{\mathsf{x}}.\texttt{AddDense}(-\eta);\\
&\texttt{IM}_2^{\mathsf{x}}.\texttt{Scale}(\max\{\texttt{IM}^{\mathsf{x}}.\texttt{GetNorm}(), 1\}^{-1});\\
&\texttt{IM}_2^{\mathsf{x}}.\texttt{UpdateSum}().
\end{aligned}$$

By assumption, each operation takes constant time because we do not support `Sample` in our instances of $\texttt{IM}_2$, giving the desired iteration complexity. It is clear that at the end of performing these operations, the invariant that $\texttt{IM}_2^{\mathsf{x}}$ maintains the x block of the iterate is preserved.

Averaging. After T iterations, we compute the average point $\bar{z}^{\mathsf{x}}$:

$$[\bar{z}^{\mathsf{x}}]_j \leftarrow \frac{1}{T} \cdot \texttt{IM}_2^{\mathsf{x}}.\texttt{GetSum}(j), \forall j \in [n].$$

By assumption, this takes $O(n)$ time.

Algorithm guarantee

Theorem 76. *In the ℓ_2-ℓ_2 setup, the implementation in Section B.3.1 has runtime*

$$O\left(\frac{\left(L_{\mathsf{co}}^{2,2}\right)^2}{\epsilon^2} + m + n\right)$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \leq \epsilon.$$

Proof. The runtime bound follows from the discussion in Section B.3.1. The correctness follows from Proposition 9. □

Remark 11. *Using our $\texttt{IterateMaintainer}_2$ data structure, the ℓ_2-ℓ_2 algorithm of [55] runs in time $O\left(\mathsf{rcs}\|A\|_F^2/\epsilon^2\right)$. Our runtime universally improves upon it since*

$$\sum_{i\in[m]} \|A_{i:}\|_1^2 + \sum_{j\in[n]} \|A_{:j}\|_1^2 \leq 2\mathsf{rcs}\,\|A\|_{\mathrm{F}}^2.$$

B.3.2 ℓ_2-ℓ_1 sublinear coordinate method

Assumptions. The algorithm in this section will assume access to every oracle listed in Section 3.2.3. However, for a specific matrix A, only one of three sampling distributions will be used in the algorithm; we describe the specific oracle requirements of each distribution following their definition. We use the ℓ_2-ℓ_1 local norm setup (Table 3.6). Throughout this section, we will assume that the linear term in (3.26) is $g(0) = 0$ uniformly.

Finally, in this section we assume access to a weighted variant of $\texttt{IterateMaintainer}_2$, which takes a nonnegative weight vector w as a static parameter. $\texttt{WeightedIterateMaintainer}_2$ supports two modified operations compared to the data structure $\texttt{IterateMaintainer}_2$: its $\texttt{GetNorm}()$ operation returns $\sqrt{\sum_j [w]_j [x]_j^2}$, and its $\texttt{Sample}()$ returns coordinate j with probability proportional to $[w]_j [x]_j^2$ (cf. Section 3.2.4). We give the implementation of this extension in Appendix B.8.

Gradient estimator

For $z \in \mathbb{B}^n \times \Delta^m$ and desired accuracy $\epsilon > 0$, we specify three distinct choices of sampling distributions $p(z), q(z)$. Each of our distributions induces an estimator with different properties.

The first one is

$$p_{ij}(z) := \frac{|A_{ij}|}{\|A_{i:}\|_1} \cdot [z^{\mathsf{y}}]_i \quad \text{and} \quad q_{ij}(z) := \frac{A_{ij}^2}{\|A\|_{\mathrm{F}}^2}. \tag{B.24}$$

The second one is

$$p_{ij}(z) := \frac{|A_{ij}|}{\|A_{i:}\|_1} \cdot [z^{\mathsf{y}}]_i \quad \text{and} \quad q_{ij}(z) := \frac{[z^{\mathsf{x}}]_j^2 \cdot \mathbf{1}_{\{A_{ij} \neq 0\}}}{\sum_{l \in [n]} \mathrm{cs}_l \cdot [z^{\mathsf{x}}]_l^2}. \tag{B.25}$$

Here, we let $\mathrm{cs}_j \leq \mathsf{rcs}$ denote the number of nonzero elements in column $A_{:j}$. The third one is

$$p_{ij}(z) := \frac{|A_{ij}|}{\|A_{i:}\|_1} \cdot [z^{\mathsf{y}}]_i \quad \text{and} \quad q_{ij}(z) := \frac{|A_{ij}| \cdot [z^{\mathsf{x}}]_j^2}{\sum_{l \in [n]} \|A_{:l}\|_1 \cdot [z^{\mathsf{x}}]_l^2}. \tag{B.26}$$

For $L_{\mathsf{co}}^{2,1,(1)}, L_{\mathsf{co}}^{2,1,(2)}$, and $L_{\mathsf{co}}^{2,1,(3)}$ to be defined, the estimators induced by these distributions are local estimators whose guarantees depend on these constants respectively. Furthermore, these Lipschitz constants are in general incomparable and depend on specific properties of the matrix. Therefore, we may choose our definition of $L_{\mathsf{co}}^{2,1}$ to be the minimum of these constants, by choosing an appropriate estimator. We now state the local properties of each estimator.

Lemma 232. *In the ℓ_2-ℓ_1 setup, estimator* (3.26) *using the sampling distributions in* (B.24), (B.25),

or (B.26) *is respectively a* $L_{\mathsf{co}}^{2,1,(k)}$*-local estimator, for* $k \in \{1,2,3\}$*, and*

$$L_{\mathsf{co}}^{2,1,(1)} := \sqrt{\max_{i\in[m]} \|A_{i:}\|_1^2 + \|A\|_{\mathrm{F}}^2},$$
$$L_{\mathsf{co}}^{2,1,(2)} := \sqrt{2\mathsf{rcs} \max_{i\in[m]} \|A_{i:}\|_2^2},$$
$$L_{\mathsf{co}}^{2,1,(3)} := \sqrt{\max_{i\in[m]} \|A_{i:}\|_1^2 + \left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right)}.$$

Proof. First, we give the proof for the sampling distribution (B.24). Unbiasedness holds by definition. For the x block, we have the variance bound:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{g}^{\mathsf{x}}(z)\|_2^2\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z)\cdot\left(\frac{A_{ij}[z^{\mathsf{y}}]_i}{p_{ij}(z)}\right)^2 = \sum_{i\in[m],j\in[n]} |A_{ij}|\|A_{i:}\|_1[z^{\mathsf{y}}]_i \\ &= \sum_{i\in[m]} \|A_{i:}\|_1^2[z^{\mathsf{y}}]_i \le \max_{i\in[m]} \|A_{i:}\|_1^2.\end{aligned}$$

For arbitrary w^{y}, we have the variance bound on the y block:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{g}^{\mathsf{y}}(z)\|_{w^{\mathsf{y}}}^2\right] &= \sum_{i\in[m],j\in[n]} q_{ij}(z)\cdot\left([w^{\mathsf{y}}]_i\cdot\left(\frac{A_{ij}[z^{\mathsf{x}}]_j}{q_{ij}(z)}\right)^2\right) = \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i\frac{A_{ij}^2[z^{\mathsf{x}}]_j^2}{q_{ij}(z)} \\ &= \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i[z^{\mathsf{x}}]_j^2\,\|A\|_{\mathrm{F}}^2 \le \|A\|_{\mathrm{F}}^2\,.\end{aligned}$$

Next, we give the proof for the sampling distribution (B.25). Unbiasedness holds by definition. By Cauchy-Schwarz and our earlier proof, we have the variance bound for the x block:

$$\mathbb{E}\left[\|\tilde{g}^{\mathsf{x}}(z)\|_2^2\right] \le \max_{i\in[m]} \|A_{i:}\|_1^2 \le \mathsf{rcs}\max_{i\in[m]} \|A_{i:}\|_2^2\,.$$

For arbitrary w^{y}, we have the variance bound on the y block, where $S_i := \left\{j \mid \mathbf{1}_{A_{ij}\neq 0} = 1\right\}$:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{g}^{\mathsf{y}}(z)\|_{w^{\mathsf{y}}}^2\right] &= \sum_{i\in[m],j\in S_i} q_{ij}(z)\cdot\left([w^{\mathsf{y}}]_i\cdot\left(\frac{A_{ij}[z^{\mathsf{x}}]_j}{q_{ij}(z)}\right)^2\right) = \sum_{i\in[m],j\in S_i} [w^{\mathsf{y}}]_i\frac{A_{ij}^2[z^{\mathsf{x}}]_j^2}{q_{ij}(z)} \\ &\le \sum_{i\in[m],j\in S_i} [w^{\mathsf{y}}]_i A_{ij}^2\mathsf{rcs} \le \mathsf{rcs}\max_{k\in[m]} \|A_{k:}\|_2^2\,.\end{aligned}$$

Finally, we give the proof for the sampling distribution (B.26). Unbiasedness and the variance bound

for the x block again hold. For arbitrary w^{y}, we have the variance bound on the y block:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{g}^{\mathsf{y}}(z)\|_{w^{\mathsf{y}}}^2\right] &= \sum_{i\in[m],j\in[n]} q_{ij}(z)\cdot\left([w^{\mathsf{y}}]_i\cdot\left(\frac{A_{ij}[z^{\mathsf{x}}]_j}{q_{ij}(z)}\right)^2\right) = \sum_{i\in[m],j\in[n]}[w^{\mathsf{y}}]_i\frac{A_{ij}^2[z^{\mathsf{x}}]_j^2}{q_{ij}(z)}\\ &\le \left(\sum_{i\in[m],j\in[n]}[w^{\mathsf{y}}]_i|A_{ij}|\right)\left(\sum_{l\in[n]}\|A_{:l}\|_1[z^{\mathsf{x}}]_l^2\right)\\ &\le \left(\max_{k\in[m]}\|A_{k:}\|_1\right)\left(\max_{l\in[n]}\|A_{:l}\|_1\right).\end{aligned}$$

□

By using the definitions of $L_{\mathsf{co}}^{2,1,(1)}, L_{\mathsf{co}}^{2,1,(2)}$, and $L_{\mathsf{co}}^{2,1,(3)}$, we define the constant

$$L_{\mathsf{co}}^{2,1} := \sqrt{\max_{i\in[m]}\|A_{i:}\|_1^2 + \min\left(\|A\|_{\mathrm{F}}^2, \mathsf{rcs}\max_{i\in[m]}\|A_{i:}\|_2^2, \left(\max_{i\in[m]}\|A_{i:}\|_1\right)\left(\max_{j\in[n]}\|A_{:j}\|_1\right)\right)}. \tag{B.27}$$

In particular, by choosing whichever of the distributions (B.24), (B.25), or (B.26) yields the minimial Lipschitz constant, we may always ensure we have a $L_{\mathsf{co}}^{2,1}$-local estimator. We now discuss the specific precomputed quantities each estimator requires, among those listed in Section 3.2.3. All distributions require access to entry queries, ℓ_1 norms of rows, and ℓ_1 sampling distributions for rows.

- Using the sampling distribution (B.24) requires additional access to ℓ_2 sampling distributions for rows and columns and the Frobenius norm of A.
- Using the sampling distribution (B.25) requires additional access to uniform sampling nonzero entries of columns.
- Using the sampling distribution (B.26) requires additional access to ℓ_1 norms of columns and ℓ_1 sampling distributions for columns.

Implementation details

In this section, we discuss the details of how to leverage the appropriate `IterateMaintainer`$_1$ and `IterateMaintainer`$_2$ data structures to implement the iterations of our algorithm. The algorithm we analyze is Algorithm 10, using the local estimator defined in (3.26), and the best choice of distribution among (B.24), (B.25), (B.26). We choose

$$\eta = \frac{\epsilon}{9\left(L_{\mathsf{co}}^{2,1}\right)^2} \text{ and } T = \left\lceil\frac{6\Theta}{\eta\epsilon}\right\rceil \ge \frac{54\left(L_{\mathsf{co}}^{2,1}\right)^2\log(2m)}{\epsilon^2}.$$

Lemma 232 implies that our estimator satisfies the remaining requirements for Proposition 9, giving the duality gap guarantee in T iterations. In order to give a runtime bound, we claim that each iteration can be implemented in $O(\log mn)$ time, with $O(m+n)$ additional runtime. For simplicity, because most of the algorithm implementation details are exactly same as the discussion of Section 3.4.1 for the simplex block $y \in \mathcal{Y}$, and exactly the same as the discussion of Section B.3.1 for the ball block $x \in \mathcal{X}$, we discuss the differences here, namely the implementations of sampling and gradient computation.

We assume that we have initialized $\texttt{IM}_1^{\mathsf{y}}$, an instantiation of $\texttt{IterateMaintainer}_1$, and $\texttt{IM}_2^{\mathsf{x}}$, an instantiation of $\texttt{IterateMaintainer}_2$. When the choice of distribution is (B.25), we also assume access to $\texttt{WIM}_2^{\mathsf{x}}$, an instantiation of $\texttt{WeightedIterateMaintainer}_2$ initialized with the weight vector of nonzero counts of columns of the matrix; similarly, for distribution (B.26) we instantiate a $\texttt{WeightedIterateMaintainer}_2$ with the weight vector of ℓ_1 norms of each column.

Sampling. Recall that

$$p_{ij}(z) := \frac{|A_{ij}|}{\|A_{i:}\|_1} \cdot [z^{\mathsf{y}}]_i.$$

We first sample coordinate i via $\texttt{IM}_1^{\mathsf{y}}.\texttt{Sample}()$ in $O(\log m)$, and then sample j using the data structure corresponding to $A_{i:}$ in $O(1)$. Next, to sample from the distribution

$$q_{ij}(z) := \frac{A_{ij}^2}{\|A\|_{\mathrm{F}}^2}$$

required by (B.24), we can sample a coordinate of the matrix proportional to its square in constant time using our matrix access. To sample from the distribution

$$q_{ij}(z) := \frac{[z^{\mathsf{x}}]_j^2 \cdot \mathbf{1}_{\{A_{ij} \neq 0\}}}{\sum_{l \in [n]} \mathrm{cs}_l \cdot [z^{\mathsf{x}}]_l^2}$$

required by (B.25), we first sample coordinate j via $\texttt{WIM}_2^{\mathsf{x}}.\texttt{Sample}()$ in $O(\log n)$, and then uniformly sample a coordinate i amongst the entries of $A_{:j}$ for which the indicator labels as nonzero. Finally, to sample from the distribution

$$q_{ij}(z) := \frac{|A_{ij}| \cdot [z^{\mathsf{x}}]_j^2}{\sum_{l \in [n]} \|A_{:l}\|_1 \cdot [z^{\mathsf{x}}]_l^2}$$

required by (B.26), we sample coordinate j via $\texttt{WIM}_2^{\mathsf{x}}.\texttt{Sample}()$, and then sample a coordinate i proportional to its absolute value using a column sampling oracle.

Computing the gradient estimator. By the proofs of Theorem 10 and Theorem 76, it suffices to compute $p_{i^{\mathsf{x}}j^{\mathsf{x}}}, q_{i^{\mathsf{y}}j^{\mathsf{y}}}$ in constant time. Calling $\texttt{IM}_2^{\mathsf{x}}.\texttt{Get}(j)$, $\texttt{IM}_1^{\mathsf{y}}.\texttt{Get}(i)$, $\texttt{IM}_2^{\mathsf{x}}.\texttt{GetNorm}()$, and $\texttt{WIM}_2^{\mathsf{x}}.\texttt{GetNorm}()$

when appropriate, and using access to precomputation allows us to obtain all relevant quantities for the computations in $O(1)$.

Algorithm guarantee

Theorem 77. *In the ℓ_2-ℓ_1 setup, the implementation in Section B.3.2 has runtime*

$$O\left(\frac{\left(L_{\mathsf{co}}^{2,1}\right)^2 \log m \log(mn)}{\epsilon^2} + m + n\right)$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \leq \epsilon.$$

Proof. The runtime bound follows from the discussion in Section B.3.2. The correctness follows from Proposition 9. □

Remark 12. *Using our* `IterateMaintainer`$_1$ *and* `IterateMaintainer`$_2$ *data structures, the ℓ_2-ℓ_1 algorithm of Clarkson et al. [140] runs in time $O(\mathsf{rcs}\max_{i\in[m]}\|A_{i:}\|_2^2 \log^2(mn)/\epsilon^2)$. By noting the definition of $L_{\mathsf{co}}^{2,1,(2)}$, our runtime universally improves upon it since $\left(L_{\mathsf{co}}^{2,1}\right)^2 \leq 2\mathsf{rcs}\max_{i\in[m]}\|A_{i:}\|_2^2$.*

B.4 Deferred proofs for variance-reduced methods

B.4.1 Helper proofs

Lemma 32. *For $y, y' \in \Delta^m$, divergence $V_y(y')$ generated by $r(y) = \sum_{i\in[m]}[y]_i \log[y]_i - [y]_i$ satisfies*

$$V_y(y') \geq \frac{1}{2}\|y' - y\|^2_{\frac{3}{2y+y'}} = \frac{1}{2}\sum_{i\in[m]}\frac{([y]_i - [y']_i)^2}{\frac{2}{3}[y]_i + \frac{1}{3}[y']_i}.$$

Proof. Let $\gamma \in \mathbb{R}^m$. Note that for every $\tau \in [0,1]$ (with elementwise multiplication, division and square root), $\langle \gamma, y - y'\rangle = \left\langle \gamma\sqrt{(1-\tau)y + \tau y'}, \frac{y-y'}{\sqrt{(1-\tau)y+\tau y'}}\right\rangle$. Therefore, using $2\langle u, w\rangle \leq \|u\|_2^2 + \|w\|_2^2$, we have for every $\tau \in [0,1]$,

$$2\langle \gamma, y - y'\rangle \leq \sum_{i\in[m]}\left((1-\tau)[y]_i + \tau[y']_i\right)[\gamma]_i^2 + \sum_{i\in[m]}\frac{([y]_i - [y']_i)^2}{(1-\tau)[y]_i + \tau[y']_i}.$$

Applying the double integral $\int_0^1 dt \int_0^t d\tau$ to both sides of the inequality, and using $\int_0^1 dt\int_0^t 1\cdot d\tau = \frac{1}{2}$ and $\int_0^1 dt \int_0^t \tau \cdot d\tau = \frac{1}{6}$ gives

$$\langle \gamma, y - y'\rangle \leq \sum_{i\in[m]}\left(\frac{1}{3}[y]_i + \frac{1}{6}[y']_i\right)[\gamma]_i^2 + \int_0^1 dt\int_0^t \sum_{i\in[m]}\frac{([y]_i - [y']_i)^2}{(1-\tau)[y]_i + \tau[y']_i}d\tau.$$

Identifying the double integral with the expression

$$V_y(y') = \sum_{i\in[m]} \left(y'_i \log \frac{y'_i}{y_i} + y_i - y'_i \right) = \int_0^1 dt \int_0^t \sum_{i\in[m]} \frac{(y_i - y'_i)^2}{(1-\tau)y_i + \tau y'_i} d\tau. \tag{B.28}$$

for the divergence induced by entropy, the result follows by choosing $[\gamma]_i = \frac{[y]_i - [y']_i}{\frac{2}{3}[y]_i + \frac{1}{3}[y']_i}$. □

Lemma 34. *Let $x' \in \Delta^n$ be a β-padding of $x \in \Delta^n$. Then,*

$$\sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} x_j \log x_j \le \frac{\beta n}{e} + \beta(1+\beta).$$

Proof. Letting $\tilde{x}$ be the point inducing x' in Definition 3, we have

$$\begin{aligned}\sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} x_j \log x_j = &\left(\sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} \tilde{x}_j \log \tilde{x}_j \right) \\ &+ \left(\sum_{j\in[n]} \tilde{x}_j \log \tilde{x}_j - \sum_{j\in[n]} x_j \log x_j \right).\end{aligned}$$

We bound these two terms separately. For the first term, let $\|\tilde{x}\|_1 = 1 + b$, for some $b \le \beta$; we see that entrywise, $(1+b)x'_j = \tilde{x}_j$. For each $j \in [n]$,

$$\begin{aligned} x'_j \log x'_j - \tilde{x}_j \log \tilde{x}_j &= x'_j \log x'_j - (1+b)x'_j \log\left((1+b)x'_j\right) \\ &= b x'_j \log \frac{1}{x'_j} - (1+b)x'_j \log(1+b) \\ &\le b x'_j \log \frac{1}{x'_j} \le \frac{\beta}{e}. \end{aligned}$$

The first inequality was due to nonnegativity of $(1+b)\log(1+b)$ and x'_j, and the second was due to the maximum value of the scalar function $z \log \frac{1}{z}$ over the nonnegative reals being $1/e$. Summing over all coordinates yields that the first term is bounded by $\beta n/e$.

For the second term, we have by integration that entrywise

$$\begin{aligned} \tilde{x}_j \log \tilde{x}_j - x_j \log x_j &= \int_{\alpha=0}^1 (1 + \log(x_j + \alpha(\tilde{x}_j - x_j)))(\tilde{x}_j - x_j) d\alpha \\ &\le \int_{\alpha=0}^1 (1 + \log(\tilde{x}_j))(\tilde{x}_j - x_j) d\alpha \\ &\le \int_{\alpha=0}^1 \tilde{x}_j (\tilde{x}_j - x_j) d\alpha \le (1+\beta)|\tilde{x}_j - x_j|. \end{aligned}$$

The first inequality is by $\tilde{x}_j \ge x_j$ for all $j \in [n]$ and $\log(x)$ is monotone in $x > 0$; the second is by

$\log(x) \le x - 1$ for all $x > 0$; the third again uses $\tilde{x}_j \ge x_j$ and that $\tilde{x}_j \le \|\tilde{x}\|_1 \le 1 + \beta$, and the second condition in Definition 3. Finally, combining yields the desired

$$\sum_{j\in[n]} x'_j \log x'_j - \sum_{j\in[n]} x_j \log x_j \le \frac{\beta n}{e} + \beta(1+\beta).$$

□

B.4.2 ℓ_2-ℓ_2 variance-reduced coordinate method

Assumptions. As in Section B.3.1, the algorithm in this section will assume access to entry queries, ℓ_1 norms of rows and columns, and ℓ_1 sampling distributions for rows and columns, and the ability to sample a row or column proportional to its squared ℓ_1 norm. We use the ℓ_2-ℓ_2 local norm setup (cf. Table 3.6). Again, we define

$$L_{\mathsf{co}}^{2,2} := \sqrt{\sum_{i\in[m]} \|A_{i:}\|_1^2 + \sum_{j\in[n]} \|A_{:j}\|_1^2}.$$

Gradient estimator

Given reference point $w_0 \in \mathbb{B}^n \times \mathbb{B}^m$, for $z \in \mathbb{B}^n \times \mathbb{B}^m$, we specify two distinct sampling distributions $p(z; w_0), q(z; w_0)$ which obtain the optimal Lipschitz constant. The first one is an oblivious distribution:

$$p_{ij}(z; w_0) := \frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z; w_0) := \frac{\|A_{:j}\|_1^2}{\sum_{k\in[n]} \|A_{:k}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}. \tag{B.29}$$

The second one is a dynamic distribution:

$$p_{ij}(z; w_0) := \frac{[w_0^{\mathsf{y}} - z^{\mathsf{y}}]_i^2}{\|w_0^{\mathsf{y}} - z^{\mathsf{y}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z; w_0) := \frac{[w_0^{\mathsf{x}} - z^{\mathsf{x}}]_j^2}{\|w_0^{\mathsf{x}} - z^{\mathsf{x}}\|_2^2} \cdot \frac{|A_{ij}|}{\|A_{:j}\|_1}. \tag{B.30}$$

We now state the local properties of each estimator.

Lemma 233. *In the ℓ_2-ℓ_2 setup, estimator* (3.28) *using the sampling distribution in* (B.29) *or* (B.30) *is a $\sqrt{2}L_{\mathsf{co}}^{2,2}$-centered-local estimator.*

Proof. Unbiasedness holds by definition in both cases. We first show the variance bound on the x

block for distribution (B.29):

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{x}}_{w_0}(z) - g^{\mathsf{x}}(w_0)\right\|_2^2\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z;w_0)\cdot\left(\frac{A_{ij}[z^{\mathsf{y}}-w_0^{\mathsf{y}}]_i}{p_{ij}(z;w_0)}\right)^2 = \sum_{i\in[m],j\in[n]}\frac{A_{ij}^2[z^{\mathsf{y}}-w_0^{\mathsf{y}}]_i^2}{p_{ij}(z;w_0)}\\ &= \sum_{i\in[m],j\in[n]}\frac{|A_{ij}|}{\|A_{i:}\|_1}[z^{\mathsf{y}}-w_0^{\mathsf{y}}]_i^2\cdot\left(\sum_{k\in[m]}\|A_{k:}\|_1^2\right)\\ &= \left(\sum_{i\in[m]}\|A_{i:}\|_1^2\right)\|z^{\mathsf{y}}-w_0^{\mathsf{y}}\|_2^2\,.\end{aligned}$$

Similarly, we have

$$\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{y}}_{w_0}(z) - g^{\mathsf{y}}(w_0)\right\|_2^2\right] \le \left(\sum_{j\in[n]}\|A_{:j}\|_1^2\right)\|z^{\mathsf{x}}-w_0^{\mathsf{x}}\|_2^2\,.$$

Combining these and using $\|z^{\mathsf{x}}-w_0^{\mathsf{x}}\|_2^2+\|z^{\mathsf{y}}-w_0^{\mathsf{y}}\|_2^2 = 2V_{w_0}(z)$ yields the desired variance bound. Now, we show the variance bound on the x block for distribution (B.30):

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{x}}_{w_0}(z) - g^{\mathsf{x}}(w_0)\right\|_2^2\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z;w_0)\cdot\left(\frac{A_{ij}[z^{\mathsf{y}}-w_0^{\mathsf{y}}]_i}{p_{ij}(z;w_0)}\right)^2 = \sum_{i\in[m],j\in[n]}\frac{A_{ij}^2[z^{\mathsf{y}}-w_0^{\mathsf{y}}]_i^2}{p_{ij}(z;w_0)}\\ &= \sum_{i\in[m],j\in[n]}|A_{ij}|\|A_{i:}\|_1\|z^{\mathsf{y}}-w_0^{\mathsf{y}}\|_2^2\\ &= \left(\sum_{i\in[m]}\|A_{i:}\|_1^2\right)\|z^{\mathsf{y}}-w_0^{\mathsf{y}}\|_2^2\,.\end{aligned}$$

and a similar bound holds on the y block. □

Again, for algorithmic considerations (i.e. an additional logarithmic factor in the complexity of sampling from (B.30)), we will only discuss using the oblivious distribution (B.29) in our algorithm.

Implementation details

In this section, we discuss the details of how to leverage the `IterateMaintainer`$_2$ data structure to implement the iterations of our algorithm. The algorithm we analyze is Algorithm 11 with $K = \alpha\Theta/\epsilon$, using Algorithm 12 as an $(\alpha, 0)$-relaxed proximal oracle. In the implementation of Algorithm 12, we use the centered-local gradient estimator defined in (B.29). For each use of Algorithm 12, we choose

$$\eta = \frac{\alpha}{20\left(L_{\mathsf{co}}^{2,2}\right)^2} \text{ and } T = \left\lceil\frac{6}{\eta\alpha}\right\rceil \ge \frac{120\left(L_{\mathsf{co}}^{2,2}\right)^2}{\alpha^2}. \tag{B.31}$$

Our discussion will follow in three steps: first, we discuss the complexity of all executions in Algorithm 11 other than the calls to the oracles, as well as the initialization procedure for each inner loop. Next, we discuss the complexity of each iteration of Algorithm 12. Finally, we discuss the complexity of computing the average iterate in each run of Algorithm 12. For simplicity, when discussing Algorithm 12, we will only discuss implementation of the x-block, and the y-block will follow symmetrically. Altogether, the guarantees of Proposition 10 and Proposition 11 imply that if the guarantees required by the algorithm hold, the expected gap of the output is bounded by ϵ.

Outer loop extragradient steps and inner loop data structures. Overall, we execute $K = \alpha\Theta/\epsilon$ iterations of Algorithm 11, and let $\varepsilon_{\text{outer}} = \varepsilon_{\text{inner}} = 0$ to obtain the desired gap, where $\Theta = 1$ in the ℓ_2-ℓ_2 setup. We spend $O(\mathsf{nnz})$ time executing each extragradient step in Algorithm 11 exactly, where the dominant term in the runtime is the computation of each $g(z_{k-1/2})$, for $k \in [K]$. Also, we can maintain the average point $\bar{z}$ throughout the duration of the algorithm, in $O(m+n)$ time per iteration. At the beginning of each inner loop, we initialize a data structure $\texttt{IM}_2^{\mathsf{x}}$ which does not support sampling, an instance of $\texttt{IterateMaintainer}_2$, with $\texttt{IM}_2^{\mathsf{x}}.\texttt{Init}(w_0^{\mathsf{x}}, v)$, for

$$v = (1-\kappa)w_0^{\mathsf{x}} - \eta\kappa g^{\mathsf{x}}(w_0),$$

where $\kappa := \frac{1}{1+\eta\alpha/2}$. The inner loop will preserve the invariant that the point maintained by $\texttt{IM}_2^{\mathsf{x}}$ is the x block of the current inner loop iterate w_t in each iteration t. To motivate this initialization, we recall the form of the updates,

$$w_{t+1}^{\mathsf{x}} \leftarrow \Pi_{\mathcal{X}}\left(\kappa\left(w_t^{\mathsf{x}} + \left(\frac{1}{\kappa} - 1\right)w_0^{\mathsf{x}} - \eta\tilde{g}_{w_0}^{\mathsf{x}}(w_t)\right)\right), \tag{B.32}$$

where $\Pi_{\mathcal{X}}(w) = \frac{w}{\max\{1, \|w\|_2\}}$, and the fixed dense part of $\tilde{g}_{w_0}^{\mathsf{x}}(w_t)$ is $g^{\mathsf{x}}(w_0)$. Therefore, in the following discussion we will be able to maintain this difference via a scaling by κ, an appropriate addition of the scaled dense vector, and a sparse update.

Finally, we also store the vector w_0 in full, supporting entry queries.

Inner loop iterations. Each inner loop iteration consists of sampling indices for the computation of $\tilde{g}_{w_0}$, computing the sparse part of $\tilde{g}_{w_0}$, and performing the update to the iterate. We show that we can run each substep in constant time. Then, this implies that the total complexity of the inner loop, other than initializing the data structures and outputting the average iterate, is

$$O(T) = O\left(\frac{\left(L_{\mathsf{co}}^{2,2}\right)^2}{\alpha^2}\right).$$

We discuss how to make appropriate modifications to the x-block. For simplicity we denote our

current iterate as z, and the next iterate as w. Recall that the distribution is given by

$$p_{ij}(z; w_0) := \frac{\|A_{i:}\|_1^2}{\sum_{k\in[m]} \|A_{k:}\|_1^2} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1}.$$

Sampling. By using precomputed distributions, we can sample $i \propto \|A_{i:}\|_1^2$ and then $j \mid i \propto |A_{ij}|$ in constant time.

Computing the gradient estimator. Computing the sparse component of the gradient estimator 3.28 requires computing A_{ij}, $[z^{\mathsf{y}} - w_0^{\mathsf{y}}]_i$, and $p_{ij}(z; w_0)$. Using appropriate use of precomputed access to entries and row norms (it is clear we may pay $O(m+n)$ at the beginning of the algorithm to store the sum $\sum_{k\in[m]} \|A_{k:}\|_1^2$), entry $[w_0^{\mathsf{y}}]_i$, and $\mathtt{IM}_2^{\mathsf{y}}.\mathtt{Get}(i)$ allows us to perform the required computation of the sparse component

$$c := [\tilde{g}_{w_0}^{\mathsf{x}}(z) - g(w_0)]_j$$

in constant time, by assumption.

Performing the update. In order to perform the update, we recall the form of the update given by (B.32). Thus, it suffices to call

$$\begin{aligned}
&\mathtt{IM}_2^{\mathsf{x}}.\mathtt{Scale}(\kappa);\\
&\mathtt{IM}_2^{\mathsf{x}}.\mathtt{AddDense}(1);\\
&\mathtt{IM}_2^{\mathsf{x}}.\mathtt{AddSparse}(j, -\kappa\eta c);\\
&\mathtt{IM}_2^{\mathsf{x}}.\mathtt{Scale}(\max\{\mathtt{IM}_2^{\mathsf{x}}.\mathtt{GetNorm}(), 1\}^{-1});\\
&\mathtt{IM}_2^{\mathsf{x}}.\mathtt{UpdateSum}()
\end{aligned}$$

By assumption, each operation takes constant time. By the discussion in the data structure initialization section, it is clear that we preserve the invariant that the point maintained by $\mathtt{IM}_2^{\mathsf{x}}$ is the x block of the current iterate.

Average iterate computation. At the end of each run of Algorithm 12, we spend $O(n)$ time computing and returning the average iterate via appropriate calls to $\mathtt{IM}_2^{\mathsf{x}}.\mathtt{GetSum}(j)$ for each $j \in [n]$, and scaling by $1/T$. This operation is asymptotically dominated by the $O(\mathtt{nnz}(A))$ cost of the extragradient step.

Algorithm guarantee

Theorem 78. *In the ℓ_2-ℓ_2 setup, the implementation in Section B.4.2 with the optimal choice of $\alpha = \max\{\epsilon, L_{\mathsf{co}}^{2,2}\sqrt{1/\mathsf{nnz}}\}$ has runtime*

$$O\left(\left(\mathsf{nnz} + \frac{\left(L_{\mathsf{co}}^{2,2}\right)^2}{\alpha^2}\right)\frac{\alpha}{\epsilon}\right) = O\left(\mathsf{nnz} + \frac{\sqrt{\mathsf{nnz}}L_{\mathsf{co}}^{2,2}}{\epsilon}\right)$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that

$$\mathbb{E}\mathrm{Gap}(\bar{z}) \le \epsilon.$$

Proof. The correctness of the algorithm is given by the discussion in Section B.4.2 and the guarantees of Proposition 10 and Proposition 11. The runtime bound is given by the discussion in Section B.4.2, and the optimal choice of α is clear. □

To better understand the strengths of our runtime guarantee, Proposition 55 shows that Theorem 78 implies a universal improvement for ℓ_2-ℓ_2 games compared to accelerated gradient descent for matrices A with nonnegative entries (or more generally, for A with $\||A|\|_{\mathrm{op}} = O(\|A\|_{\mathrm{op}})$).

Proposition 55. *For any $A \in \mathbb{R}^{m\times n}$, we have*

$$L_{\mathsf{co}}^{2,2} := \max\left\{\sqrt{\sum_i \|A_{i:}\|_1^2}, \sqrt{\sum_j \|A_{:j}\|_1^2}\right\} \le \sqrt{m+n}\cdot\||A|\|_{\mathrm{op}}.$$

Proof. Denote $\mathbf{1}_k$ as the all 1 vector in $\mathbb{R}^k$. We have the following sequence of inequalities:

$$\sqrt{\sum_{i\in[m]}\|A_{i:}\|_1^2} = \left\||A|^\top\mathbf{1}_m\right\|_2 = \max_{x\in\mathbb{B}^n}\mathbf{1}_m^\top|A|x \le \|\mathbf{1}_m\|_2\max_{x\in\mathbb{B}^n}\||A|x\|_2 \le \sqrt{m}\,\||A|\|_{\mathrm{op}}.$$

Similarly, bounding $\max_{y\in\mathbb{B}^n} y^\top|A|\mathbf{1}_n$ implies $\sqrt{\sum_{j\in[n]}\|A_{:j}\|_1^2} \le \sqrt{n}\,\||A|\|_{\mathrm{op}}$. Taking a maximum and using $\max\{\sqrt{m},\sqrt{n}\} \le \sqrt{m+n}$ implies the result. □

Remark 13. *For matrix $A \in \mathbb{R}^{m\times n}$, combining the guarantees of Theorem 78 with the bound from Proposition 55 implies a runtime bounded by*

$$O\left(\mathsf{nnz} + \frac{\sqrt{\mathsf{nnz}\cdot(m+n)}\,\||A|\|_{\mathrm{op}}}{\epsilon}\right).$$

Whenever $\|A\|_{\mathrm{op}} \ge \||A|\|_{\mathrm{op}}$, this is an improvement by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$ compared to the accelerated full-gradient method (c.f. Table 3.2), which obtains a runtime of $O(\mathsf{nnz}\cdot\|A\|_{\mathrm{op}}/\epsilon)$. This applies without any sparsity or numerical sparsity assumptions, and is the same speedup factor as we obtain for ℓ_1-ℓ_1 and ℓ_2-ℓ_1 games using a variance reduction framework with row and column based

gradient estimators in Appendix B.5. The ℓ_2-ℓ_2 variance reduction algorithms of Appendix B.5 and [55] do not offer such improvements, and our improvement stems from our coordinate-based gradient estimators and our data structure design.

B.4.3 ℓ_2-ℓ_1 variance-reduced coordinate method

Assumptions. The algorithm in this section will assume access to entry queries, ℓ_1 norms of rows, ℓ_2 sampling distributions for rows and columns, and the Frobenius norm of A. We use the ℓ_2-ℓ_1 local norm setup (cf. Table 3.6). Again, we define

$$L_{\mathsf{co}}^{2,1,(1)} := \sqrt{\max_{i\in[m]} \|A_{i:}\|_1^2 + \|A\|_{\mathrm{F}}^2}, \tag{B.33}$$

$$L_{\mathsf{co}}^{2,1,(2)} := \sqrt{2\mathsf{rcs} \max_{i\in[m]} \|A_{i:}\|_2^2}, \tag{B.34}$$

$$L_{\mathsf{co}}^{2,1,(3)} := \sqrt{\max_{i\in[m]} \|A_{i:}\|_1^2 + \left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right)}. \tag{B.35}$$

Finally, in this section we assume access to a centered variant of $\texttt{WeightedIterateMaintainer}_2$, which takes a point x_0 as a static parameter, where x_0 is in the space as the iterates x maintained. $\texttt{CenteredIterateMaintainer}_2$ supports two additional operations compared to the data structure $\texttt{WeightedIterateMaintainer}_2$: $\texttt{Sample}()$ returns coordinate j with probability proportional to $[w]_j[x-x_0]_j^2$ (cf. Section 3.2.4) in $O(\log n)$ time, and we may query $\|x-x_0\|_w^2$ in constant time, where w is a specified weight vector. We give the implementation of this extension in Appendix B.8.

Gradient estimator

Given reference point $w_0 \in \mathbb{B}^n \times \Delta^m$, for $z \in \mathbb{B}^n \times \Delta^m$ and a parameter $\alpha > 0$, as in Section B.3.2, we specify three distinct choices of sampling distributions $p(z; w_0), q(z; w_0)$.

The first one is

$$p_{ij}(z; w_0) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z; w_0) := \frac{A_{ij}^2}{\|A\|_{\mathrm{F}}^2}. \tag{B.36}$$

The second one is

$$p_{ij}(z; w_0) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{[z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2 \cdot \mathbf{1}_{\{A_{ij}\neq 0\}}}{\sum_{l\in[n]} \mathrm{cs}_l \cdot [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_l^2}. \tag{B.37}$$

As in Section B.3.2, cs_j is the number of nonzeros of $A_{:j}$. The third one is

$$p_{ij}(z;w_0) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1} \quad \text{and} \quad q_{ij}(z) := \frac{|A_{ij}| \cdot [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2}{\sum_{l\in[n]} \|A_{:l}\|_1 \cdot [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_l^2}. \tag{B.38}$$

We now state the local properties of each estimator.

Lemma 234. *In the ℓ_2-ℓ_1 setup, estimator* (3.28) *using the sampling distributions in* (B.36), (B.37), *or* (B.38) *is respectively a $\sqrt{2}L_{\mathsf{co}}^{2,1,(k)}$-centered-local estimator, for $k \in \{1,2,3\}$.*

Proof. First, we give the proof for the sampling distribution (B.36). Unbiasedness holds by definition. For the x block, we have the variance bound:

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}_{w_0}^{\mathsf{x}}(z) - g^{\mathsf{x}}(w_0)\right\|_2^2\right] &= \sum_{i\in[m],j\in[n]} p_{ij}(z;w_0)\left(\frac{A_{ij}[z^{\mathsf{y}} - w_0^{\mathsf{y}}]_i}{p_{ij}(z;w_0)}\right)^2 = \sum_{i\in[m],j\in[n]} \frac{A_{ij}^2[z^{\mathsf{y}} - w_0^{\mathsf{y}}]_i^2}{p_{ij}(z;w_0)} \\ &\le 2\max_{i\in[m]} \|A_{i:}\|_1^2 \, V_{w_0^{\mathsf{y}}}(z^{\mathsf{y}}),\end{aligned}$$

where in the last inequality we used Lemma 32.

For arbitrary w^{y}, we have the variance bound on the y block:

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}_{w_0}^{\mathsf{y}}(z) - g^{\mathsf{y}}(w_0)\right\|_{w^{\mathsf{y}}}^2\right] &= \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i \frac{A_{ij}^2[z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2}{q_{ij}(z;w_0)} \\ &= \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2 \|A\|_{\mathrm{F}}^2 \le 2\|A\|_{\mathrm{F}}^2 \, V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}}).\end{aligned}$$

Combining these and using

$$\|\tilde{g}_{w_0}(z) - g(w_0)\|_w^2 := \|\tilde{g}_{w_0}(z)^{\mathsf{x}} - g(w_0)^{\mathsf{x}}\|_2^2 + \|\tilde{g}_{w_0}(z)^{\mathsf{y}} - g(w_0)^{\mathsf{y}}\|_{w^{\mathsf{y}}}^2$$

yields the desired variance bound. For the remaining two distributions, the same argument demonstrates unbiasedness and the variance bound for the x block. For sampling distribution (B.37) and arbitrary w^{y}, we have the variance bound on the y block:

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}_{w_0}^{\mathsf{y}}(z) - g^{\mathsf{y}}(w_0)\right\|_{w^{\mathsf{y}}}^2\right] &= \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i \frac{A_{ij}^2[z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2}{q_{ij}(z;w_0)} \\ &\le \left(\sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i A_{ij}^2\right)\left(\mathsf{rcs}\sum_{j\in[n]} [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2\right) \\ &\le 2\mathsf{rcs}\max_{i\in[m]} \|A_{i:}\|_2^2 \, V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}}).\end{aligned}$$

Finally, for sampling distribution (B.38), we have the variance bound on the y block:

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}^{\mathsf{y}}_{w_0}(z) - g^{\mathsf{y}}(w_0)\right\|^2_{w^{\mathsf{y}}}\right] &= \sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i \frac{A_{ij}^2[z^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2}{q_{ij}(z; w_0)} \\ &\le \left(\sum_{i\in[m],j\in[n]} [w^{\mathsf{y}}]_i |A_{ij}|\right)\left(\sum_{l\in[n]} \|A_{:l}\|_1 \cdot [z^{\mathsf{x}} - w_0^{\mathsf{x}}]_l^2\right) \\ &\le 2\left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right) V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}}).\end{aligned}$$

□

Finally, as in Section B.3.2, we define the constant

$$L_{\mathsf{co}}^{2,1} := \sqrt{\max_{i\in[m]} \|A_{i:}\|_1^2 + \min\left(\|A\|_{\mathrm{F}}^2, \mathsf{rcs}\max_{i\in[m]} \|A_{i:}\|_2^2, \left(\max_{i\in[m]} \|A_{i:}\|_1\right)\left(\max_{j\in[n]} \|A_{:j}\|_1\right)\right)},$$

and note that Lemma 234 implies that we can obtain a $\sqrt{2}L_{\mathsf{co}}^{2,1}$-centered-local estimator by appropriately choosing a sampling distribution depending on the minimizing parameter.

Implementation details

The algorithm we analyze is Algorithm 11 with $K = 3\alpha\Theta/\epsilon$, $\varepsilon_{\text{outer}} = 2\epsilon/3$ using Algorithm 12 as an $(\alpha, \varepsilon_{\text{inner}} = \epsilon/3)$-relaxed proximal oracle with $\varphi = \epsilon/18$. In the implementation of Algorithm 11, we again apply the $\mathsf{truncate}(\cdot, \delta)$ operation to each iterate $z_k^\star$, where the $\mathsf{truncate}$ operation only affects the y block; choosing $\delta = \frac{\varepsilon_{\text{outer}} - \varepsilon_{\text{inner}}}{\alpha m}$ suffices for its guarantees (see Section 3.4.2 for the relevant discussion). In the implementation of Algorithm 12, we use the centered-local gradient estimator defined in (3.28), using the sampling distribution amongst (B.36), (B.37), or (B.38) which attains the variance bound $L_{\mathsf{co}}^{2,1}$. For each use of Algorithm 12, we choose

$$\eta = \frac{\alpha}{20\left(L_{\mathsf{co}}^{2,1}\right)^2} \text{ and } T = \left\lceil \frac{6}{\eta\alpha} \right\rceil = \frac{120\left(L_{\mathsf{co}}^{2,1}\right)^2}{\alpha^2}.$$

For simplicity, because most of the algorithm implementation details are exactly the same as the discussion of Section 3.4.2 for the simplex block $y \in \mathcal{Y}$, and exactly the same as the discussion of Section B.4.2 for the ball block $x \in \mathcal{X}$, we discuss the differences here.

Outer loop extragradient steps. We execute $3\alpha\log(2m)/\epsilon$ iterations of Algorithm 11 to obtain the desired gap. We spend $O(\mathsf{nnz})$ time executing each extragradient step exactly, and then $O(m+n)$ time applying the $\mathsf{truncate}$ operation and maintaining the average point $\bar{z}$. When we initialize the inner loop, we also create a data structure supporting sampling from w_0^{y} in constant time.

Data structure initializations and invariants.

On the simplex block, we follow the strategy outlined in Section 3.4.2. We initialize our simplex maintenance data structure $\texttt{AEM}^{\mathsf{y}}(w_0^{\mathsf{y}}, v, \kappa, \tilde{\varepsilon})$ with parameters

$$\kappa := \frac{1}{1+\eta\alpha/2},\ v := (1-\kappa)\log w_0^{\mathsf{y}} - \eta\kappa g^{\mathsf{y}}(w_0),\ \tilde{\varepsilon} := (m+n)^{-8}.$$

We will again maintain the invariant that the data structures maintain "exact" and "approximate" points corresponding to the iterates of our algorithm. The correctness of this setting with respect to the requirements of Proposition 11, i.e. the approximation conditions in Line 5, 7 and 8 in Algorithm 12, follows from the discussion of Section 3.4.2; we note that the condition $\min_j [w_0^{\mathsf{x}}]_j \geq (m+n)^{-5} = \lambda$ again holds, and that $1-\kappa \geq (m+n)^{-8}$. Thus, for the parameter ω used in the interface of `ApproxExpMaintainer`, we have

$$\log(\omega) = \log\left(\max\left(\frac{1}{1-\kappa}, \frac{m}{\lambda\tilde{\varepsilon}}\right)\right) = O(\log(mn)).$$

On the ball block, we follow the strategy outlined in Section B.4.2, but instead of using an $\texttt{IterateMaintainer}_2$ on the x-block, we use $\texttt{CIM}_2^{\mathsf{x}}$, an instance of $\texttt{CenteredIterateMaintainer}_2$ data structure initialized with the point w_0^{x}, supporting the required sampling operation. For the sampling distribution (B.37), we use the weight vector of column nonzero counts, and for (B.38) we use the weight vector of column ℓ_1 norms. Overall, the complexity of the initializations on both blocks is bounded by $O(n + m\log^2(m)\log^2(mn))$.

Inner loop iterations.

We discuss how to sample from each of the distributions (B.36), (B.37), and (B.38) in $O(\log(m)\log(mn))$. Combining with the discussions of implementing the inner loop in Sections 3.4.2 and B.4.2, the total complexity of the inner loop, other than outputting the average iterate, is

$$\begin{aligned} &O\left(T\log^2(m)\log^2(mn) + \mathsf{nnz} + m\log(m)\log^2(mn)\right) \\ = O&\left(\frac{\left(L_{\mathsf{co}}^{2,1,(1)}\right)^2\log^2(m)\log^2(mn)}{\alpha^2} + \mathsf{nnz} + m\log(m)\log^2(mn)\right). \end{aligned}$$

As in the variance-reduced ℓ_1-ℓ_1 setting, the dominant term in the runtime is the complexity of calling $\texttt{AEM}^{\mathsf{y}}.\texttt{AddSparse}$ in each iteration. Recall that the distribution p in every case is given by

$$p_{ij}(z; w_0) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \cdot \frac{|A_{ij}|}{\|A_{i:}\|_1}$$

With probability $2/3$ we sample a coordinate i from the precomputed data structure for sampling from w_0^{y}, and otherwise we sample i via $\texttt{AEM}^{\mathsf{y}}.\texttt{Sample}()$. Then, we sample an entry j proportional to its magnitude from the ℓ_1 sampling oracle for $A_{i:}$ in constant time. The runtime is dominated by

$O(\log(m)\log(mn))$.

To sample from the distribution q in (B.36), we follow the outline in Section B.4.3. Similarly, for sampling from distributions (B.37) and (B.38), we follow the outline in Section B.4.3 but replace all calls to an `IterateMaintainer` instance with a call to $\texttt{CIM}_2^{\mathsf{x}}$ initialized with an appropriate weight vector. In all cases, the runtime is $O(\log m)$ which does not dominate the iteration complexity.

Finally, it is clear from discussions in previous sections that the iterate maintenance invariants of our data structures are preserved by the updates used in this implementation.

Algorithm guarantee

Theorem 79. *In the ℓ_2-ℓ_1 setup, let $\mathsf{nnz}' := \mathsf{nnz} + m\log(m)\log^2(mn)$. The implementation in Section B.4.3 with the optimal choice of $\alpha = \max\left(\epsilon/3, L_{\mathsf{co}}^{2,1}\log(m)\log(mn)/\sqrt{\mathsf{nnz}'}\right)$ has runtime*

$$O\left(\left(\mathsf{nnz}' + \frac{\left(L_{\mathsf{co}}^{2,1}\right)^2\log^2(m)\log^2(mn)}{\alpha^2}\right)\frac{\alpha\log(m)}{\epsilon}\right) = O\left(\mathsf{nnz}' + \frac{\sqrt{\mathsf{nnz}'}L_{\mathsf{co}}^{2,1}\log(mn)\log^2(m)}{\epsilon}\right)$$

and outputs a point $\bar{z} \in \mathcal{Z}$ such that

$$\mathbb{E}\left[\mathrm{Gap}(z)\right] \leq \epsilon.$$

Proof. The correctness of the algorithm is given by the discussion in Section B.4.3 and the guarantees of Proposition 10 with $K = 3\alpha\Theta/\epsilon$, $\varepsilon_{\text{outer}} = 2\epsilon/3$, $\varepsilon_{\text{inner}} = \epsilon/3$, Proposition 11 with $\varphi = \epsilon/18$ and data structure `ApproxExpMaintainer` with our choice of

$$\tilde{\varepsilon} := (m+n)^{-8}$$

to meet the approximation conditions in Line 5, 7 and 8 in Algorithm 12. The runtime bound is given by the discussion in Section B.4.3, and the optimal choice of α is clear. □

B.5 Additional results on variance-reduced methods

B.5.1 Row-column sparsity variance-reduced methods

By instantiating relaxed proximal oracles with row-column based gradient estimators, implemented with the data structures we develop in Section 3.5, we obtain the row-column sparsity based runtimes as stated in Table 3.2. Notably, up to logarithmic factors, we generically replace a dependence on $O(m+n)$ in a prior version of this work [111] with $O(\mathsf{rcs})$, where rcs is defined as the maximum number of nonzero entries for any row or column. In this section, we give implementation details, as well as the instantiation of our framework using row-column gradient estimators.

Our row-column estimators $\tilde{g}_{w_0}$ in this section, parameterized by reference point w_0, sample a full column or row of the matrix (rather than a coordinate). To compute $\tilde{g}_{w_0}(z)$ we sample $i \sim p(z)$

and $j \sim q(z)$ independently according to a specified distribution depending on the setup, and use the estimator

$$\tilde{g}_{w_0}(z) := \left(A^\top w_0^{\mathsf{y}} + A_{i:} \frac{[z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i}{p_i(w)}, -Aw_0^{\mathsf{x}} - A_{:j} \frac{[z^{\mathsf{x}}]_j - [w_0^{\mathsf{x}}]_j}{q_j(w)} \right), \tag{B.39}$$

The key difference between this estimator with that of Section 8 is that its difference with $g(w_0)$ is $O(\mathsf{rcs})$-sparse rather than $O(1)$-sparse, requiring `MultSparse` steps with $O(\mathsf{rcs})$-sparse vectors. In all other respects, the implementation details are exactly the same as those in Section 3.4.2 and Appendix B.4, so we omit them for brevity. We now state our sampling distributions used with the estimator form (B.39), and the corresponding centered local variance bounds.

In the ℓ_1-ℓ_1 setup, we use the sampling distribution (from reference point $w_0 \in \Delta^m \times \Delta^n$)

$$p_i(z) := \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \quad \text{and} \quad q_j(z) := \frac{[z^{\mathsf{x}}]_j + 2[w_0^{\mathsf{x}}]_j}{3}. \tag{B.40}$$

Lemma 235. *In the ℓ_1-ℓ_1 setup, gradient estimator* (B.39) *using the sampling distribution in* (B.40) *is a $\sqrt{2}\,\|A\|_{\max}$-centered-local estimator.*

Proof. Unbiasedness holds by definition. For the variance bound, it suffices to show that

$$\mathbb{E}\,\|\tilde{g}_{w_0}(z) - g(w_0)\|_\infty^2 \le 2\,\|A\|_{\max}^2\, V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}});$$

clearly this implies the weaker relative variance bound statement (along with an analogous bound on the y block). To this end, we have

$$\mathbb{E}\,\|\tilde{g}_{w_0}(z) - g(w_0)\|_\infty^2 \le \sum_{i\in[m]} \frac{\|A_{i:}\|_\infty^2\,[z^{\mathsf{y}} - w_0^{\mathsf{y}}]_i^2}{p_i(z)} \le 2\,\|A\|_{\max}^2\, V_{w_0^{\mathsf{x}}}(z^{\mathsf{x}}),$$

where the last inequality used Lemma 32. □

In the ℓ_2-ℓ_2 setup, we use the oblivious sampling distribution

$$p_i = \frac{\|A_{i:}\|_2^2}{\|A\|_{\mathrm{F}}^2} \quad \text{and} \quad q_j = \frac{\|A_{:j}\|_2^2}{\|A\|_{\mathrm{F}}^2}. \tag{B.41}$$

We proved that gradient estimator (B.39) using the sampling distribution in (B.41) admits a $\|A\|_{\mathrm{F}}$-centered estimator in [111], which is an equivalent definition to Definition 5 in the ℓ_2-ℓ_2 setup. For self-containment, we reproduce this proof here.

Lemma 236. *In the ℓ_2-ℓ_2 setup, gradient estimator* (B.39) *using the sampling distribution in* (B.41) *is a $\|A\|_{\mathrm{F}}$-centered estimator.*

Proof. Unbiasedness holds by definition. For the variance bound,

$$\begin{aligned}\mathbb{E}\left\|\tilde{g}_{w_0}(z) - g(w_0)\right\|_2^2 &= \sum_{i\in[m]} \frac{\|A_{i:}\|_2^2}{p_i}([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i)^2 + \sum_{j\in[n]} \frac{\|A_{:j}\|_2^2}{q_j}([z^{\mathsf{x}}]_j - [w_0^{\mathsf{x}}]_j)^2 \\ &= \|A\|_{\mathrm{F}}^2 \left\|z - w_0\right\|_2^2.\end{aligned}$$

□

In the ℓ_2-ℓ_1 setup, we use the sampling distribution (from reference point $w_0 \in \mathbb{B}^n \times \Delta^m$)

$$p_i(z) = \frac{[z^{\mathsf{y}}]_i + 2[w_0^{\mathsf{y}}]_i}{3} \quad \text{and} \quad q_j(z) = \frac{([z^{\mathsf{x}}]_j - [w_0^{\mathsf{x}}]_j)^2}{\|z^{\mathsf{x}} - w_0^{\mathsf{x}}\|_2^2}. \tag{B.42}$$

Lemma 237. *In the ℓ_2-ℓ_1 setup, gradient estimator* (B.39) *using the sampling distribution in* (B.42) *is a $\sqrt{2}L$-centered-local estimator with $L = \max_{i\in[m]} \|A_{i:}\|_2 = \|A\|_{2\to\infty}$.*

Proof. Unbiasedness holds by definition. For the variance bound, we first note

$$\begin{aligned}\mathbb{E}\left[\left\|\tilde{g}_{w_0}^{\mathsf{x}}(z) - g^{\mathsf{x}}(w_0)\right\|_2^2\right] &\le \sum_{i\in[m]} \|A_{i:}\|_2^2 \frac{([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i)^2}{\frac{1}{3}[z^{\mathsf{y}}]_i + \frac{2}{3}[w_0^{\mathsf{y}}]_i} \le \max_{i\in[m]} \|A_{i:}\|_2^2 \left(\sum_{i\in[m]} \frac{([z^{\mathsf{y}}]_i - [w_0^{\mathsf{y}}]_i)^2}{\frac{1}{3}[z^{\mathsf{y}}]_i + \frac{2}{3}[w_0^{\mathsf{y}}]_i}\right) \\ &\le 2 \max_{i\in[m]} \|A_{i:}\|_2^2 \, V_{w_0^{\mathsf{y}}}(z^{\mathsf{y}}),\end{aligned}$$

where for the last inequality we use Lemma 32. On the other block, we have

$$\max_{i\in[m]} \mathbb{E}\left[\tilde{g}_{w_0}^{\mathsf{y}}(w) - g^{\mathsf{y}}(w_0)\right]_i^2 \le \max_{i\in[m]} \sum_{j\in[n]} \frac{A_{ij}^2 [w^{\mathsf{x}} - w_0^{\mathsf{x}}]_j^2}{q_j(w)} = 2 \max_{i\in[m]} \|A_{i:}\|_2^2 \, V_{w_0^{\mathsf{x}}}(w^{\mathsf{x}}).$$

Summing these two bounds concludes the proof. □

B.5.2 Extensions with composite terms

In this section, we give a brief discussion of how to change Proposition 11 and implementations of the procedures in Sections 3.3.1 and 3.3.2 to handle modified regularization in the context of Proposition 19, and composite regularization terms in the objective in the methods of Section 3.6. Specifically we consider a composite optimization problem of the form:

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} y^\top A x + \mu^{\mathsf{x}}\phi(x) - \mu^{\mathsf{y}}\psi(y) \text{ where } \phi = V_{x'}^{\mathsf{x}} \text{ and } \psi = V_{y'}^{\mathsf{y}}.$$

For simplicity of notation we define $\Upsilon(x, y) := \mu^{\mathsf{x}}\phi(x) + \mu^{\mathsf{y}}\psi(y)$. We remark that $x' = 0$ recovers the case of $\phi = r^{\mathsf{x}}$ when $\mathcal{X} = \mathbb{B}^n$, and $x' = \frac{1}{n}\mathbf{1}$ recovers the case of $\phi = r^{\mathsf{x}}$ when $\mathcal{X} = \Delta^n$ (similarly setting y' allows us to recover this for the y block).

Changes to inner loop

In this section, we first discuss the necessary changes to Algorithm 12 and Proposition 11. For simplicity of notation, we denote $\rho := \sqrt{\mu^{\mathsf{x}}/\mu^{\mathsf{y}}}$, $\hat{V}^{\mathsf{x}} := \rho V^{\mathsf{x}}$, $\hat{V}^{\mathsf{y}} := \frac{1}{\rho} V^{\mathsf{y}}$, $\hat{V} := \hat{V}^{\mathsf{x}} + \hat{V}^{\mathsf{y}}$.

Algorithm 68: $\texttt{InnerLoop}(w_0, \tilde{g}_{w_0}, \varphi)$

Input: Initial $w_0 \in \mathcal{Z}$, (L, α)-centered-local gradient estimator $\tilde{g}_{w_0}$, oracle quality $\alpha > 0$;
Parameters: Step size η, number of iterations T, approximation tolerance φ;
Output: Point $\tilde{w}$ satisfying Definition 6;
for $t = 1, \dots, T$ **do**
  $\hat{w}_{t-1} \approx w_{t-1}$ satisfying $\hat{V}_{w_0}(\hat{w}_{t-1}) - \hat{V}_{w_0}(w_{t-1}) \le \frac{\varphi}{\alpha}$ and $\|\hat{w}_{t-1} - w_{t-1}\| \le \frac{\varphi}{LD}$;
  $w_t^\star \leftarrow \operatorname{argmin}\left\{\langle \operatorname{clip}(\eta \tilde{g}_{w_0}(\hat{w}_{t-1}) - \eta g(w_0)), w\rangle + \eta \Upsilon(w) + \frac{\eta\alpha}{2}\hat{V}_{w_0}(w) + \hat{V}_{w_{t-1}}(w)\right\}$;
  $w_t \approx w_t^\star$ satisfying $\max_u \left[\hat{V}_{w_t}(u) - \hat{V}_{w_t^\star}(u)\right] \le \frac{\varphi}{1+\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}$, $\hat{V}_{w_0}(w_t) - \hat{V}_{w_0}(w_t^\star) \le \frac{\varphi}{\alpha}$, and $\hat{V}_{z'}(w_t) - \hat{V}_{z'}(w_t^\star) \le \frac{\varphi}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}$;
Return: $\tilde{w} \approx \frac{1}{T}\sum_{t=1}^T w_t$ satisfying $\left\|\tilde{w} - \frac{1}{T}\sum_{t=1}^T w_t\right\| \le \frac{\varphi}{LD}$, $\max_u \left[\hat{V}_{\tilde{w}}(u) - \hat{V}_{\tilde{w}}(u)\right] \le \frac{\varphi}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}$, $\hat{V}_{w'}(\tilde{w}) - \hat{V}_{w'}(\bar{w}) \le \frac{\varphi}{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}$, and $\|w_t - w_t^\star\| \le \frac{\varphi}{2LD}$

Corollary 58. *Let $(\mathcal{Z}, \|\cdot\|, r, \Theta, \operatorname{clip})$ be any local norm setup. Let $w_0 \in \mathcal{Z}$, $\varepsilon_{\text{inner}} > 0$, and $\tilde{g}_{w_0}$ be an L-centered-local estimator for some $L \ge \alpha \ge \varepsilon_{\text{inner}}$. Assume the problem has bounded domain size $\max_{z\in\mathcal{Z}} \|z\| \le D$, g is L-Lipschitz, i.e. $\|g(z) - g(z')\|_* \le L\|z - z'\|$, that g is LD-bounded, i.e. $\max_{z\in\mathcal{Z}} \|g(z)\|_* \le LD$, and $\hat{w}_0 = w_0$. Then, for $\eta = \frac{\alpha}{10L^2}$, $T \ge \frac{8}{\eta\alpha} \ge \frac{60L^2}{\alpha^2}$, $\varphi = \frac{\varepsilon_{\text{inner}}}{10}$, Algorithm 68 outputs a point $\hat{w} \in \mathcal{Z}$ such that*

$$\mathbb{E} \max_{u\in\mathcal{Z}} [\langle g(\tilde{w}) + \nabla\Upsilon(\tilde{w}), \tilde{w} - u\rangle - \alpha V_{w_0}(u)] \le \varepsilon_{\text{inner}}, \tag{B.43}$$

i.e. Algorithm 68 is an $(\alpha, \varepsilon_{\text{inner}})$-relaxed proximal oracle.

Proof sketch. Note that the only change is in the definition of the regularized mirror descent step with extra composite terms

$$w_t^\star \leftarrow \operatorname{argmin}\left\{\langle \operatorname{clip}(\eta \tilde{g}_{w_0}(\hat{w}_{t-1}) - \eta g(w_0)), w\rangle + \eta\Upsilon(w) + \frac{\eta\alpha}{2}\hat{V}_{w_0}(w) + \hat{V}_{w_{t-1}}(w)\right\}.$$

Denote $\nabla\Upsilon(w) = (\mu^{\mathsf{x}}\nabla\phi(w^{\mathsf{x}}), \mu^{\mathsf{y}}\nabla\psi(w^{\mathsf{y}}))$, so that for the final regret bound there are two additional error terms. The first term comes from the error in regularized mirror descent steps via (denoting

$z' = (x', y')$)

$$\begin{aligned}&\frac{1}{T}\sum_{t\in[T]}\left[-\langle\nabla\Upsilon(w_t^\star), w_t^\star - u\rangle + \langle\nabla\Upsilon(w_t), w_t - u\rangle\right]\\ \le&\frac{\sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}}{T}\sum_{t\in[T]}\left(\hat{V}_{z'}(w_t) - \hat{V}_{z'}(w_t^\star) + \hat{V}_{w_t}(u) - \hat{V}_{w_t^\star}(u)\right) \le 2\varphi\end{aligned}$$

following the approximation guarantee in Line 7. The other term comes from averaging error. Denote the true average iterate by $\bar{w} := \frac{1}{T}\sum_{t\in[T]} w_t$. We have $\forall u \in \mathcal{Z}$,

$$\begin{aligned}\langle g(\tilde{w}), \tilde{w} - u\rangle - \frac{1}{T}\sum_{t\in[T]}\langle g(w_t), w_t - u\rangle =& -\langle g(\tilde{w}), u\rangle - \langle g(\bar{w}), \bar{w} - u\rangle\\ =& \langle g(\bar{w}) - g(\tilde{w}), u\rangle \le \varphi,\end{aligned}$$

and also

$$\begin{aligned}\langle\nabla\Upsilon(\tilde{w}), \tilde{w} - u\rangle &= \langle\nabla\Upsilon(\tilde{w}) - \nabla\Upsilon(\bar{w}), \tilde{w} - u\rangle + \langle\nabla\Upsilon(\bar{w}), \tilde{w} - \bar{w}\rangle + \langle\nabla\Upsilon(\bar{w}), \bar{w} - u\rangle\\ &\overset{(i)}{=} \sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}\left(-\hat{V}_{\bar{w}}(u) + \hat{V}_{\tilde{w}}(u) + \hat{V}_{\bar{w}}(\tilde{w})\right) + \langle\nabla\Upsilon(\bar{w}), \tilde{w} - \bar{w}\rangle + \langle\nabla\Upsilon(\bar{w}), \bar{w} - u\rangle\\ &\overset{(ii)}{=} \sqrt{\mu^{\mathsf{x}}\mu^{\mathsf{y}}}\left(-\hat{V}_{\bar{w}}(u) + \hat{V}_{\tilde{w}}(u) + \hat{V}_{w'}(\tilde{w}) - \hat{V}_{w'}(\bar{w})\right) + \langle\nabla\Upsilon(\bar{w}), \bar{w} - u\rangle,\\ &\overset{(iii)}{\le} 2\varphi + \langle\nabla\Upsilon(\bar{w}), \bar{w} - u\rangle\\ &\overset{(iv)}{\le} 2\varphi + \frac{1}{T}\sum_{t\in[T]}\langle\nabla\Upsilon(w_t), w_t - u\rangle.\end{aligned}$$

where we use (i) the three-point property of Bregman divergence, (ii) the fact that $\hat{V}_{\bar{w}}(\tilde{w}) + \langle\nabla\Upsilon(\bar{w}), \tilde{w} - \bar{w}\rangle = \hat{V}_{w'}(\tilde{w}) - \hat{V}_{w'}(\bar{w})$ again by the three-point property, (iii) the approximation guarantee of Line 8, and (iv) the fact that $\langle\nabla\Upsilon(w), w - u\rangle$ is convex in w for our choices of Υ. Hence incorporating the above extra error terms into the regret bound yields the conclusion, as $10\varphi = \varepsilon_{\text{inner}}$ by our choice of φ. □

Changes to implementation

Broadly speaking, all of these modifications can easily be handled via appropriate changes to the initial data given to our data structures `CenteredIterateMaintainer`$_2$ and `ApproxExpMaintainer`. We discuss general formulations of iterations with these modifications in both simplices and Euclidean balls, and provide appropriate modifications to the inital data given to our data structures. Finally, it is simple to check that all relevant parameters are still bounded by a polynomial in the dimensions of variables, so no additional cost due to the data structure is incurred. For simplicity here we only considerfor the x-block when $\phi^{\mathsf{x}}(x) = \mu r(x)$ and remark that the case when $\phi^{\mathsf{x}}(x) = \mu V_{x'}(x)$

for some x' follows similarly.

ℓ_1 domains. For this section, define a domain $\mathcal{X} = \Delta^n$, let $r(x) = \sum_{j\in[n]} x_j \log x_j$ be entropy, and let μ, α, η, ρ be nonnegative scalar parameters. Consider a sequence of iterates of the form

$$x_{t+1} \leftarrow \operatorname{argmin}_{x\in\mathcal{X}} \langle \tilde{g}_{x_0}(x_t), x\rangle + \mu r(x) + \frac{\alpha\rho}{2} V_{x_0}(x) + \frac{\rho}{\eta} V_{x_t}(x).$$

This update sequence, for the form of gradient estimator

$$\tilde{g}_{x_0}(x) = g(x_0) + b + g'(x),$$

where $g'(x)$ is a vector with suitable sparsity assumptions depending on the point x, and b is some fixed vector, generalizes all of the settings described above used in our various relaxed proximal oracle implementations. Optimality conditions imply that the update may be rewritten as

$$x_{t+1} \leftarrow \Pi_\Delta \left(\exp\left(\frac{\frac{\rho}{\eta}\log x_t + \frac{\alpha\rho}{2}\log x_0 - g(x_0) - b - g'(x_t)}{\mu + \frac{\alpha\rho}{2} + \frac{\rho}{\eta}} \right) \right).$$

Thus, initializing an `ApproxExpMaintainer` instance with

$$\kappa = \frac{1}{\frac{\mu\eta}{\rho} + \frac{\alpha\eta}{2} + 1},\ v = \frac{\frac{\alpha\rho}{2}\log x_0 - g(x_0) - b}{\mu + \frac{\alpha\rho}{2} + \frac{\rho}{\eta}}$$

enables `DenseStep` to propagate the necessary changes to the iterate; we propagate changes due to $g'(x_t)$ via `AddSparse` and the appropriate sparsity assumptions.

ℓ_2 domains. For this section, define a domain $\mathcal{X} = \mathbb{B}^n$, let $r(x) = \frac{1}{2}\|x\|_2^2$ be entropy, and let μ, α, η, ρ be nonnegative scalar parameters. Consider a sequence of iterates of the form

$$x_{t+1} \leftarrow \operatorname{argmin}_{x\in\mathcal{X}} \langle \tilde{g}_{x_0}(x_t), x\rangle + \mu r(x) + \frac{\alpha\rho}{2} V_{x_0}(x) + \frac{\rho}{\eta} V_{x_t}(x).$$

This update sequence, for the form of gradient estimator

$$\tilde{g}_{x_0}(x) = g(x_0) + b + g'(x),$$

where $g'(x)$ is a vector with suitable sparsity assumptions depending on the point x, and b is some fixed vector, generalizes all of the settings described above used in our various relaxed proximal oracle implementations. Optimality conditions imply that the update may be rewritten as

$$x_{t+1} \leftarrow \Pi_{\mathbb{B}^n} \left(\frac{\frac{\rho}{\eta}x_t + \frac{\alpha\rho}{2}x_0 - g(x_0) - b - g'(x_t)}{\mu + \frac{\alpha\rho}{2} + \frac{\rho}{\eta}} \right).$$

Thus, initializing an `CenteredIterateMaintainer` instance with

$$v = \frac{\frac{\alpha\rho}{2}x_0 - g(x_0) - b}{\mu + \frac{\alpha\rho}{2} + \frac{\rho}{\eta}}$$

enables `AddDense`, `Scale`, and `GetNorm` to propagate the necessary changes to the iterate; we propagate changes due to $g'(x_t)$ via `AddSparse` and the appropriate sparsity assumptions.

B.6 Deferred proofs from Section 3.6

B.6.1 Proofs from Section 3.6.1

Proof of Lemma 39. We consider the following (μ, μ)-strongly monotone problem, for various levels of μ:

$$\max_{x\in\mathbb{B}^n}\min_{y\in\Delta_m} f_\mu(x,y) := y^\top \tilde{A}x + y^\top b + \mu\sum_{i\in[m]}[y]_i\log[y]_i - \frac{\mu}{2}\|x\|_2^2.$$

We claim we can implement an (α, ε)-relaxed proximal oracle for this problem in time

$$\widetilde{O}\left(\frac{\left(L_{\mathsf{co}}^{2,1}\right)^2}{\alpha^2}\right).$$

The oracle is a composite implementation of Algorithm 12 as in Algorithm 68, using the estimator of Appendix B.4.3. By an application of Proposition 19, the overall complexity of solving this problem is (by choosing the optimal α, and overloading the constant $L_{\mathsf{co}}^{2,1}$ to be with respect to $\tilde{A}$):

$$\widetilde{O}\left(\left(\mathsf{nnz} + \frac{\left(L_{\mathsf{co}}^{2,1}\right)^2}{\alpha^2}\right)\frac{\alpha}{\mu}\right) = \widetilde{O}\left(\mathsf{nnz} + \frac{\sqrt{\mathsf{nnz}}\cdot L_{\mathsf{co}}^{2,1}}{\mu}\right).$$

By conducting a line search over the parameter μ via repeatedly halving, the total cost of solving each of these problems is dominated by the last setting, wherein $\mu = \Theta(r^*/\log m)$, and $R/\mu = \widetilde{O}(\rho)$; here, we recall that we rescaled $\tilde{A}$ so that $L_{\mathsf{co}}^{2,1} = O(R)$. We defer details of the line search procedure to Lemma C.3 of [19]. □

Proof of Theorem 12. We solve the problem (3.61) to duality gap $\epsilon\hat{r}/8 \le \epsilon r^*$, using the algorithm of Appendix B.4.3 for ℓ_2-ℓ_1 games. The complexity of this algorithm is (choosing α optimally)

$$\widetilde{O}\left(\left(\mathsf{nnz}(\tilde{A}) + \frac{\left(L_{\mathsf{co}}^{2,1}\right)^2}{\alpha^2}\right)\cdot\frac{\alpha}{\epsilon\hat{r}}\right) = \widetilde{O}\left(\mathsf{nnz} + \frac{\rho\sqrt{\mathsf{nnz}}\cdot L_{\mathsf{co}}^{2,1}}{\epsilon}\right),$$

as claimed. Here, we used that $\tilde{A}$ is a rescaling of A by $2R$, and $\hat{r}$ is a constant multiplicative approximation of r. The approximate solution $(x^*_{\epsilon'}, y^*_{\epsilon'})$ obtains the requisite duality gap in expectation;

Markov's inequality implies that with logarithmic overhead in the runtime, we can obtain a pair of points satisfying with high probability

$$\max_x f(x, y^*_{\epsilon'}) - \min_y f(x^*_{\epsilon'}, y) = \max_x f(x, y^*_{\epsilon'}) - f(x^*, y^*) + f(x^*, y^*) - \min_y f(x^*_{\epsilon'}, y) \leq \epsilon'.$$

Because y^* is the best response to x^*, we have $f(x^*, y^*_{\epsilon'}) \geq f(x^*, y^*)$, which implies

$$\max_x f(x, y^*_{\epsilon'}) - f(x^*, y^*) = \max_x f(x, y^*_{\epsilon'}) - f(x^*, y^*_{\epsilon'}) + f(x^*, y^*_{\epsilon'}) - f(x^*, y^*) \geq 0.$$

Combining yields $f(x^*, y^*) - \min_y f(x^*_{\epsilon'}, y) \leq \epsilon' \leq \epsilon r^*$, so since $f(x^*, y^*) = r^*$, rearranging implies $\min_y f(x^*_{\epsilon'}, y) \geq r^* - \epsilon' \geq (1-\epsilon)r^*$. Thus, $x^*_{\epsilon'}$ is an ϵ-approximate solution for Max-IB. □

B.6.2 Proofs from Section 3.6.2

Proof of Lemma 40. If (x', y') is an approximately optimal solution with duality gap $\epsilon/16$ for (3.64), by definition

$$\max_{y \in \Delta^m} f_{\epsilon'}(x', y) - \min_{x \in \mathbb{R}^n} f_{\epsilon'}(x, y') \leq \frac{\epsilon}{16}.$$

Therefore, the following sequence of inequalities hold:

$$\begin{aligned}
&\max_{y \in \Delta^m} f(x', y) - \min_{x \in \mathbb{R}^n} f(x, y') = \left(\max_{y \in \Delta^m} f(x', y) - \max_{y \in \Delta^m} f_{\epsilon'}(x', y)\right) \\
&+ \left(\max_{y \in \Delta^m} f_{\epsilon'}(x', y) - \min_{x \in \mathbb{R}^n} f_{\epsilon'}(x, y')\right) + \left(\min_{x \in \mathbb{R}^n} f_{\epsilon'}(x, y') - \min_{x \in \mathbb{R}^n} f(x, y')\right) \\
&\overset{(i)}{\leq} \left(\max_{y \in \Delta^m} f(x', y) - \max_{y \in \Delta^m} f_{\epsilon'}(x', y)\right) + \frac{\epsilon}{16} + \left(\min_{x \in \mathbb{R}^n} f_{\epsilon'}(x, y') - \min_{x \in \mathbb{R}^n} f(x, y')\right) \\
&\overset{(ii)}{\leq} \frac{\epsilon}{32} + \frac{\epsilon}{16} + \frac{\epsilon}{32} = \frac{\epsilon}{8}.
\end{aligned}$$

In (i), we used the fact that the pair (x', y') has good duality gap with respect to $f_{\epsilon'}$, and in (ii) we used that for the first summand, $f_{\epsilon'}(x', \cdot)$ approximates $f(x', \cdot)$ to an additive $\epsilon/32$, and for the third summand, $-\epsilon' \sum_{i \in [m]} [y']_i \log [y']_i$ is bounded by $\epsilon/32$, and all other terms cancel. □

B.6.3 Proofs from Section 3.6.3

Proof of Lemma 41. At optimality for (3.65), it holds that

$$\begin{cases} y^*_{x'} = \frac{1}{\beta}(Ax^*_{x'} - b) \\ x^*_{x'} = x' - \frac{1}{\beta} A^\top y^*_{x'} \end{cases}.$$

By substituting $y^*_{x'}$ and rearranging terms we get

$$\left(I + \frac{1}{\beta^2} A^\top A\right)(x^*_{x'} - x^*) = x' - x^*,$$

which in turn gives

$$\|x^*_{x'} - x^*\|_2 = \left\| \left(I + \frac{1}{\beta^2} A^\top A\right)^{-1} (x' - x^*) \right\|_2 \leq \frac{1}{1 + \mu/\beta^2} \|x' - x^*\|_2 .$$

For the last inequality we use the fact that

$$\left\| I + \frac{1}{\beta^2} A^\top A \right\|_2^{-1} = \lambda_{\min}\left(I + \frac{1}{\beta^2} A^\top A\right)^{-1} = \frac{1}{1 + \mu/\beta^2},$$

by the definition of μ and since I and $A^\top A$ commute. □

Theorem 14. *Given data matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$, and desired accuracy $\epsilon \in (0, 1)$, assuming $A^\top A \succeq \mu I$ for $\mu > 0$, Algorithm 69 outputs an expected ϵ-accurate solution $\tilde{x}$, i.e.*

$$\mathbb{E}\left[\|\tilde{x} - x^*\|_2\right] \leq \epsilon,$$

and runs in time

$$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \frac{\max\left\{\sqrt{\sum_i \|A_{i:}\|_1^2}, \sqrt{\sum_j \|A_{:j}\|_1^2}\right\}}{\sqrt{\mu}}\right).$$

Proof. We first prove correctness. We bound the progress from $x^{(h)}$ to $x^{(h+1)}$, for some $h \in [H]$, by

$$\frac{1}{2}\left\|x^{(h+1)} - x^*\right\|_2^2 \leq \left\|x^{(h+1)} - x^*_{x^{(h)}}\right\|_2^2 + \left\|x^*_{x^{(h)}} - x^*\right\|_2^2 \leq 2V_{z^{(h+1)}}(z^*_{x^{(h)}}) + \left\|x^*_{x^{(h)}} - x^*\right\|_2^2 . \quad \text{(B.44)}$$

The first inequality used $\|a + b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$, and the second used the definition of the divergence in the ℓ_2-ℓ_2 setup. Next, choosing a sufficiently large value of $K = \widetilde{O}(\beta/\mu)$, we use Proposition 19 to obtain a point $z^{(h+1)}$ satisfying

$$V_{z^{(h+1)}}(z^*_{x^{(h)}}) \leq \frac{\epsilon^2}{80} V_{z^{(h)}}(z^*_{x^{(h)}}) \leq \frac{\epsilon^2}{40} V_{z^{(h)}}(z^*) + \frac{\epsilon^2}{40} V_{z^*}(z^*_{x^{(h)}}). \quad \text{(B.45)}$$

Further, using Lemma 41 with $x' = x^{(h)}, \beta = \sqrt{\mu}$ yields

$$\left\|x^*_{x^{(h)}} - x^*\right\|_2 \leq \frac{1}{2}\left\|x^{(h)} - x^*\right\|_2 . \quad \text{(B.46)}$$

Algorithm 69: Coordinate variance reduced method for linear regression

Input: Matrix $A \in \mathbb{R}^{m \times n}$ with ith row $A_{i:}$ and jth column $A_{:j}$, vector $b \in \mathbb{R}^m$, accuracy ϵ
Output: A point $\tilde{x}$ with $\|\tilde{x} - x^*\|_2 \leq \epsilon$

$L \leftarrow \max\left\{\sqrt{\sum_i \|A_{i:}\|_1^2}, \sqrt{\sum_j \|A_{:j}\|_1^2}\right\}$, $\alpha \leftarrow L/\sqrt{\mathsf{nnz}}$, $\beta = \sqrt{\mu}$, $\eta \leftarrow \frac{\alpha}{4L^2}$;

$T \leftarrow \left\lceil \frac{4}{\eta\alpha} \right\rceil$, $K \leftarrow \widetilde{O}(\alpha/\beta)$, $H = \widetilde{O}(1)$, $z^{(0)} = (x^{(0)}, y^{(0)}) \leftarrow (\mathbf{0}_n, \mathbf{0}_m)$, $(z_0^{\mathsf{x}}, z_0^{\mathsf{y}}) \leftarrow (\mathbf{0}_n, \mathbf{0}_m)$;

for $h = 1, 2, \cdots, H$ **do**

  for $k = 1, \ldots, K$ **do**

    ▷ *Relaxed oracle query:*

    $(x_0, y_0) \leftarrow (z_{k-1}^{\mathsf{x}}, z_{k-1}^{\mathsf{y}})$, $(g_0^{\mathsf{x}}, g_0^{\mathsf{y}}) \leftarrow (A^\top y_0 + \beta(x_0 - x^{(h-1)}), -Ax_0 + \beta y_0)$;

    for $t = 1, \ldots, T$ **do**

      ▷ *Gradient estimation:*

      Sample $i \sim p$ where $p_i = \dfrac{([y_{t-1}]_i - [y_0]_i)^2}{\|y_{t-1} - y_0\|_2^2}$;

      Sample $j \sim q$ where $q_j = \dfrac{([x_{t-1}]_j - [x_0]_j)^2}{\|x_{t-1} - x_0\|_2^2}$;

      Set $\tilde{g}_{t-1} = g_0 + \left(A_{i:}\dfrac{[y_{t-1}]_i - [y_0]_i}{p_i}, -A_{:j}\dfrac{[x_{t-1}]_j - [x_0]_j}{q_j}\right)$;

      ▷ *Mirror descent step:*

      $x_t \leftarrow \dfrac{1}{1 + \eta\alpha/2}\left(x_{t-1} + \dfrac{\eta\alpha}{2}x_0 - \eta\tilde{g}_{t-1}^{\mathsf{x}}\right)$

      $y_t \leftarrow \Pi_{\mathcal{Y}}\left(\dfrac{1}{1 + \eta\alpha/2}\left(y_{t-1} + \dfrac{\eta\alpha}{2}y_0 - \eta\tilde{g}_{t-1}^{\mathsf{y}}\right)\right)$ ▷ $\Pi_{\mathcal{Y}}(v) = \frac{v}{\max\{1, \|v\|_2\}}$;

    $z_{k-1/2} \leftarrow \dfrac{1}{T}\sum_{t=1}^{T}(x_t, y_t)$;

    ▷ *Extragradient step:*

    $z_k^{\mathsf{x}} \leftarrow \dfrac{\alpha}{\alpha + 2\beta}z_{k-1}^{\mathsf{x}} + \dfrac{2\beta}{\alpha + 2\beta}z_{k-1/2}^{\mathsf{x}} - \dfrac{1}{\alpha + 2\beta}\left(A^\top z_{k-1/2}^{\mathsf{y}} + \beta(z_{k-1/2}^{\mathsf{x}} - x^{(h-1)})\right)$;

    $z_k^{\mathsf{y}} \leftarrow \Pi_{\mathcal{Y}}\left(\dfrac{\alpha}{\alpha + 2\beta}z_{k-1}^{\mathsf{y}} + \dfrac{2\beta}{\alpha + 2\beta}z_{k-1/2}^{\mathsf{y}} + \dfrac{1}{\alpha + 2\beta}\left(Az_{k-1/2}^{\mathsf{x}} - \beta z_{k-1/2}^{\mathsf{y}}\right)\right)$;

  ▷ *Reshifting the oracle:*

  $z^{(h)} = (x^{(h)}, y^{(h)}) \leftarrow z_K = (z_K^{\mathsf{x}}, z_K^{\mathsf{y}})$;

Return: $\tilde{x} \leftarrow x^{(H)}$

Plugging these two bounds into (B.44), and using the form of the divergence in the ℓ_2-ℓ_2 setup,

$$\begin{aligned}\frac{1}{2}\left\|x^{(h+1)}-x^*\right\|_2^2 &\overset{(B.45)}{\leq} \frac{\epsilon^2}{20}V_{z^{(h)}}(z^*)+\frac{\epsilon^2}{20}V_{z^*}(z^*_{x^{(h)}})+\left\|x^*_{x^{(h)}}-x^*\right\|_2^2 \\ &\overset{(B.46)}{\leq} \frac{1}{2}\left(\frac{\epsilon^2}{20}+\frac{\epsilon^2}{20}+\frac{1}{2}\right)\left\|x^{(h)}-x^*\right\|_2^2+\frac{\epsilon^2}{40}\left(\left\|y^{(h)}-y^*\right\|_2^2+\left\|y^*_{x^{(h)}}-y^*\right\|_2^2\right) \\ &\leq \frac{3}{4}\cdot\frac{1}{2}\left\|x^{(h)}-x^*\right\|_2^2+\frac{\epsilon^2}{5}.\end{aligned} \tag{B.47}$$

In the last inequality we use the conditions that $\epsilon \in (0,1)$ and $\mathcal{Y} = \mathbb{B}^m$. Recursively applying this bound for $h \in [H]$, and for a sufficiently large value of $H = \widetilde{O}(1)$, we have the desired

$$\left\|x^{(H)}-x^*\right\|_2^2 \leq \left(\frac{3}{4}\right)^H\left\|x^{(0)}-x^*\right\|_2^2+\frac{4\epsilon^2}{5} \leq \epsilon^2.$$

To bound the runtime, recall the inner loop runs for $T = O((L^{2,2}_{\mathsf{co}})^2/\alpha^2)$ iterations, each costing constant time, and the outer loop runs for $K = \widetilde{O}(\alpha/\beta)$ iterations, each costing $O(T+\mathsf{nnz})$. Finally, since $H = \widetilde{O}(1)$, the overall complexity of the algorithm is

$$\widetilde{O}\left(\left(\mathsf{nnz}+\frac{\left(L^{2,2}_{\mathsf{co}}\right)^2}{\alpha^2}\right)\frac{\alpha}{\beta}\right).$$

Choosing $\alpha = \max\{L^{2,2}_{\mathsf{co}}/\sqrt{\mathsf{nnz}}, \beta\}$ optimally and substituting

$$\beta = \sqrt{\mu},\ L^{2,2}_{\mathsf{co}} = \max\left\{\sqrt{\sum_i \|A_{i:}\|_1^2}, \sqrt{\sum_j \|A_{:j}\|_1^2}\right\},$$

we have the desired runtime bound on Algorithm 69. □

B.7 Strongly monotone proximal method

We provide a proof of a generalization of Proposition 19, namely Proposition 56. It is straightforward to deduce Proposition 19 from Proposition 56; by the definition of our distance-generating function r in Proposition 19, g is μ-strongly monotone in r, and the range of the regularizer is affected by a $\rho+\rho^{-1}$ factor. Finally, we mention that we proved a similar convergence rate in Appendix A.4, but it used a slightly different algorithm (namely, it generated the intermediate step using a first-order extrapolation step rather than a relaxed proximal oracle). For our variance reduction results, we require the specialization of strongly monotone extragradient methods to proximal oracles developed in this section.

As a brief refresher, in this section we consider variational inequalities with *strongly monotone*

operators. Following [375], we say that an operator g is μ-strongly monotone relative to the distance generating function r if it satisfies

$$\langle g(z') - g(z), z' - z \rangle \geq \mu \cdot V_{z'}(z) \quad \text{for all} \quad z', z \in \mathcal{Z}. \tag{B.48}$$

By strong convexity of r, a μ-strongly monotone operator relative to r is also μ-strongly monotone in the standard sense, i.e., $\langle g(z') - g(z), z' - z \rangle \geq \frac{\mu}{2} \|z' - z\|^2$ for all $z, z' \in \mathcal{Z}$. For saddle point problems $\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y)$ with gradient mapping $g(x, y) = (\nabla_x f(x, y), -\nabla_y f(x, y))$ and separable $r(x, y) = r^{\mathsf{x}}(x) + r^{\mathsf{y}}(y)$, the property (B.48) holds whenever $f(x, y) - \mu r^{\mathsf{x}}(x) + \mu r^{\mathsf{y}}(y)$ is convex-concave, i.e., whenever $f(\cdot, y)$ is μ-strongly convex relative to r^{x} for every y and $f(x, \cdot)$ is μ-strongly-concave relative to r^{y} for every x.

Strongly monotone operators have a unique saddle point $z^\star$ satisfying

$$\max_{u \in \mathcal{Z}} \langle g(z^\star), z^\star - u \rangle \leq 0. \tag{B.49}$$

We show that Algorithm 70—a small modification of Algorithm 11—converges linearly to $z^\star$ (i.e., with the distance to $z^\star$ decaying exponentially in the number of iterations). The main difference between the algorithms is that the extra-gradient step in Algorithm 70 has an additional regularization term around $z_{k-1/2}$, making it similar in form to the regularized mirror descent steps we use in Algorithm 12. In addition, Algorithm 70 outputs the last iterate rather the iterate average.

Algorithm 70: `OuterLoopStronglyMonotone`$(\mathcal{O})$

Input: (α, ε)-relaxed proximal oracle $\mathcal{O}(z)$ for gradient mapping g satisfying (B.48);
Parameters: Number of iterations K;
Output: Point z_K with $\mathbb{E} V_{z_K}(z^\star) \leq (\frac{\alpha}{\mu + \alpha})^K \Theta + \frac{\varepsilon}{\mu}$;
$z_0 \leftarrow \operatorname{argmin}_{z \in \mathcal{Z}} r(z)$;
for $k = 1, \ldots, K$ **do**
  $z_{k-1/2} \leftarrow \mathcal{O}(z_{k-1})$ ▷ We implement $\mathcal{O}(z_{k-1})$ by calling `InnerLoop`$(z_{k-1}, \tilde{g}_{z_{k-1}}, \alpha)$;
  $z_k \leftarrow \operatorname{argmin}_{z \in \mathcal{Z}} \left\{ \left\langle g\left(z_{k-1/2}\right), z \right\rangle + \alpha V_{z_{k-1}}(z) + \mu V_{z_{k-1/2}}(z) \right\}$;
Return: z_K;

Proposition 56. *Let $\mathcal{O}$ be an (α,ε)-relaxed proximal oracle for a gradient mapping g that is μ-strongly monotone relative to distance-generating function r with range at most Θ. Let z_K be the output of Algorithm 70. Then*

$$\mathbb{E}\, V_{z_K}(z^\star) \leq \left(\frac{\alpha}{\mu + \alpha} \right)^K \Theta + \frac{\varepsilon}{\mu}.$$

If in addition g is a gradient mapping for f and $\|(\nabla_x f(z), -\nabla_y f(z'))\|_* \le G$ for all $z, z' \in \mathcal{Z}$, then

$$\mathbb{E}\,\mathrm{Gap}(z_K) \le \sqrt{2}G\sqrt{\left(\frac{\alpha}{\mu+\alpha}\right)^K \Theta + \frac{\varepsilon}{\mu}}.$$

Proof. Fix an iteration k. Using μ-strong-monotonicity (B.48) and optimality of $z^\star$ (B.49) yields

$$\begin{aligned}\mu V_{z_{k-1/2}}(z^\star) &\overset{\text{(B.48)}}{\le} \left\langle g(z_{k-1/2}) - g(z^\star), z_{k-1/2} - z^\star \right\rangle \overset{\text{(B.49)}}{\le} \left\langle g(z_{k-1/2}), z_{k-1/2} - z^\star \right\rangle \\ &= \left\langle g(z_{k-1/2}), z_k - z^\star \right\rangle + \left\langle g(z_{k-1/2}), z_{k-1/2} - z_k \right\rangle. \end{aligned} \tag{B.50}$$

Next,

$$\begin{aligned}\left\langle g(z_{k-1/2}), z_k - z^\star \right\rangle &\overset{(i)}{\le} -\langle \alpha \nabla V_{z_{k-1}}(z_k) + \mu \nabla V_{z_{k-1/2}}(z_k), z_k - z^\star \rangle \\ &\overset{(ii)}{\le} \mu(V_{z_{k-1/2}}(z^\star) - V_{z_k}(z^\star) - V_{z_{k-1/2}}(z_k)) + \alpha(V_{z_{k-1}}(z^\star) - V_{z_k}(z^\star) - V_{z_{k-1}}(z_k)) \\ &\le \alpha V_{z_{k-1}}(z^\star) - (\mu+\alpha) V_{z_k}(z^\star) + \mu V_{z_{k-1/2}}(z^\star) - \alpha V_{z_{k-1}}(z_k), \end{aligned} \tag{B.51}$$

where we use (i) the first-order optimality condition for z_k and (ii) the three-point property of Bregman divergence. In the last inequality, we also use nonnegativity of $V_{z_{k-1/2}}(z_k)$. Now, combining (E.8) and (E.10) yields

$$\mu V_{z_{k-1/2}}(z^\star) \le \alpha V_{z_{k-1}}(z^\star) - (\mu+\alpha) V_{z_k}(z^\star) + \mu V_{z_{k-1/2}}(z^\star) + [\left\langle g(z_{k-1/2}), z_{k-1/2} - z_k \right\rangle - \alpha V_{z_{k-1}}(z_k)].$$

Rearranging terms, taking an expectation, and using the Definition 6 of an (α, ε)-relaxed proximal oracle yields

$$\mathbb{E} V_{z_k}(z^\star) \le \frac{\alpha}{\mu+\alpha} \mathbb{E} V_{z_{k-1}}(z^\star) + \frac{\varepsilon}{\mu+\alpha}.$$

Applying this bound recursively K times and using that $V_{z_0}(u) = r(u) - r(z_0) \le \Theta$ for z_0 the minimizer of r, we have

$$\mathbb{E} V_{z_K}(z^\star) \le \left(\frac{\alpha}{\mu+\alpha}\right)^K \Theta + \sum_{k=0}^{K-1} \left(\frac{\alpha}{\mu+\alpha}\right)^k \left(\frac{\varepsilon}{\mu+\alpha}\right) \le \left(\frac{\alpha}{\mu+\alpha}\right)^K \Theta + \frac{\varepsilon}{\mu}.$$

To bound $\mathrm{Gap}(z_K)$ (defined in (3.24)), write $\mathrm{gap}(z; u) = f(z^{\mathsf{x}}, u^{\mathsf{y}}) - f(u^{\mathsf{x}}, z^{\mathsf{y}})$. Then we have $\|\nabla_z \mathrm{gap}(z; u)\|_* = \|(\nabla_x f(z^{\mathsf{x}}, u^{\mathsf{y}}), -\nabla_y f(u^{\mathsf{y}}, z^{\mathsf{x}}))\|_* \le G$ by assumption (since $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$), and therefore $\mathrm{gap}(z; u)$ is G-Lipschitz in z. Consequently, for any $u \in \mathcal{Z}$,

$$\mathrm{gap}(z_K; u) \le \mathrm{gap}(z^\star; u) + G\,\|z_K - z^\star\| \le G\,\|z_K - z^\star\|,$$

where the second transition follows from the optimality of $z^\star$. Therefore, the definition (3.24) of

Gap, strong convexity of r and Jensen's inequality yield

$$\mathbb{E}\mathrm{Gap}(z_K) = \mathbb{E}\max_{u\in\mathcal{Z}}\mathrm{gap}(z_K;u) \le G\,\mathbb{E}\,\|z_K - z^\star\| \le \mathbb{E}G\sqrt{2V_{z_K}(z^\star)} \le \sqrt{2}G\sqrt{\mathbb{E}V_{z_K}(z^\star)}.$$

Substituting the bound on $\mathbb{E}V_{z_K}(z^\star)$ concludes the proof. □

B.8 IterateMaintainer$_2$: numerical stability and variations

B.8.1 Numerical stability of IterateMaintainer$_1$.

We discuss the implementation of a numerically stable version of IterateMaintainer$_1$, and the complexity of its operations, for use in our sublinear algorithms in Section 3.4.1 and Section B.3.2. We discuss this implementation for a simplex block, e.g. a simplex variable of dimension n, as for an ℓ_2 geometry numerical stability is clear. The main modifications we make are as follow.

- We reinitialize the data structure whenever the field ν grows larger than some fixed polynomial in n, or if $n/2$ iterations have passed.
- We track the coordinates modified between restarts.
- Every time we reinitialize, we maintain the invariant that the multiplicative range of coordinates of x is bounded by a polynomial in n, i.e. $\max_j x_j / \min_j x_j$ is bounded by some fixed polynomial in n. We will implement this via an explicit truncation, and argue that such an operation gives negligible additive error compared to the accuracy of the algorithm.
- We implicitly track the set of truncated coordinates at each data structure restart. We do so by explicitly tracking the set of non-truncated coordinates whenever a truncation operation happens (see the discussion below), in constant amortized time.

We now discuss the complexity and implementation of these restarts. First, note that ν can never decrease by more than a multiplicative polynomial in n between restarts, because of nonnegativity of the exponential, the fact that the original range at the time of the last restart is multiplicatively bounded, and we restart every time half the coordinates have been touched. Thus, the only source of numerical instability comes from when ν grows by more than a multiplicative polynomial in n. Suppose this happens in τ iterations after the restart. Then,

- If $\tau < n/2$, we claim we can implement the restart in $O(\tau)$, so the amortized cost per iteration is $O(1)$. To see this, for every coordinate touched in these τ iterations, we either keep or explicitly truncate if the coordinate is too small. For every coordinate not touched in these τ iterations, the relative contribution is at most inverse polynomial in n; we truncate all such coordinates. Then, we compute the normalization constant according to all non-truncated coordinates, such

that the value of all truncated coordinates is set to a fixed inverse polynomial in n. We can implement this by implicitly keeping track of the set of truncated coordinates as well as their contribution to the normalization factor, and explicitly setting their value in the data structure when they are updated by `AddSparse`. Overall, this does not affect the value of the problem by more than a small multiple of ϵ, by our assumptions on L_{rc}/ϵ. To see that we can track the non-truncated coordinates explicitly, we note that it is a subset of the at most τ coordinates that were touched, so this can be done in constant amortized time.

- If $\tau = n/2$, we claim we can implement the restart in $O(n)$, so the amortized cost per iteration is $O(1)$. This is clear: we can do so by explicitly recomputing all coordinates, and truncating any coordinates which have become too small.

We describe how the data structure implements this through its maintained fields: for non-truncated coordinates, we do not do anything other than change the scaling factor ν, and for truncated coordinates, we reset the values of u, u' in that coordinate appropriately once they have been sparsely updated. Overall, this does not affect the amortized runtime of our algorithm.

B.8.2 $\texttt{WeightedIterateMaintainer}_2$

In this section, we give implementation details for a weighted generalization of $\texttt{IterateMaintainer}_2$, which we will call $\texttt{WeightedIterateMaintainer}_2$. It is used in Section B.3.2, when using the sampling distribution (B.26). At initialization, $\texttt{WeightedIterateMaintainer}_2$ is passed an additional parameter $w \in \mathbb{R}^n_{\geq 0}$, a nonnegative weight vector. We let

$$\langle u, v \rangle_w := \sum_{j \in [n]} [w]_j [u]_j [v]_j, \|v\|_w := \sqrt{\langle v, v \rangle_w}.$$

$\texttt{WeightedIterateMaintainer}_2$ supports all the same operations as $\texttt{IterateMaintainer}_2$, with two differences:

- For the current iterate x, $\texttt{WeightedIterateMaintainer}_2.\texttt{Norm}()$ returns weighted norm $\|x\|_w$.
- For the current iterate x, $\texttt{WeightedIterateMaintainer}_2.\texttt{Sample}()$ returns a coordinate j with probability proportional to $[w]_j [x]_j^2$.

Similarly to $\texttt{IterateMaintainer}_2$, $\texttt{WeightedIterateMaintainer}_2$ maintains the following fields.

- Scalars ξ_u, ξ_v, σ_u, σ_v, ι, ν
- Vectors u, u', v, w
- Precomputed value $\|v\|_w^2$.

We maintain the following invariants on the data structure fields at the end of every operation:

- $x = \xi_u u + \xi_v v$, the internal representation of x
- $s = v + \sigma_u u + \sigma_v v$, the internal representation of running sum s
- $\iota = \langle x, v \rangle_w$, the weighted inner product of the iterate with fixed vector v
- $\nu = \|x\|_w$, the weighted norm of the iterate

To support sampling, our data structure also maintains a binary tree dist_x of depth $O(\log n)$. For the node corresponding to $S \subseteq [n]$ (where S may be a singleton), we maintain

- $\sum_{j \in S} [w]_j [u]_j^2$, $\sum_{j \in S} [w]_j [u]_j [v]_j$, $\sum_{j \in S} [w]_j [v]_j^2$

We now give the implementation of the necessary operations for $\texttt{WeightedIterateMaintainer}_2$, giving additional proofs of correctness when applicable.

Initialization.

- $\texttt{Init}(x_0, v, w)$. Runs in time $O(n)$.
  1. $(\xi_u, \xi_v, u) \leftarrow (1, 0, x_0)$.
  2. $(\sigma_u, \sigma_v, u') \leftarrow (0, 0, \mathbf{0}_n)$.
  3. $(\iota, \nu) \leftarrow (\langle x_0, v \rangle_w, \|x_0\|_w)$.
  4. Compute and store $\|v\|_w^2$.
  5. Initialize dist_x, storing the relevant sums in each internal node.

Updates.

$\texttt{Scale}(c)$ and $\texttt{UpdateSum}()$ follow identically to the analysis of $\texttt{IterateMaintainer}_2$.

- $\texttt{AddSparse}(j, c)$: $[x]_j \leftarrow [x]_j + c$. Runs in time $O(\log n)$.
  1. $u \leftarrow u + \frac{c}{\xi_u} e_j$.
  2. $u' \leftarrow u' - \frac{c\sigma_u}{\xi_u} e_j$.
  3. $\nu \leftarrow \sqrt{\nu^2 + 2c[w]_j [\xi_u u + \xi_v v]_j + c^2 [w]_j}$.
  4. $\iota \leftarrow \iota + c[w]_j [v]_j$.
  5. For internal nodes of dist_x on the path from leaf j to the root, update $\sum_{j \in S} [w]_j [u]_j^2$, $\sum_{j \in S} [w]_j [u]_j [v]_j$ appropriately.
- $\texttt{AddDense}(c)$: $x \leftarrow x + cv$. Runs in time $O(1)$.
  1. $\xi_v \leftarrow \xi_v + c$.

2. $\nu \leftarrow \sqrt{\nu^2 + 2c\iota + c^2 \|v\|_w^2}$.
3. $\iota \leftarrow \iota + c\|v\|_w^2$.

We demonstrate that the necessary invariants on ι, ν are preserved. Regarding correctness of `AddSparse`, the updates to u and u' are identical to in the analysis of $\texttt{IterateMaintainer}_2$. Next, because only $[x]_j$ changes, the updates to ν, ι are correct respectively by

$$[w]_j \cdot [\xi_u u + \xi_v v + c]_j^2 = [w]_j \cdot \left([\xi_u u + \xi_v v]_j^2 + 2c[\xi_u u + \xi_v v]_j + c^2\right),$$
$$[w]_j \cdot ([\xi_u u + \xi_v v + c]_j) \cdot [v]_j = [w]_j \cdot ([\xi_u u + \xi_v v]_j \cdot [v]_j + c[v]_j).$$

Regarding correctness of `AddDense`,

$$\|x + cv\|_w^2 = \nu^2 + 2c\iota + c^2 \|v\|_w^2,$$
$$\langle x + cv, v\rangle_w = \iota + c\|v\|_w^2.$$

Here, we used that the invariants $\nu = \|x\|_w$ and $\iota = \langle x, v\rangle_w$ held. *Queries.* $\texttt{Get}(j)$ and $\texttt{GetSum}(j)$ follow identically to the analysis of $\texttt{IterateMaintainer}_2$.

- $\texttt{Norm}()$: Return $\|x\|_w$. Runs in time $O(1)$.
    1. Return ν.

Sampling.

To support `Sample`, we must produce a coordinate j with probability proportional to $[w]_j[x]_j^2$. To do so, we recursively perform the following procedure, where the recursion depth is at most $O(\log n)$, starting at the root node and setting $S = [n]$: the proof of correctness is identical to the proof in the analysis of $\texttt{IterateMaintainer}_2.\texttt{Sample}()$.

1. Let S_1, S_2 be the subsets of coordinates corresponding to the children of the current node.
2. Using scalars ξ_u, ξ_v, and the maintained $\sum_{j\in S_i}[w]_j[u]_j^2$, $\sum_{j\in S_i}[w]_j[u]_j[v]_j$, $\sum_{j\in S_i}[w]_j[v]_j^2$, compute $\sum_{j\in S_i}[w]_j[x]_j^2 = \sum_{j\in S_i}[w]_j[\xi_u u + \xi_v v]_j^2$ for $i \in \{1, 2\}$.
3. Sample a child $i \in \{1, 2\}$ of the current node proportional to $\sum_{j\in S_i}[w]_j[x]_j^2$ by flipping an appropriately biased coin. Set $S \leftarrow S_i$.

B.8.3 $\texttt{CenteredIterateMaintainer}_2$

In this section, we give implementation details for a generalization of $\texttt{WeightedIterateMaintainer}_2$, which we call $\texttt{CenteredIterateMaintainer}_2$. It is used in Section B.4.3, when using the sampling

distributions (B.37) and (B.38). At initialization, `CenteredIterateMaintainer`$_2$ is passed an additional parameter $x_0 \in \mathbb{R}^n$, a reference point. `CenteredIterateMaintainer`$_2$ supports all the same operations as `WeightedIterateMaintainer`$_2$, with two differences:

- For the current iterate x, `CenteredIterateMaintainer`$_2$.`Sample`() returns a coordinate j with probability proportional to $[w]_j[x - x_0]_j^2$.
- `CenteredIterateMaintainer`$_2$ supports querying $\|x - x_0\|_w^2$ in constant time.

Because all the other operations, fields, and invariants supported and maintained by the data structure are exactly the same as `IterateMaintainer`$_2$, we only discuss the changes made to the binary tree dist_x in this section for brevity. In particular, to support sampling, our data structure also maintains a binary tree dist_x of depth $O(\log n)$. For the node corresponding to $S \subseteq [n]$ (where S may be a singleton), we maintain

- $\sum_{j \in S}[w]_j[u]_j^2$, $\sum_{j \in S}[w]_j[u]_j[v]_j$, $\sum_{j \in S}[w]_j[v]_j^2$
- $\sum_{j \in S}[w]_j[x_0]_j^2$, $\sum_{j \in S}[w]_j[u]_j[x_0]_j$, $\sum_{j \in S}[w]_j[v]_j[x_0]_j$

At initialization, `CenteredIterateMaintainer`$_2$ creates this data structure and stores the relevant sums in each internal node. Upon modifications to u due to updates of the form `AddSparse`(j, c), `CenteredIterateMaintainer`$_2$ propagates the changes along internal nodes of dist_x on the path from leaf j to the root. Thus, using these maintained values and the stored values ξ_u, ξ_v, it is clear that for any appropriate subset S, we are able to compute the quantity

$$\sum_{j \in S}[w]_j[\xi_u + \xi_v v - x_0]_j^2 = \sum_{j \in S}[w]_j \left(\xi_u^2[u]_j^2 + \xi_v^2[v]_j^2 + 2\xi_u\xi_v[u]_j[v]_j + [x_0]_j^2 + 2\xi_u[u]_j[x_0]_j + 2\xi_v[v]_j[x_0]_j\right)$$

in constant time, admitting the sampling oracle in time $O(\log n)$ by propagating down the tree maintained by dist_x. This proves the desired sampling complexity. Finally, by appropriately querying the stored values in the root node, we can return $\|x - x_0\|_w^2$ in constant time.

Appendix C

Deferred proofs from Chapter 4

C.1 Missing proofs from Section 4.1 and Section 4.2

C.1.1 Folklore bound on size of ℓ_∞-strongly-convex functions

In this section, we prove the following claim which occurs in the literature, but does not seem to usually be formally shown:

Lemma 238. *Suppose ψ is 1-strongly convex with respect to the ℓ_∞ norm on $[-1,1]^n$. Then,*

$$\max_{x\in[-1,1]^n} \psi(x) - \min_{x\in[-1,1]^n} \psi(x) \geq \frac{n}{2}$$

Furthermore, this lower bound is tight, i.e. there is a 1-strongly convex function in the ℓ_∞ norm for which equality holds.

Proof. We will prove this by iteratively constructing a set of points $x_0, x_1, \ldots x_n \in [-1,1]^n$ such that for all i with $0 \leq i \leq n-1$, we have

$$\psi(x_i) \leq \psi(x_{i+1}) - \frac{1}{2}$$

and consequently,

$$\psi(x_0) \leq \psi(x_n) - \frac{n}{2}$$

Let e_i be the i^{th} standard basis vector, namely the n-dimensional vector which is 1 in the i^{th} coordinate and 0 elsewhere. Let $x_0 = (0, 0, \ldots 0)$, the n-dimensional point which is 0 in every coordinate. Let $x_1^+ = x_0 + e_1$ and let $x_1^- = x_0 - e_1$, such that $x_0 = \frac{1}{2}x_1^+ + \frac{1}{2}x_1^-$. By strong convexity,

$$\psi(x_0) \leq \frac{1}{2}\psi(x_1^+) + \frac{1}{2}\psi(x_1^-) - \frac{1}{8}\left\|x_1^+ - x_1^-\right\|_\infty^2 = \frac{1}{2}\psi(x_1^+) + \frac{1}{2}\psi(x_1^-) - \frac{1}{2}$$

Consequently, it must be the case that at least one of

$$\psi(x_0) \leq \psi(x_1^+) - \frac{1}{2}$$
$$\psi(x_0) \leq \psi(x_1^-) - \frac{1}{2}$$

holds. Let x_1 be the point x_1^+ or x_1^- for which this holds.

More generally, suppose we have constructed $x_0, x_1, \ldots x_i$ in this fashion, such that x_i is 0 in the coordinates $i+1, i+2, \ldots n$. Then, let $x_{i+1}^+ = x_i + e_{i+1}$ and let $x_{i+1}^- = x_i - e_{i+1}$, such that $x_i = \frac{1}{2}x_{i+1}^+ + \frac{1}{2}x_{i+1}^-$. Again by strong convexity, we have that at least one of

$$\psi(x_i) \leq \psi(x_{i+1}^+) - \frac{1}{2}$$
$$\psi(x_i) \leq \psi(x_{i+1}^-) - \frac{1}{2}$$

holds, and therefore we can pick one of the points x_{i+1}^+, x_{i+1}^- to be the point x_{i+1}. We can clearly iteratively construct a point x_n in this fashion, proving the claim.

To show that the lower bound is tight, consider $\psi(x) = \frac{1}{2}\|x\|_2^2$. Clearly this function has range $\frac{n}{2}$ over $[-1,1]^n$. Furthermore, for all $x \in [-1,1]^n$, and arbitrary vector z, we have

$$z^\top \nabla^2 \psi(x) z = z^\top I z = \|z\|_2^2 \geq \|z\|_\infty^2$$

where this second-order condition is well-known to be equivalent to 1-strong convexity, for twice-differentiable functions. □

C.1.2 Reduction from general box-constrained ℓ_∞ regression to Definition 8

In this section, we describe a general reduction from unconstrained ℓ_∞ regression and more arbitrary box constraints to the setting where the domain of the argument is $[-1,1]^m$, proving Corollary 6. Consider first the problem of solving the generalized box-constrained regression problem

$$\min_{x \in [-r,r]^m} \|Ax - b\|_\infty, \tag{C.1}$$

for some $r > 0$. By performing the change of variables $\tilde{x} = x/r$, $\tilde{b} = b/r$, it suffices to find an ϵ/r-approximate minimizer to

$$\min_{\tilde{x} \in [-1,1]^m} \left\|A\tilde{x} - \tilde{b}\right\|_\infty, \tag{C.2}$$

which under the change of variables $x \leftarrow r\tilde{x}$ recovers an ϵ-approximate minimizer to the original problem. To see this, let x^* be the minimizer to (C.1); it is clear under a simple rescaling and linearity of norms that $\tilde{x}^* := x^*/r$ is the minimizer to (C.2). Next, let $\tilde{x}$ be any point in $[-1,1]^m$ with

$$\left\|A\tilde{x} - \tilde{b}\right\|_\infty - \left\|A\tilde{x}^* - \tilde{b}\right\|_\infty \le \frac{\epsilon}{r}.$$

By linearity of norms, we see that $x = r\tilde{x}$ has $x \in [-r,r]^m$ and

$$\|Ax - b\|_\infty - \|Ax^* - b\|_\infty \le \epsilon,$$

i.e. x is an ϵ-approximate minimizer to (C.1). To bound the complexity of solving (C.2) to ϵ/r additive accuracy, it suffices to invoke Theorem 15.

Next, to deal with the unconstrained case with the promise $\|x_0 - x^*\|_\infty \le r$, it suffices to perform a change of variables $b' \leftarrow b - Ax_0$, $x' \leftarrow x - x_0$, and solve the problem

$$\min_{x' \in [-r,r]^m} \|Ax' - b'\|_\infty = \min_{x' \in [-r,r]^m} \|A(x - x_0) - (b - Ax_0)\|_\infty = \min_{x \in \mathbb{R}^m} \|Ax - b\|_\infty .$$

To see the last inequality, we use the guarantee that $\|x_0 - x^*\|_\infty \le r$, i.e. $x^* - x_0$ is a valid point x'. We then can invoke the general box-constrained case with radius r.

Finally, we remark that similar additive shifts and rescalings allow us to handle the more general box constraint $\prod_{j \in [m]} [\ell_j, r_j]$ with appropriate (weighted) dependences on the quantities $r_j - \ell_j$.

C.1.3 Convergence rates of first-order methods

In this section, we give guarantees for the convergence rates of the classical unaccelerated first-order methods of gradient descent in general norms and coordinate descent.

Gradient descent in general norms

We briefly review the basic guarantees of gradient descent applied to a convex function f which is L-smooth in an arbitrary norm $\|\cdot\|$. The general framework of gradient descent initializes at some point x^0 and iteratively maximizes the primal progress using the upper bound guaranteed by the smoothness. In particular, we perform the following update:

$$x^{k+1} \leftarrow \mathrm{argmin}_y \Big\{ f(x^k) + \nabla f(x^k)^\top (y - x^k) + \frac{L}{2}\|y - x^k\|^2 \Big\}$$

The $O(\frac{1}{T})$ convergence rate of gradient descent is well-known in the literature. We state the convergence guarantee here.

Lemma 239. *Let x^T be the result of running gradient descent for T iterations. Then for the global minimizer x^*, we have $f(x^T) - f(x^*) \le \frac{2LR^2}{T}$, where $R = \max_{y : f(y) \le f(x^0)} \|y - x^*\|$.*

Coordinate descent

Next, we briefly review the basic guarantees of randomized coordinate descent when applied to a convex function f which is L_j-smooth in the j^{th} coordinate. Here, we analyze the convergence rate of the simple unaccelerated variant of coordinate descent where coordinate j is sampled with probability $\frac{L_j}{S}$, where $S := \sum_j L_j$. In particular, we perform the following update after sampling a coordinate j:

$$x^{k+1} \leftarrow \text{argmin}_y \left\{ f(x^k) + \nabla_j f(x^k)^\top (y - x^k) + \frac{L_j}{2} |y_j - x_j^k|^2 \right\} = x^k - \frac{1}{L_j} \nabla_j f(x^k)$$

Here, we give the convergence rate of this simple coordinate descent algorithm.

Lemma 240. *Let x^T be the result of running gradient descent for T iterations. Then for the global minimizer x^*, we have $f(x^T) - f(x^*) \leq \frac{2SR^2}{T}$, where $R = \max_{y: f(y) \leq f(x^0)} \|y - x^*\|_2$.*

We remark that for any randomized iterative method for minimizing a convex function which converges in expectation, it is easy to use Markov's inequality to bound the convergence with constant probability. For example, if an algorithm terminates with a ϵ-approximate minimizer on expectation, with probability at least $\frac{1}{2}$ it terminates with a 2ϵ-approximate minimizer. Thus, if one desires a high probability result for the approximate minimization, the runtime only incurs a logarithmic multiplicative loss in the failure probability.

C.1.4 Proof of Lemma 239

First we give an intermediate progress bound which will be useful in the final proof.

Lemma 241. $f(x^k) - f(x^{k+1}) \geq \frac{1}{2L} \|\nabla f(x^k)\|_*^2$

Proof. We will prove that $\min_y \left\{ \nabla f(x)^\top (y - x) + \frac{L}{2} \|y - x\|^2 \right\} \leq -\frac{1}{2L} \|\nabla f(x)\|_*^2$; clearly this yields the desired claim. Let z be such that $\|z\| = 1$ and $z^\top \nabla f(x) = \|\nabla f(x)\|_*$, by the definition of dual norm; let $y = x - \frac{\|\nabla f(x)\|_*}{L} z$. Then,

$$\nabla f(x)^\top (y - x) + \frac{L}{2} \|y - x\|^2 = -\left(\frac{\|\nabla f(x)\|_*}{L} \right) z^\top \nabla f(x) + \frac{L}{2} \frac{\|\nabla f(x)\|_*^2}{L^2} \|z\|^2 = -\frac{1}{2L} \|\nabla f(x)\|_*^2$$

Thus, the minimizer of the upper bound yields the desired progress result. □

Next, we prove Lemma 239.

Proof. Let $\epsilon_k := f(x^k) - f(x^*)$. Note that by convexity and Cauchy-Schwarz, we have

$$f(x^k) - f(x^*) \leq (\nabla f(x^k))^\top (f(x^k) - f(x^*)) \leq \|\nabla f(x^k)\|_* \|x^k - x^*\|$$

Thus, we have the two equations $\epsilon_k - \epsilon_{k+1} \geq \frac{1}{2L}\|\nabla f(x^k)\|_*^2$ and $\epsilon_k \leq R\|\nabla f(x^k)\|_*$. Combining the two, it's easy to see that

$$\epsilon_k^2 \leq 2LR^2(\epsilon_k - \epsilon_{k+1}) \leftrightarrow \Big(\frac{1}{\epsilon_{k+1}} - \frac{1}{\epsilon_k}\Big) \geq \frac{\epsilon_k}{2LR^2\epsilon_{k+1}} \geq \frac{1}{2LR^2}$$

Thus, telescoping we have $\frac{1}{\epsilon_T} \geq \frac{T}{2LR^2}$, which yields the desired rate of convergence. □

C.1.5 Proof of Lemma 240

The progress of a step in the j^{th} coordinate is thus lower bounded by $-\frac{1}{2L_j}|\nabla_j f(x^k)|^2$, which can be verified by computing the upper bound on $f(x^{k+1})$. The analysis of convergence follows directly from the following result on the expected progress of a single step.

Lemma 242. $f(x^k) - \mathbb{E}_k[f(x^{k+1})] \geq \frac{1}{2S}\|\nabla f(x^k)\|_2^2$

Proof. We directly compute the expectation. We have

$$\mathbb{E}[f(x^{k+1})] = \sum_j \frac{L_j}{S}\Big(f(x^k) - \frac{1}{2L_j}|\nabla_j f(x^k)|^2\Big) = f(x^k) - \frac{1}{2S}\sum_j |\nabla_j f(x^k)|^2$$

□

Thus, we can immediately plug in this expected progress result into the convergence rate proof of gradient descent, and obtain the desired result.

C.2 Missing proofs from Section 4.4

C.2.1 Reducing undirected maximum flow to ℓ_∞ regression

In this section, we prove Lemma 52, via giving the reduction and analyzing its convergence. First, suppose we have a subroutine, Almost-Route, which takes in matrices R (an $\alpha = \tilde{O}(1)$-congestion approximator), B (an edge-incidence matrix), U (the capacities of edges), α, an error tolerance ϵ, and a demand vector d, and returns some x such that

$$2\alpha\|RBUx - Rd\|_\infty + \|x\|_\infty \leq (1+\epsilon)(2\alpha\|RBUx^* - Rd\|_\infty + \|x^*\|_\infty) := (1+\epsilon)\mathrm{OPT}(d) \quad \text{(C.3)}$$

Here, under a change of variables we have that $x = U^{-1}f$. Note that we are writing with an ϵ-multiplicative approximation to OPT instead of an additive one. We do this without loss of generality: assume we have scaled the problem appropriately so that the optimal value is 1, which it will be when we find the true maximum flow instead of the minimum congestion flow. We can find this optimal value via a binary search, as we argued before, losing a $\tilde{O}(1)$ factor in the runtime.

Now, we show a key property of the function we try to minimize. Intuitively, the next lemma says that if we are able to ϵ-approximately minimize our regression problem, the cost of routing the residual demands $d - Bf$ is only an ϵ fraction of routing the original demands, allowing us to quickly recurse. This is a restatement of Lemma 2.2 in [482].

Lemma 243. *Define the change of variables $Ux = f$. Suppose $2\alpha\|RBUx - Rd\|_\infty + \|x\|_\infty = 2\alpha\|R(d - Bf)\|_\infty + \|U^{-1}f\|_\infty \leq (1+\epsilon)\mathrm{OPT}(d)$. Then, $\|R(d - Bf)\|_\infty \leq \epsilon\|Rd\|_\infty$.*

Proof. Let f' be the optimal routing of the residual demands $d - Bf$, namely the argument which achieves $\mathrm{OPT}(d - Bf)$. Then, $Bf' = d - Bf$, and by the definition of a congestion approximator,

$$\mathrm{OPT}(d - Bf) = \|U^{-1}f'\|_\infty + 2\alpha\|R((d - Bf) - Bf')\|_\infty = \|U^{-1}f'\|_\infty \leq \alpha\|R(d - Bf)\|_\infty \quad \text{(C.4)}$$

For simplicity we write $d' := d - Bf$. Furthermore, we have by assumption of the quality of the initial solution f,

$$\begin{aligned}
\mathrm{OPT}(d) + \alpha\|Rd'\|_\infty &\leq \|U^{-1}(f + f')\|_\infty + 2\alpha\|R(d - B(f + f'))\|_\infty + \alpha\|Rd'\|_\infty && \text{(C.5)}\\
&\leq \|U^{-1}f\|_\infty + \|U^{-1}f'\|_\infty + \alpha\|Rd'\|_\infty && \text{(C.6)}\\
&\leq \|U^{-1}f\|_\infty + 2\alpha\|Rd'\|_\infty \leq (1+\epsilon)\mathrm{OPT}(d) && \text{(C.7)}
\end{aligned}$$

Here, we used that $d = B(f + f')$ and our bound $\|U^{-1}f'\|_\infty \leq \alpha\|Rd'\|_\infty$. Subtracting $\mathrm{OPT}(d)$, and noting that $\mathrm{OPT}(d) \leq \alpha\|Rd\|_\infty$, we have the desired claim. □

Now, we give the full reduction to calling Almost-Route. Note that it was shown in [482] that routing through a maximal spanning tree yields an $O(m)$-congestion approximator.

We now need to prove the correctness of our algorithm. This is a restatement of ideas presented in [482].

Lemma 244. *The output of* Flow-To-Regress *is an ϵ-approximate solution to the minimum congestion flow problem.*

Proof. By the guarantees of Almost-Route, we have the following guarantees:

$$\|U^{-1}f^0\|_\infty + 2\alpha\|Rd^1\|_\infty \leq (1+\epsilon)\mathrm{OPT}(d), \quad \text{(C.8)}$$

$$\|U^{-1}f^k\|_\infty + 2\alpha\|Rd^{k+1}\|_\infty \leq \frac{3}{2}\mathrm{OPT}(d^k) \leq \frac{3}{2}\alpha\|Rd^k\|_\infty, k \geq 1. \quad \text{(C.9)}$$

Now, using the second inequality and repeatedly applying it to the first, we have the following guarantee:

$$\frac{1}{2}\alpha\|Rd^1\|_\infty + \|U^{-1}f^0\|_\infty + \ldots + \|U^{-1}f^T\|_\infty \leq (1+\epsilon)\mathrm{OPT}(d). \quad \text{(C.10)}$$

$f^{final} = \text{Flow-To-Regress}(G, d, \epsilon)$

1. Let $T = \log 2m$.
2. Initialize $d^0 = d$. Initialize $f^0 = U\text{Almost-Route}(R, B, U, d^0, \alpha, \epsilon)$.
3. Let $f^{final} = f^0$.
4. Iterate for $k = 1, 2, \ldots T$:
   (a) Let $d^k = d^{k-1} - Bf^{k-1}$.
   (b) Let $f^k = U\text{Almost-Route}(R, B, U, D^k, \alpha, \frac{1}{2})$.
   (c) Let $f^{final} = f^{final} + f^k$.
5. Let f^{T+1} be an (exact) routing of $d^k - Bf^k$ in a maximal spanning tree. Let $f^{final} + f^{T+1}$.
6. Return f^{final}

Figure C.1: The reduction from solving the approximate maximum flow problem to solving $\tilde{O}(1)$ approximate regression problems.

It suffices to note that by our choice of T and seeing that by applying Lemma 243 T times, we have $\alpha\|Rd^{T+1}\|_\infty \leq \frac{1}{2m}\alpha\|Rd^1\|_\infty$. Thus because we routed d^{T+1} exactly through a m-congestion approximator, we have $\|U^{-1}f^{T+1}\|_\infty \leq \frac{1}{2}\alpha\|Rd^1\|_\infty$. Finally, $Bf^{final} = d$, and

$$\|U^{-1}f^{final}\|_\infty \leq \|U^{-1}f^{T+1}\|_\infty + \|U^{-1}f^0\|_\infty + \ldots + \|U^{-1}f^T\|_\infty \tag{C.11}$$

$$\leq \|U^{-1}f^{final}\|_\infty \leq \frac{1}{2}\alpha\|Rd^1\|_\infty + \|U^{-1}f^0\|_\infty + \ldots + \|U^{-1}f^T\|_\infty \tag{C.12}$$

$$\leq (1+\epsilon)\text{OPT}(d). \tag{C.13}$$

□

Lemma 245. *The runtime of our routine* Flow-To-Regress *is the cost of solving the first associated regression problem, $2\alpha\|RBUx - Rd\|_\infty + \|x\|_\infty$, to an ϵ approximation, plus an additional $\tilde{O}(m)$ additive overhead.*

Proof. We analyze the time of each of the calls to Almost-Route. Clearly, the first call is the cost of solving the first associated regression problem.

Note that we have flexibility in terms of how to implement Almost-Route; for all remaining calls, we consider the implementation in the form of unaccelerated gradient descent in the ℓ_∞ norm. The runtime as we demonstrated in Appendix C.1.3 for each round k is

$$\frac{m\|f_*^k\|_\infty^2\|\alpha RBU\|_\infty^2}{(\frac{1}{2})^2} = \tilde{O}(m) \tag{C.14}$$

where f_*^k is the optimal solution to the k^{th} regression problem. Here, we used the known properties of αRBU, as well as the fact that the implications of Lemma 243 allow us to bound the ℓ_∞ norm of the optimal solution by $O(1)$ as well. □

As a final note in the proof of Lemma 52, observe that to optimize the first objective $2\alpha \|Ax - b\|_\infty + \|x\|_\infty$ it suffices to binary search over values $r \geq \|U^{-1}f\|_\infty = \|x\|_\infty$, and solve the associated regression problem $\|Ax - b\|_\infty$ over $x \in [-r, r]^m$. More formally, since r is our guess of OPT to the original flow problem, we repeatedly solve the problem over $[-r, r]^m$ to ϵr additive error; if the conclusion is that the optimal value cannot be 0 (i.e. the additive approximation is larger than ϵr), then we conclude that this value of r is not routable. This only incurs a multiplicative loss in the runtime by a factor of $\tilde{O}(1)$, due to the binary search. By normalizing x, b appropriately, it suffices to consider the case where $r = 1$, and solve to ϵ additive error (see Appendix C.1.2). Finally, we need only consider the case where $\|b\|_\infty \leq \|A\|_\infty$, as $x \in [-1, 1]^\infty$ the unit box, so the demands are clearly not routable otherwise.

C.2.2 Reducing directed maximum flow to undirected maximum flow

In this section, we give an overview of the main result in [363]. In particular, we prove the following statement, which is used in our algorithms for finding exact maximum flows in unit-capacity graphs.

Lemma 246 (Summary of results in [363])**.** *Suppose we wish to find an $s - t$ maximum flow in a unit-capacity directed (multi)graph G with m edges and maximum flow value F. Then, it suffices to find the $s - t$ maximum flow f_{max} in an undirected (multi)graph G' with $O(m)$ edges, such that edges of G' have capacity $\frac{1}{2}$, and the maximum flow in G' has value $F + \frac{m}{2}$. Furthermore, we are able to initialize the undirected maximum flow algorithm in G' with some f_{init} such that $\|f_{init} - f_{max}\|_2^2 = F$.*

Proof. First, we give the construction of the undirected graph G'. For every directed edge (u, v) of weight 1 in G, G' has the undirected edges (s, v), (v, u), and (u, t) of weight $\frac{1}{2}$. Clearly, G' has $O(m)$ edges, since each edge in G is replaced with 3 edges in G'.

Next, we give the (algorithmic) proof that one can recover a maximum flow in G from a maximum flow in G', and that the maximum flow in G' has value $F + \frac{m}{2}$. Consider the following algorithm.

We will now prove correctness of the algorithm UMF-TO-DMF, namely that $f_{final} - f_{init}$ is a maximum flow in graph G. To do so, we show that f_{final} has value $\frac{m}{2} + F$, and that $f_{final} - f_{init}$ puts flow only in the (u, v) direction and does not put any flow on any new edges (s, v) or (u, t). Note that this immediately implies the statement $\|f_{init} - f_{max}\|_2^2 = F$.

We begin by showing that f_{final} has value $\frac{m}{2} + F$. The residual graph of G' with respect to the flow f_{init} is the directed graph G. Thus, the maximum flow in the residual graph has value F by assumption, and the flow f_{init} has value $\frac{m}{2}$, yielding the conclusion.

f = UMF-TO-DMF(G)

1. Let G' be the undirected graph with edges $(s,v),(v,u),(u,t)$ of weight $\frac{1}{2}$ for every directed edge (u,v) in G.
2. Let f_{init} be the flow which puts $\frac{1}{2}$ units of flow on each of the $(s,v),(v,u),(u,t)$.
3. Compute f_{final}, the maximum flow of G'.
4. Return $f_{final} - f_{init}$.

Figure C.2: Recovering a maximum flow in directed G via a maximum flow in undirected G'.

Next, we show that for every edge (u,v) in G' which resulted from a directed edge (u,v) in G, $f_{final} - f_{init}$ puts flow only in the (u,v) direction, and does not violate the capacity constraint. This is simple to see because f_{final} puts a flow with value in $\{-\frac{1}{2}, 0, \frac{1}{2}\}$ in the (u,v) direction, and $-f_{init}$ puts a flow with value $\frac{1}{2}$ in the (u,v) direction; adding yields the result.

Finally, we show that $f_{final} - f_{init}$ puts no flow on any of the new edges (s,v) (the same statement holds for edges (u,t) by a similar argument). Again, f_{final} puts a flow with value in $\{-\frac{1}{2}, 0, \frac{1}{2}\}$ in the (v,s) direction, and $-f_{init}$ puts a flow with value $\frac{1}{2}$ in the (v,s) direction, thus $f_{final} - f_{init}$ puts a flow with nonnegative value in the (v,s) direction. If this value was strictly positive, it would be part of a path in the flow decomposition sending flow into s, contradicting the maximality of f_{final}. □

Appendix D

Deferred proofs from Chapter 5

D.1 Area convexity and optimal transport

In this section, we provide the optimization tools which will be used in our semi-streaming solver of Section 5.3. These tools were originally developed in our earlier work [293] to obtain a direct parallel solver for optimal transport. For this reason, we choose to first give an overview of the optimal transport problem in Section D.1.1, and then our general-purpose tools in Section D.1.3.

D.1.1 Optimal transport

Optimal transport is playing an increasingly important role as a subroutine in tasks arising in machine learning [32], computer vision [90, 490], robust optimization [215, 83], and statistics [434]. Given these applications for large scale learning, designing algorithms for efficiently approximately solving the problem has been the subject of extensive recent research [151, 25, 236, 115, 208, 365, 82, 450].

Given two vectors r and c in the n-dimensional probability simplex Δ^n and a cost matrix $\mathbf{C} \in \mathbb{R}^{n\times n1}_{\geq 0}$[1], the optimal transportation problem is

$$\min_{\mathbf{X}\in\mathcal{U}_{r,c}} \langle \mathbf{C}, \mathbf{X}\rangle, \quad \text{where} \quad \mathcal{U}_{r,c} := \left\{ X \in \mathbb{R}^{n\times n}_{\geq 0},\ \mathbf{X}\mathbf{1} = r,\ \mathbf{X}^\top \mathbf{1} = c \right\}. \tag{D.1}$$

This problem arises from defining the *Wasserstein* or *Earth mover's* distance between discrete probability measures r and c, as the cheapest coupling between the distributions, where the cost of the coupling $\mathbf{X} \in \mathcal{U}_{r,c}$ is $\langle \mathbf{C}, \mathbf{X}\rangle$. If r and c are viewed as distributions of masses placed on n points in some space (typically metric), the Wasserstein distance is the cheapest way to move mass to transform r into c. In (D.1), $\mathbf{X}$ represents the transport plan ($\mathbf{X}_{ij}$ is the amount moved from r_i

[1] Similarly to earlier works, we focus on square matrices; generalizations to rectangular matrices are straightforward.

to c_j) and $\mathbf{C}$ represents the cost of movement ($\mathbf{C}_{ij}$ is the cost of moving mass from r_i to c_j).

Throughout, the value of (D.1) is denoted OPT. We call $\hat{\mathbf{X}} \in \mathcal{U}_{r,c}$ an *ϵ-approximate transportation plan* if $\left\langle \mathbf{C}, \hat{\mathbf{X}} \right\rangle \leq \text{OPT} + \epsilon$. Our goal is to design an efficient algorithm to produce such a $\hat{\mathbf{X}}$.

Our contributions

Our main contribution is an algorithm running in $\tilde{O}(\|\mathbf{C}\|_{\max}/\epsilon)$ parallelelizable iterations[2] and $\tilde{O}(n^2\|\mathbf{C}\|_{\max}/\epsilon)$ total work producing an ϵ-approximate transport plan.

Matching runtimes were given in the recent work of [82, 450]. Their runtimes were obtained via reductions to matrix scaling and positive linear programming, each well-studied problems in theoretical computer science. However, the matrix scaling algorithm is a second-order Newton-type method which makes calls to structured linear system solvers, and the positive LP algorithm is not parallelizable (i.e. has depth polynomial in dimension). These features potentially limit the practicality of these algorithms. The key remaining open question this paper addresses is, *is there an efficient first-order, parallelizable algorithm for approximating optimal transport?* We answer this affirmatively and give an efficient, parallelizable primal-dual first-order method; the only additional overhead is a scheme for implementing steps, incurring roughly an additional $\log \epsilon^{-1}$ factor.

Our approach heavily leverages the recent improvement to the maximum flow problem, and more broadly two-player games on a simplex (ℓ_1 ball) and a box (ℓ_∞ ball), due to the breakthrough work of [483]. First, we recast (D.1) as a minimax game between a box and a simplex, proving correctness via a rounding procedure known in the optimal transport literature. Second, we show how to adapt the dual extrapolation scheme under the weaker convergence requirements of area-convexity, following [483], to obtain an approximate minimizer to our primal-dual objective in the stated runtime. En route, we slightly simplify analysis in [483] and relate it more closely to the existing extragradient literature.

Finally, we give preliminary experimental evidence showing our algorithm can be practical, and highlight some open directions in bridging the gap between theory and practice of our method, as well as accelerated gradient schemes [208, 365] and Sinkhorn iteration.

Previous work

Optimal transport. The problem of giving efficient algorithms to find ϵ-approximate transport plans $\hat{\mathbf{X}}$ which run in nearly linear time[3] has been addressed by a line of recent work, starting with [151] and improved upon in [236, 25, 208, 365, 82, 450]. We briefly discuss their approaches here.

[2] Our iterations consist of vector operations and matrix-vector products, which are easily parallelizable. Throughout $\|\mathbf{C}\|_{\max}$ is the largest entry of C.

[3] We use "nearly linear" to describe complexities which have an $n^2\text{polylog}(n)$ dependence on the dimension (where the size of input $\mathbf{C}$ is n^2), and polynomial dependence on $\|\mathbf{C}\|_{\max}, \epsilon^{-1}$.

Works by [151, 25] studied the Sinkhorn algorithm, an alternating minimization scheme. Regularizing (D.1) with an η^{-1} multiple of entropy and computing the dual, we arrive at the problem

$$\min_{x,y\in\mathbb{R}^n} \mathbf{1}^\top \mathbf{B}_{\eta\mathbf{C}}(x,y)\mathbf{1} - r^\top x - c^\top y \quad \text{where} \quad \mathbf{B}_{\eta\mathbf{C}}(x,y)_{ij} = e^{x_i+y_j-\eta\mathbf{C}_{ij}}.$$

This problem is equivalent to computing diagonal scalings $\mathbf{X}$ and $\mathbf{Y}$ for $\mathbf{M} = \exp(-\eta\mathbf{C})$ such that $\mathbf{XMY}$ has row sums r and column sums c. The Sinkhorn iteration alternates fixing the row sums and the column sums by left and right scaling by diagonal matrices until an approximation of such scalings is found, or equivalently until $\mathbf{XMY}$ is close to being in $\mathcal{U}_{r,c}$.

As shown in [25], we can round the resulting almost-transportation plan to a transportation plan which lies in $\mathcal{U}_{r,c}$ in linear time, losing at most $2\|\mathbf{C}\|_{\max}(\|\mathbf{X}\mathbf{1} - r\|_1 + \left\|\mathbf{X}^\top\mathbf{1} - c\right\|_1)$ in the objective. Further, [25] showed that $\tilde{O}(\|\mathbf{C}\|_{\max}^3/\epsilon^3)$ iterations of this scheme sufficed to obtain a matrix which $\epsilon/\|\mathbf{C}\|_{\max}$-approximately meets the demands in ℓ_1 with good objective value, by analyzing it as an instance of mirror descent with an entropic regularizer. The same work proposed an alternative algorithm, Greenkhorn, based on greedy coordinate descent. [208, 365] showed that $\tilde{O}\left(\|\mathbf{C}\|_{\max}^2/\epsilon^2\right)$ iterations, corresponding to $\tilde{O}\left(n^2\|\mathbf{C}\|_{\max}^2/\epsilon^2\right)$ work, suffice for both Sinkhorn and Greenkhorn, the current state-of-the-art for this line of analysis.

An alternative approach based on first-order methods was studied by [208, 365]. These works considered minimizing an entropy-regularized Equation D.1; the resulting weighted softmax function is prevalent in the literature on approximate linear programming [419], and has found similar applications in near-linear algorithms for maximum flow [482, 318, 486] and positive linear programming [541, 23]. An unaccelerated algorithm, viewable as ℓ_∞ gradient descent, was analyzed in [208] and ran in $\tilde{O}(\|\mathbf{C}\|_{\max}^2/\epsilon^2)$ iterations. Further, an accelerated algorithm was discussed, for which the authors claimed an $\tilde{O}(n^{1/4}\|\mathbf{C}\|_{\max}^{0.5}/\epsilon)$ iteration count. [365] showed that the algorithm had an additional dependence on a parameter as bad as $n^{1/4}$, roughly due to a gap between the ℓ_2 and ℓ_∞ norms. Thus, the state of the art runtime in this line is the better of $\tilde{O}\left(n^{2.5}\|\mathbf{C}\|_{\max}/\epsilon\right)$, $\tilde{O}\left(n^2\|\mathbf{C}\|_{\max}^2/\epsilon^2\right)$ operations. The dependence on dimension of the former of these runtimes matches that of the linear programming solver of [348, 349], which obtain a polylogarithmic dependence on ϵ^{-1}, rather than a polynomial dependence; thus, the question of obtaining an accelerated ϵ^{-1} dependence without worse dimension dependence remained open.

This was partially settled in [82, 450], which studied the relationship of optimal transport to fundamental algorithmic problems in theoretical computer science, namely positive linear programming and matrix scaling, for which significantly-improved runtimes have been recently obtained [23, 552, 145]. In particular, they showed that optimal transport could be reduced to instances of either of these objectives, for which $\tilde{O}\left(\|\mathbf{C}\|_{\max}/\epsilon\right)$ iterations, each of which required linear $O(n^2)$ work, sufficed. However, both of these reductions are based on black-box methods for which practical implementations are not known; furthermore, in the case of positive linear programming a parallel $\tilde{O}(1/\epsilon)$-iteration algorithm is not known. [82] also showed any polynomial improvement to

Year	Author	Complexity	Approach	1st-order	Parallel
2015	[349]	$\tilde{O}(n^{2.5})$	Interior point	No	No
2017-19	[25]	$\tilde{O}(n^2\|\mathbf{C}\|_{\max}^2/\epsilon^2)$	Sink/Greenkhorn	Yes	Yes
2018	[208]	$\tilde{O}(n^2\|\mathbf{C}\|_{\max}^2/\epsilon^2)$	Gradient descent	Yes	Yes
2018-19	[365]	$\tilde{O}(n^{2.5}\|\mathbf{C}\|_{\max}/\epsilon)$	Acceleration	Yes	Yes
2018	[82]	$\tilde{O}(n^2\|\mathbf{C}\|_{\max}/\epsilon)$	Matrix scaling	No	Yes
2018-19	[82, 450]	$\tilde{O}(n^2\|\mathbf{C}\|_{\max}/\epsilon)$	Positive LP	Yes	No
2019	This work	$\tilde{O}(n^2\|\mathbf{C}\|_{\max}/\epsilon)$	Dual extrapolation	Yes	Yes

Table D.1: Optimal transport algorithms. Algorithms using second-order information use potentially-expensive SDD system solvers; the runtime analysis of Sink/Greenkhorn is due to [208, 365].

the runtime of our paper in the dependence on either ϵ or n would result in maximum-cardinality bipartite matching in dense graphs faster than $\tilde{O}(n^{2.5})$ without fast matrix multiplication [468], a fundamental open problem unresolved for almost 50 years [274].

Specializations of the transportation problem to ℓ_p metric spaces or arising from geometric settings have been studied [479, 11, 28]. These specialized approaches seem fundamentally different than those concerning the more general transportation problem.

Finally, we note recent work [24] showed the promise of using the Nyström method for low-rank approximations to achieve speedup in theory and practice for transport problems arising from specific metrics. As our method is based on matrix-vector operations, where low-rank approximations may be applicable, we find it interesting to see if our method can be combined with these improvements.

Remark. During the revision process for this work, an independent result [336] was published to arXiv, obtaining improved runtimes for optimal transport via a combinatorial algorithm. The work obtains a runtime of $\tilde{O}(n^2\|\mathbf{C}\|_{\max}/\epsilon + n\|\mathbf{C}\|_{\max}^2/\epsilon^2)$, which is worse than our runtime by a low-order term. Furthermore, it does not appear to be parallelizable.

Box-simplex objectives. Our main result follows from improved algorithms for bilinear minimax problems over one simplex domain and one box domain developed in [483]. This fundamental minimax problem captures ℓ_1 and ℓ_∞ regression over a simplex and box respectively, and inspired the development of conjugate smoothing [419] as well as mirror prox / dual extrapolation [415, 420]. These latter two approaches are extragradient methods (using two gradient operations per iteration rather than one) for approximately solving a family of problems, which includes convex minimization and finding a saddle point to a convex-concave function. These methods simulate backwards Euler discretization of the gradient flow, similar to how mirror descent simulates forwards Euler discretization [193]. The role of the extragradient step is a fixed point iteration (of two steps) which is a good approximation of the backwards Euler step when the operator is Lipschitz.

Nonetheless, the analysis of [415, 420] fell short in obtaining a $1/T$ rate of convergence without

worse dependence on dimension for these domains, where T is the iteration count (which would correspond to a $\tilde{O}(1/\epsilon)$ runtime for approximate minimization). The fundamental barrier was that over a box, any strongly-convex regularizer in the ℓ_∞ norm has a dimension-dependent domain size (shown in [486]). This barrier can also be viewed as the reason for the worse dimension dependence in the accelerated scheme of [208, 365].

The primary insight of [483] was that previous approaches attempted to regularize the schemes of [415, 420] with separable regularizers, i.e. the sum of a regularizer which depends only on the primal block and one which depends only on the dual. If, say, the domain of the primal block was a box, then such a regularization scheme would run into the ℓ_∞ barrier and incur a worse dependence on dimension. However, by more carefully analyzing the requirements of these algorithms, [483] constructed a non-separable regularizer with small domain size, satisfying a property termed *area-convexity* which sufficed for provable convergence of dual extrapolation [420]. Interestingly, the property seems specialized to dual extrapolation and not mirror prox [415].

D.1.2 Overview

In this section, we describe a reformulation of (D.1) as a primal-dual objective, which we solve approximately in Section D.1.3. We adapt the view of [82, 450] of the objective (D.1) as a positive linear program. Let d be the (vectorized) cost matrix $\mathbf{C}$ associated with the instance and let Δ^{n^2} be the n^2 dimensional simplex[4]. We recall r, c are specified row and column sums with $\mathbf{1}^\top r = \mathbf{1}^\top c = 1$. The optimal transport problem can be written as, for $m = n^2$, and $\mathbf{A} \in \{0,1\}^{2n \times m}, b \in \mathbb{R}^{2n}_{\geq 0}$, for $\mathbf{A}$ the (unsigned) *edge-incidence matrix* of the underlying bipartite graph and b the concatenation of r and c.

$$\min_{x \in \Delta^m, \mathbf{A}x = b} d^\top x. \tag{D.2}$$

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}.$$

Figure D.1: Edge-incidence matrix $\mathbf{A}$ of a 3×3 bipartite graph and uniform demands.

In particular, $\mathbf{A}$ is the 0-1 matrix on $V \times E$ such that $\mathbf{A}_{ve} = 1$ iff v is an endpoint of edge e. We summarize some additional properties of the constraint matrix $\mathbf{A}$ and vector b.

[4]We use d because $\mathbf{C}$ often arises from distances in a metric space, and to avoid overloading c.

Fact 37. $\mathbf{A}$, b *have the following properties.*

1. $\mathbf{A} \in \{0,1\}^{2n \times m}$ *has 2-sparse columns and n-sparse rows. Thus* $\|\mathbf{A}\|_{1\to 1} = 2$.
2. $b^\top = \begin{pmatrix} r^\top & c^\top \end{pmatrix}$, *so that* $\|b\|_1 = 2$.
3. $\mathbf{A}$ *has* $2n^2$ *nonzero entries.*

Section D.1.4 recalls the proof of the following theorem, which first appeared in [25].

Theorem 80 (Rounding guarantee, Lemma 7 in [25])**.** *There is an algorithm which takes* $\tilde{x}$ *with* $\|\mathbf{A}\tilde{x} - b\|_1 \le \delta$ *and produces* $\hat{x}$ *in* $O(n^2)$ *time, with*

$$\mathbf{A}\hat{x} = b, \|\tilde{x} - \hat{x}\|_1 \le 2\delta.$$

We now show how the rounding procedure gives a roadmap for our approach. Consider the following ℓ_1 regression objective over the simplex (a similar penalized objective appeared in [482]):

$$\min_{x \in \Delta^m} d^\top x + 2\|d\|_\infty \|\mathbf{A}x - b\|_1 . \tag{D.3}$$

We show that the penalized objective value is still OPT, and furthermore any approximate minimizer yields an approximate transport plan.

Lemma 247 (Penalized ℓ_1 regression)**.** *The value of* (D.3) *is* OPT. *Also, given* $\tilde{x}$, *an* ϵ*-approximate minimizer to* (D.3)*, we can find* ϵ*-approximate transportation plan* $\hat{x}$ *in* $O(n^2)$ *time.*

Proof. Recall $\text{OPT} = \min_{x \in \Delta^m, \mathbf{A}x = b} d^\top x$. Let $\tilde{x}$ be the minimizing argument in (D.3). We claim there is some optimal $\tilde{x}$ with $\mathbf{A}\tilde{x} = b$; clearly, the first claim is then true. Suppose otherwise, and let $\|\mathbf{A}\tilde{x} - b\|_1 = \delta > 0$. Then, let $\hat{x}$ be the result of the algorithm in Theorem 80, applied to $\tilde{x}$, so that $\mathbf{A}\hat{x} = b, \|\tilde{x} - \hat{x}\|_1 \le 2\delta$. We then have

$$d^\top \hat{x} + 2\|d\|_\infty \|\mathbf{A}\hat{x} - b\|_1 = d^\top(\hat{x} - \tilde{x}) + d^\top \tilde{x} \le d^\top \tilde{x} + \|d\|_\infty \|\hat{x} - \tilde{x}\|_1 \le d^\top \tilde{x} + 2\|d\|_\infty \delta.$$

The objective value of $\hat{x}$ is no more than of $\tilde{x}$, a contradiction. By this discussion, we can take any approximate minimizer to (D.3) and round it to a transport plan without increasing the objective. □

Section D.1.3 proves Theorem 81, which says we can efficiently find an approximate minimizer to (D.3).

Theorem 81 (Approximate ℓ_1 regression over the simplex)**.** *There is an algorithm (Algorithm 71) taking input* ϵ, *which has* $O((\|d\|_\infty \log n \log \gamma)/\epsilon)$ *parallel depth for* $\gamma = \log n \cdot \|d\|_\infty / \epsilon$, *and total work* $O(n^2(\|d\|_\infty \log n \log \gamma)/\epsilon)$, *and obtains* $\tilde{x}$ *an* ϵ*-additive approximation to the objective in* (D.3).

We will approach proving Theorem 81 through a primal-dual viewpoint, in light of the following (based on the definition of the ℓ_1 norm):

$$\min_{x\in\Delta^m} d^\top x + 2\|d\|_\infty \|\mathbf{A}x - b\|_1 = \min_{x\in\Delta^m} \max_{y\in[-1,1]^{2n}} d^\top x + 2\|d\|_\infty \left(y^\top \mathbf{A}x - b^\top y\right). \tag{D.4}$$

Further, a low-*duality gap* pair to (D.4) yields an approximate minimizer to (D.3).

Lemma 248 (Duality gap to error)**.** *Suppose x, y is feasible ($x \in \Delta^m, y \in [-1,1]^{2n}$), and for any feasible u, v,*

$$\left(d^\top x + 2\|d\|_\infty \left(v^\top \mathbf{A}x - b^\top v\right)\right) - \left(d^\top u + 2\|d\|_\infty \left(y^\top \mathbf{A}u - b^\top y\right)\right) \le \delta.$$

Then, we have $d^\top x + 2\|d\|_\infty \|\mathbf{A}x - b\|_1 \le \delta + \mathrm{OPT}$.

Proof. The result follows from maximizing over v, and noting that for the minimizing u,

$$d^\top u + 2\|d\|_\infty \left(y^\top \mathbf{A}u - b^\top y\right) \le d^\top u + 2\|d\|_\infty \|\mathbf{A}u - b\|_1 = \mathrm{OPT}.$$

□

Correspondingly, Section D.1.3 gives an algorithm which obtains (x, y) with bounded duality gap within the runtime of Theorem 81.

Additional notation

We recall the definitions of a Bregman divergence of a regularizer and the proximal operator of a divergence here, which will be used in Section D.1.3.

Definition 48 (Bregman divergence)**.** *For (differentiable) regularizer r and z, w in its domain, the* Bregman divergence *from z to w is*

$$V_z^r(w) := r(w) - r(z) - \langle \nabla r(z), w - z\rangle.$$

Definition 49 (Proximal operator)**.** *For (differentiable) regularizer r, z in its domain, and g in the dual space (when the domain is in $\mathbb{R}^d$, so is the dual space), we define the* proximal operator *as*

$$\mathrm{Prox}_z^r(g) := \mathrm{argmin}_w \left\{\langle g, w\rangle + V_z^r(w)\right\}.$$

Several variables have specialized meaning throughout the remainder of this section. All graphs considered will be on $2n$ vertices with m edges, i.e. $m = n^2$. $\mathbf{A} \in \mathbb{R}^{2n\times m}$ is the edge-incidence matrix. d is the vectorized cost matrix C. b is the constraint vector, concatenating row and column constraints r, c. In algorithms for solving (D.4), x and y are primal (in a simplex) and dual (in a

box) variables respectively. In Section D.1.3, we adopt the linear programming perspective where the decision variable $x \in \Delta^m$ is a vector. In Section D.1.4, for convenience we take the perspective where $\mathbf{X}$ is an unflattened $n \times n$ matrix. $\mathcal{U}_{r,c}$ is the feasible polytope: when the domain is vectors, $\mathcal{U}_{r,c}$ is $x \mid \mathbf{A}x = b$, and when it is matrices, $\mathcal{U}_{r,c}$ is $\mathbf{X} \mid \mathbf{X}\mathbf{1} = r, \mathbf{X}^\top \mathbf{1} = c$ (by flattening $\mathbf{X}$ this is consistent).

D.1.3 Main algorithm

This section describes our algorithm for finding a primal-dual pair (x, y) with a small duality gap, with respect to the objective in (D.4), which we restate here for convenience:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} d^\top x + 2 \|d\|_\infty \left(y^\top \max -b^\top y\right), \ \mathcal{X} := \Delta^m, \ \mathcal{Y} := [-1, 1]^{2n}. \qquad \text{(Restatement of (D.4))}$$

Our algorithm is a specialization of the algorithm in [483]. One of our technical contributions in this regard is an analysis of the algorithm which more closely relates it to the analysis of dual extrapolation [420], an algorithm for finding approximate saddle points with a more standard analysis. In the first part of this section, we give the algorithmic framework and convergence analysis. In the second part of this section, we provide analysis of an alternating minimization scheme for implementing steps of the procedure. The same procedure was used in [483] which claimed without proof the linear convergence rate of the alternating minimization; we hope the analysis will make the method more broadly accessible to the optimization community. We defer many proofs to Appendix D.1.5.

Finally, we remark that the dual extrapolation analysis in this section (based on [483]) is somewhat different than its earlier incarnation in Appendix A.4; the main difference is that we allow for the subproblems to be solved to a specified level of *relative function error*, rather than *regret*. This difference ends up being crucial in improving our parameter dependences in our semi-streaming solver of Section 5.3.

For an objective $F(x, y)$ convex in x and concave in y, the standard way to measure the duality gap is to define the *gradient operator* $g(x, y) = (\nabla_x F(x, y), -\nabla_y F(x, y))$, and show that for $z = (x, y)$ and any u on the product space, the *regret*, $\langle g(z), z - u \rangle$, is small. Correspondingly, we define

$$g(x, y) := \left(d + 2 \|d\|_\infty \mathbf{A}^\top y, \ 2 \|d\|_\infty (b - \mathbf{A}x)\right).$$

The dual extrapolation framework [420] requires a regularizer on the product space. The algorithm is simple to state; it takes two "mirror descent-like" steps each iteration, maintaining a state s_t in the dual space[5]. A typical setup is a Lipschitz gradient operator and a regularizer which is the sum of canonical strongly-convex regularizers in the norms corresponding to the product space $\mathcal{X}, \mathcal{Y}$. However, recent works have shown that this setup can be greatly relaxed and still obtain similar

[5] In this regard, it is more similar to the "dual averaging" or "lazy" mirror descent setup [98].

rates of convergence. In particular, [483] introduced the following definition.

Definition 50 (Area-convexity)**.** *Regularizer r is κ-area-convex with respect to operator g if for any points a, b, c in its domain,*

$$\kappa \left(r(a) + r(b) + r(c) - 3r\left(\frac{a+b+c}{3}\right)\right) \geq \langle g(b) - g(a), b - c\rangle . \tag{D.5}$$

Area-convexity is so named because $\langle g(b) - g(a), b - c\rangle$ can be viewed as measuring the "area" of the triangle with vertices a, b, c with respect to some Jacobian matrix. In the case of bilinear objectives, the left hand side in the definition of area-convexity is invariant to permuting a, b, c, whereas the sign of the right hand side can be flipped by interchanging a, c, so area-convexity implies convexity. However, it does not even imply the regularizer r is strongly-convex, a typical assumption for the convergence of mirror descent methods.

We state the algorithm for time horizon T; the only difference from [420] is a factor of 2 in defining s_{t+1}, i.e. adding a $1/2\kappa$ multiple rather than $1/\kappa$. We find it of interest to explore whether this change is necessary or specific to the analysis of [483].

Algorithm 71: $\bar{w} = \texttt{Dual-Extrapolation}(\kappa, r, g, T)$: Dual extrapolation with area-convex r.

Initialize $s_0 = 0$, let $\bar{z}$ be the minimizer of r;
for $t < T$ **do**
  $z_t \leftarrow \text{Prox}_{\bar{z}}^{r}(s_t)$;
  $w_t \leftarrow \text{Prox}_{\bar{z}}^{r}\left(s_t + \frac{1}{\kappa} g(z_t)\right)$;
  $s_{t+1} \leftarrow s_t + \frac{1}{2\kappa} g(w_t)$;
  $t \leftarrow t + 1$;
Return: $\bar{w} := \frac{1}{T}\sum_{t\in[T]} w_t$;

Lemma 249 (Dual extrapolation convergence)**.** *Suppose r is κ-area-convex with respect to g. Further, suppose for some u, $\Theta \geq r(u) - r(\bar{z})$. Then, the output $\bar{w}$ to Algorithm 71 satisfies*

$$\langle g(\bar{w}), \bar{w} - u\rangle \leq \frac{2\kappa\Theta}{T}.$$

In fact, by more carefully analyzing the requirements of dual extrapolation we have the following.

Corollary 59. *Suppose in Algorithm 71, the proximal steps are implemented with $\epsilon'/4\kappa$ additive error. Then, the upper bound of the regret in Lemma 249 is $2\kappa\Theta/T + \epsilon'$.*

We now state a useful second-order characterization of area-convexity involving a relationship between the Jacobian of g and the Hessian of r, which was proved in [483].

Theorem 82 (Second-order area-convexity, Theorem 1.6 in [483])**.** *For bilinear minimax objectives, i.e. whose associated operator g has Jacobian*

$$\mathbf{J} = \begin{pmatrix} 0 & \mathbf{M}^\top \\ -\mathbf{M} & 0 \end{pmatrix},$$

and for twice-differentiable r, if for all z in the domain,

$$\begin{pmatrix} \kappa \nabla^2 r(z) & -\mathbf{J} \\ \mathbf{J} & \kappa \nabla^2 r(z) \end{pmatrix} \succeq 0,$$

then r is 3κ-area-convex with respect to g.

Finally, we complete the outline of the algorithm by stating the specific regularizer we use, which first appeared in [483]. We then prove its 3-area-convexity with respect to g by using Theorem 82.

$$r(x, y) = 2\bar{\delta}_{\max} \left(10 \sum_{j \in [n]} x_j \log x_j + x^\top \mathbf{A}^\top (y^2) \right), \tag{D.6}$$

where (y^2) is entry-wise. To give some motivation for this regularizer, one ℓ_∞-strongly convex regularizer is $\frac{1}{2} \|y\|_2^2$, but over the ℓ_∞ ball, this regularizer has large range. The term $x^\top \mathbf{A}^\top (y^2)$ in (D.6) captures the curvature required for strong-convexity locally, but has a smaller range due to the restrictions on $x, \mathbf{A}$. The constants chosen were the smallest which satisfy the assumptions of the following Lemma 250. We note that similar intuition was used to develop the local coordinate descent method of Chapter 4.

Lemma 250 (Area-convexity of the Sherman regularizer)**.** *For the Jacobian $\mathbf{J}$ associated with the objective in* (D.4) *and the regularizer r defined in* (D.6)*, we have*

$$\begin{pmatrix} \nabla^2 r(z) & -\mathbf{J} \\ \mathbf{J} & \nabla^2 r(z) \end{pmatrix} \succeq 0.$$

We now give the proof of Theorem 81, requiring some claims in Appendix D.1.5 for the complexity of Algorithm 71. In particular, Appendix D.1.5 implies that although the minimizer to the proximal steps cannot be computed in closed form because of non-separability, a simple alternating scheme converges to an approximate-minimizer in near-constant time.

Proof of Theorem 81. The algorithm is Algorithm 71, using the regularizer r in (D.6). Clearly, in the feasible region the range of the regularizer is at most $20 \|d\|_\infty \log n + 4 \|d\|_\infty$, where the former summand comes from the range of entropy and the latter $\left\|A^\top\right\|_\infty = 2$. We may choose

$\Theta = O(\|d\|_\infty \log n)$ to be the range of r to satisfy the assumptions of Lemma 249, since for all u, $\langle \nabla r(\bar{z}), \bar{z} - u\rangle \leq 0 \Rightarrow V_{\bar{z}}^r(u) \leq r(u) - r(\bar{z})$.

By Theorem 82 and Lemma 250, r is 3-area-convex with respect to g. By Corollary 59, $T = 12\Theta/\epsilon$ iterations suffice, implementing each proximal step to $\epsilon/12$-additive accuracy. Finally, using Theorem 83 to bound this implementation runtime concludes the proof. □

D.1.4 Rounding to $\mathcal{U}_{r,c}$

We state the rounding procedure in [25] for completeness here, which takes a transport plan $\tilde{\mathbf{X}}$ close to $\mathcal{U}_{r,c}$ and transforms it into a plan which exactly meets the constraints and is close to $\tilde{\mathbf{X}}$ in ℓ_1, and then prove its correctness. Throughout $r(\mathbf{X}) := \mathbf{X}\mathbf{1}, c(\mathbf{X}) := \mathbf{X}^\top \mathbf{1}$.

Algorithm 72: $\hat{X} = \texttt{Rounding}(\tilde{\mathbf{X}}, r, c)$: Rounding to feasible polytope

$\mathbf{X}' \leftarrow \mathbf{diag}\left(\min\left(\frac{r}{r(\tilde{\mathbf{X}})}, 1\right)\right)\tilde{\mathbf{X}}$;
$\mathbf{X}'' \leftarrow \mathbf{X}'\mathbf{diag}\left(\min\left(\frac{c}{c(X')}, 1\right)\right)$;
$e_r \leftarrow r - \mathbf{1}^\top r(\mathbf{X}''), e_c \leftarrow c - \mathbf{1}^\top c(\mathbf{X}''), E \leftarrow \mathbf{1}^\top e_r$;
$\hat{\mathbf{X}} \leftarrow \mathbf{X}'' + \frac{1}{E} e_r e_c^\top$;
Return: $\hat{\mathbf{X}}$;

Theorem 80 (Rounding guarantee, Lemma 7 in [25])**.** *There is an algorithm which takes $\tilde{x}$ with $\|\mathbf{A}\tilde{x} - b\|_1 \leq \delta$ and produces $\hat{x}$ in $O(n^2)$ time, with*

$$\mathbf{A}\hat{x} = b, \|\tilde{x} - \hat{x}\|_1 \leq 2\delta.$$

Proof. The algorithm is Algorithm 72. We adopt the alternative view of $\tilde{x}$ as a $n \times n$ matrix $\tilde{\mathbf{X}}$ in the simplex, and define operations $r(\mathbf{X}) = \mathbf{X}\mathbf{1}, c(\mathbf{X}) = \mathbf{X}^\top\mathbf{1}$, recalling the first and last n entries of b are r, c, i.e. the row and column constraints. Recall we assume we have

$$\left\|r(\tilde{\mathbf{X}}) - r\right\|_1 + \left\|c(\tilde{\mathbf{X}}) - c\right\|_1 \leq \delta.$$

Clearly all operations in Algorithm 72 take $O(n^2)$ time. To explain briefly, $\mathbf{X}'$ is fixed so that its row sums are feasible (i.e. $\mathbf{X}'\mathbf{1} \leq r$) and $\mathbf{X}''$ is fixed so that its column sums are feasible. Further, entrywise $\mathbf{X}'' \leq \mathbf{X}' \leq \tilde{\mathbf{X}}$, so $\mathbf{X}''$ is feasible. We first bound

$$d := \left\|\mathbf{X}'' - \tilde{\mathbf{X}}\right\|_1 = \left(\sum_{i:r_i(\tilde{\mathbf{X}}) > r_i} r_i(\tilde{\mathbf{X}}) - r_i\right) + \left(\sum_{j:c_j(\mathbf{X}') > c_j} c_j(\mathbf{X}') - c_j\right).$$

Note $\left\|r(\tilde{\mathbf{X}}) - r\right\|_1 \geq \sum_{i:r_i(\tilde{\mathbf{X}})>r_i} r_i(\tilde{X}) - r_i$. Further, by $\mathbf{X}' \leq \tilde{\mathbf{X}}$ entrywise,

$$\sum_{j:c_j(\mathbf{X}')>c_j} c_j(\mathbf{X}') - c_j \leq \left\|c(\tilde{\mathbf{X}}) - c\right\|_1.$$

Thus $d \leq \delta$. $\hat{\mathbf{X}} \in \mathcal{U}_{r,c}$, since $e_r, e_c \geq 0$ and $\mathbf{1}^\top e_r = \mathbf{1}^\top e_c = e$, so $\hat{\mathbf{X}}\mathbf{1} = r$, $\hat{\mathbf{X}}^\top \mathbf{1} = c$. Also,

$$\left\|\hat{\mathbf{X}} - \tilde{\mathbf{X}}\right\|_1 \leq \left\|\mathbf{X}'' - \tilde{\mathbf{X}}\right\|_1 + \left\|\hat{\mathbf{X}} - \mathbf{X}''\right\|_1 \leq \delta + e.$$

Finally,

$$e = 1 - \mathbf{1}^\top \mathbf{X}'' \mathbf{1} = 1 - \left(\mathbf{1}^\top \tilde{\mathbf{X}} \mathbf{1} - d\right) = d.$$

Thus using $d \leq \delta$ proves the claim. □

D.1.5 Missing proofs from Section D.1.3

In this section, we state missing proofs from Section D.1.3. We will also provide the efficient implementation of the proximal steps required by Algorithm 71.

Proof of Lemma 249. Our first step is to prove the following inequality:

$$\frac{1}{2\kappa}\langle g(w_t), w_t - \bar{z}\rangle \leq \langle s_{t+1}, z_{t+1} - \bar{z}\rangle + V_{\bar{z}}^r(z_{t+1}) - \langle s_t, z_t - \bar{z}\rangle - V_{\bar{z}}^r(z_t). \tag{D.7}$$

Let $c_t = \frac{z_t + w_t + z_{t+1}}{3}$. The proof follows from minimality of z_t with respect to c_t, minimality of w_t with respect to z_{t+1}, and area-convexity (D.5) with respect to z_t, w_t, and z_{t+1}. Respectively,

$$\begin{aligned} \langle s_t, z_t\rangle + r(z_t) &\leq \langle s_t, c_t\rangle + r(c_t) \\ \langle s_t, w_t\rangle + \frac{1}{\kappa}\langle g(z_t), w_t\rangle + r(w_t) &\leq \langle s_t, z_{t+1}\rangle + \frac{1}{\kappa}\langle g(z_t), z_{t+1}\rangle + r(z_{t+1}) \\ \frac{1}{\kappa}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle &\leq r(z_t) + r(w_t) + r(z_{t+1}) - 3r\left(c_t\right). \end{aligned} \tag{D.8}$$

Adding three times the first equation to the third, rearranging, and using the definition of c_t, we have

$$\frac{1}{\kappa}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle \leq r(w_t) + r(z_{t+1}) - 2r(z_t) + \langle s_t, w_t + z_{t+1} - 2z_t\rangle.$$

Rearranging the second equation, we have

$$\frac{1}{\kappa}\langle g(z_t), w_t - z_{t+1}\rangle \leq r(z_{t+1}) - r(w_t) + \langle s_t, z_{t+1} - w_t\rangle.$$

Adding these two equations, we have

$$\frac{1}{\kappa}\left\langle g(w_t), w_t - z_{t+1}\right\rangle \leq 2r(z_{t+1}) - 2r(z_t) + \left\langle s_t, 2z_{t+1} - 2z_t\right\rangle.$$

Dividing by 2 and adding $\frac{1}{2\kappa}\left\langle g(w_t), z_{t+1} - \bar{z}\right\rangle$ to both sides, we obtain the desired (D.7). Now, define the potential function

$$\Phi_k = \frac{1}{2\kappa}\sum_{t=0}^{k-1}\left\langle g(w_t), w_t - \bar{z}\right\rangle - \left\langle s_k, z_k - \bar{z}\right\rangle - V_{\bar{z}}^r(z_k)$$

Then, by (D.7), Φ_k is nonincreasing in k. Therefore for any u, by the definition of Θ,

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\left\langle g(w_t), w_t - u\right\rangle &\leq \frac{1}{T}\sum_{t=0}^{T-1}\left\langle g(w_t), w_t - \bar{z}\right\rangle + \frac{1}{T}\sum_{t=0}^{T-1}\left\langle g(w_t), \bar{z} - u\right\rangle + \left(\frac{2\kappa\Theta}{T} - \frac{2\kappa V_{\bar{z}}(u)}{T}\right) \\
&\leq \frac{1}{T}\sum_{t=0}^{T-1}\left\langle g(w_t), w_t - \bar{z}\right\rangle + \frac{1}{T}\sum_{t=0}^{T-1}\left\langle g(w_t), \bar{z} - z_T\right\rangle + \left(\frac{2\kappa\Theta}{T} - \frac{2\kappa V_{\bar{z}}(z_T)}{T}\right) \\
&= \frac{2\kappa}{T}\Phi_T + \frac{2\kappa\Theta}{T} \leq \frac{2\kappa}{T}\Phi_0 + \frac{2\kappa\Theta}{T} = \frac{2\kappa\Theta}{T}.
\end{aligned}$$

The inequality on the second line used the definition of $z_T = \operatorname{Prox}_{\bar{z}}^r\left(\frac{1}{2\kappa}\sum_{t\in[T-1]} g(w_t)\right)$, and the last inequality is $\Phi_T \leq \Phi_0$. The conclusion follows from the definition of g (because it is linear). □

Proof of Corollary 59. We see that (D.7) now holds up to $\frac{\epsilon'}{2\kappa}$ additive error, so that Φ_k is increasing by at most $\frac{\epsilon'}{2\kappa}$ each step. Thus, we obtain $\Phi_T \leq \Phi_0 + \frac{T\epsilon'}{2\kappa}$, yielding the conclusion. □

Proof of Lemma 250. We scale both r and $\mathbf{J}$ down by $2\left\|d\right\|_\infty$, which does not affect positive-semidefiniteness. By computation we have (recalling all columns of $\mathbf{A}$ have ℓ_1 norm of 2)

$$\nabla^2 r(x,y) = \begin{pmatrix} 5\left\|\mathbf{A}_{:j}\right\|_1 \mathbf{diag}\left(\frac{1}{x_j}\right) & 2\mathbf{A}^\top \mathbf{diag}\left(y_i\right) \\ 2\mathbf{diag}\left(y_i\right)\mathbf{A} & 2\mathbf{diag}\left(\mathbf{A}_i^\top x\right) \end{pmatrix}.$$

It suffices to show that for any vector $\begin{pmatrix} a & b & c & d \end{pmatrix}$ we have

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 5\left\|\mathbf{A}_{:j}\right\|_1 \mathbf{diag}\left(\frac{1}{x_j}\right) & 2\mathbf{A}^\top \mathbf{diag}\left(y_i\right) & 0 & -\mathbf{A}^\top \\ 2\mathbf{diag}\left(y_i\right)\mathbf{A} & 2\mathbf{diag}\left(\mathbf{A}_i^\top x\right) & A & 0 \\ 0 & \mathbf{A}^\top & 5\left\|\mathbf{A}_{:j}\right\|_1 \mathbf{diag}\left(\frac{1}{x_j}\right) & 2\mathbf{A}^\top \mathbf{diag}\left(y_i\right) \\ -\mathbf{A} & 0 & 2\mathbf{diag}\left(y_i\right)\mathbf{A} & 2\mathbf{diag}\left(\mathbf{A}_i^\top x\right) \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

is nonnegative. Upon simplifying and gathering like terms, it suffices to show

$$\sum_{i,j} \mathbf{A}_{ij} \left(\frac{5a_j^2}{x_j} + 4a_j b_i y_i + 2b_i^2 x_j - 2a_j d_i + 2c_j b_i + \frac{5c_j^2}{x_j} + 4c_j d_i y_i + 2d_i^2 x_j \right) \geq 0.$$

However, this is true for $y_i \in [-1, 1]$, since each coefficient groups into clearly nonnegative terms,

$$\left(\frac{4a_j^2}{x_j} + 4a_j b_i y_i + b_i^2 x_j \right) + \left(\frac{a_j^2}{x_j} - 2a_j d_i + d_i^2 x_j \right)$$
$$+ \left(\frac{4c_j^2}{x_j} + 4c_j d_i y_i + d_i^2 x_j \right) + \left(\frac{c_j^2}{x_j} + 2c_j b_i + b_i^2 x_j \right).$$

□

Alternating minimization analysis

We now give the convergence analysis of an alternating minimization procedure for minimizing a function of the form (throughout this section, $r(x, y)$ is as in (D.6))

$$f(x, y) := \langle \xi, x \rangle + \langle \eta, y \rangle + r(x, y) \tag{D.9}$$

which is the type of minimization problem arising from steps of the form $\text{Prox}_{\tilde{z}}^{r}(g)$. As we will see, $f(x, y)$ is jointly convex. Throughout this section, let $x_{\text{OPT}}, y_{\text{OPT}}$ be the minimizer to f. Corollary 59 states that $O(\epsilon)$ additive error to f gives the same asymptotic convergence rate in Algorithm 71. We will show that a simple alternating minimization scheme enjoys a linear rate of convergence in our setting; thus, roughly $O(\log \epsilon^{-1})$ iterations suffice. We first give a proof of a general condition which suffices for linear convergence.

Lemma 251. *Suppose $f(x, y)$ is twice-differentiable and jointly convex, over the product space $\mathcal{X} \times \mathcal{Y}$. Consider the alternating minimization scheme,*

1. $x_{k+1} := \text{argmin}_{x \in \mathcal{X}} f(x, y_k)$
2. $y_{k+1} := \text{argmin}_{y \in \mathcal{Y}} f(x_{k+1}, y)$

Further, suppose there are convex regions $\mathcal{X}_{k+1} \subseteq \mathcal{X}$, $\mathcal{Y}_k \subseteq \mathcal{Y}$ which contain x_{k+1}, y_k respectively, such that for any $x' \in \mathcal{X}_{k+1}$, $y', y'' \in \mathcal{Y}_k$, and for some $\sigma \geq 1$,

$$\nabla^2 f(x', y') \succeq \frac{1}{\sigma} \nabla_{yy}^2 f(x_{k+1}, y''), \tag{D.10}$$

where ∇_{yy}^2 is the Hessian with all but the yy block zeroed out. Then, for any $x^* \in \mathcal{X}_{k+1}$, $y^* \in \mathcal{Y}_k$,

$$f(x_{k+1}, y_k) - f(x_{k+1}, y_{k+1}) \geq \frac{1}{\sigma} \left(f(x_{k+1}, y_k) - f(x^*, y^*) \right).$$

Proof. Let $\tilde{y} = \left(1 - \frac{1}{\sigma}\right) y_k + \frac{1}{\sigma} y^*$. We will prove instead that

$$f(x_{k+1}, y_k) - f(x_{k+1}, \tilde{y}) \geq \frac{1}{\sigma} \left(f(x_{k+1}, y_k) - f(x^*, y^*) \right),$$

from which the conclusion will follow since $f(x_{k+1}, y_{k+1}) \leq f(x_{k+1}, \tilde{y})$. Note by definition of $\tilde{y}$, as well as optimality of x_{k+1} which implies $0 \geq \langle \nabla_x f(x_{k+1}, y_k), x_{k+1} - x^* \rangle$,

$$\langle \nabla_y f(x_{k+1}, y_k), y_k - \tilde{y} \rangle = \frac{1}{\sigma} \langle \nabla_y f(x_{k+1}, y_k), y_k - y^* \rangle \geq \frac{1}{\sigma} \left\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^* \right\rangle \tag{D.11}$$

where $z_{k+\frac{1}{2}} := (x_{k+1}, y_k)$ and $z^* := (x^*, y^*)$. Further, let $y_\alpha := (1-\alpha)y_k + \alpha y^*$, $\tilde{y}_\alpha := (1-\alpha)y_k + \alpha\tilde{y}$, and $x_\alpha := (1-\alpha)x_{k+1} + \alpha x^*$. Then, by Taylor expansion we have $f(x_{k+1}, y_k) - f(x_{k+1}, \tilde{y})$ equals

$$\begin{aligned}
&\langle \nabla_y f(x_{k+1}, y_k), y_k - \tilde{y} \rangle - \int_0^1 \int_0^\beta (\tilde{y} - y_k)^\top \nabla^2_{yy} f(x_{k+1}, \tilde{y}_\alpha)(\tilde{y} - y_k) d\alpha d\beta \\
&\geq \frac{1}{\sigma} \left\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^* \right\rangle - \frac{1}{\sigma^2} \int_0^1 \int_0^\beta (y^* - y_k)^\top \nabla^2_{yy} f(x_{k+1}, \tilde{y}_\alpha)(y^* - y_k) d\alpha d\beta \\
&\geq \frac{1}{\sigma} \left(\left\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^* \right\rangle - \int_0^1 \int_0^\beta (z^* - z_{k+\frac{1}{2}})^\top \nabla^2 f(x_\alpha, y_\alpha)(z^* - z_{k+\frac{1}{2}}) d\alpha d\beta \right) \\
&= \frac{1}{\sigma} \left(f(x_{k+1}, y_k) - f(x^*, y^*) \right).
\end{aligned}$$

In the first inequality, we used (D.11) and the definition of $\tilde{y}$, and in the second we used (D.10) (since $x_\alpha \in \mathcal{X}_{k+1}, y_\alpha, \tilde{y}_\alpha \in \mathcal{Y}_k$ by convexity). □

We now give a helper lemma specialized to the particular f in (D.9), which will be used in the proof of convergence.

Lemma 252. *For some x_{k+1}, y_k, let $\mathcal{X}_{k+1} = \left\{ x \mid x \geq \frac{1}{2} x_{k+1} \right\}$ where the inequality is entrywise, and let $\mathcal{Y}_k$ be the entire domain of y (i.e. $\mathcal{Y}$). Then for any $x' \in \mathcal{X}_{k+1}, y', y'' \in \mathcal{Y}_k$,*

$$\nabla^2 r(x', y') \succeq \frac{1}{12} \nabla^2_{yy} r(x_{k+1}, y'').$$

Proof. Recall that (since $\|\mathbf{A}_{:j}\|_1 = 2$)

$$\nabla^2 r(x, y) = 2 \|d\|_\infty \begin{pmatrix} 5 \|\mathbf{A}_{:j}\|_1 \operatorname{\mathbf{diag}}\left(\frac{1}{x_j}\right) & 2\mathbf{A}^\top \operatorname{\mathbf{diag}}(y_i) \\ 2\operatorname{\mathbf{diag}}(y_i)\, \mathbf{A} & 2\operatorname{\mathbf{diag}}\left(\mathbf{A}_i^\top x\right) \end{pmatrix}.$$

Consider the diagonal approximation

$$\mathbf{D}(x) = 2 \|d\|_\infty \begin{pmatrix} \|\mathbf{A}_{:j}\|_1 \operatorname{\mathbf{diag}}\left(\frac{1}{x_j}\right) & 0 \\ 0 & \operatorname{\mathbf{diag}}\left(\mathbf{A}_i^\top x\right) \end{pmatrix}.$$

We claim for any y,

$$\mathbf{D}(x) \preceq \nabla^2 r(x,y) \preceq 6\mathbf{D}(x). \tag{D.12}$$

To see this, consider the quadratic forms with respect to some vector $\begin{pmatrix} u & v \end{pmatrix}$:

$$\begin{pmatrix} u & v \end{pmatrix} \nabla^2 r(x,y) \begin{pmatrix} u \\ v \end{pmatrix} = 2\|d\|_\infty \sum_{i,j} \mathbf{A}_{ij} \left(\frac{5u_j^2}{x_j} + 4u_j v_i y_i + 2v_i^2 x_j \right),$$
$$\begin{pmatrix} u & v \end{pmatrix} \mathbf{D}(x) \begin{pmatrix} u \\ v \end{pmatrix} = 2\|d\|_\infty \sum_{i,j} \mathbf{A}_{ij} \left(\frac{u_j^2}{x_j} + v_i^2 x_j \right).$$

Now (D.12) follows because for any $y_i \in [-1,1]$, it's easy to verify

$$\frac{u_j^2}{x_j} + v_i^2 x_j \le \frac{5u_j^2}{x_j} + 4u_j v_i y_i + 2v_i^2 x_j \le 6\left(\frac{u_j^2}{x_j} + v_i^2 x_j \right).$$

Therefore, to prove the lemma statement we can use

$$\nabla^2 r(x',y') \succeq \mathbf{D}(x') \succeq \frac{1}{2}\mathbf{D}(x_{k+1}) \succeq \frac{1}{12}\nabla^2_{yy} r(x_{k+1}, y'').$$

The inequality $\mathbf{D}(x') \succeq \frac{1}{2}\mathbf{D}(x_{k+1})$ followed from the definition of $\mathcal{X}_{k+1}$, and the last inequality followed from $\mathbf{D}(x_{k+1})$ spectrally dominating $\frac{1}{6}\nabla^2 r(x_{k+1}, y'')$, and restrictions of $\mathbf{D}(x_{k+1})$ to the yy block can only decrease the quadratic form. □

We now give the proof of the linear rate of convergence.

Lemma 253. *For $f(x,y)$ defined in* (D.9)*, the alternating minimization scheme*

1. $x_{k+1} := \operatorname{argmin}_{x \in \mathcal{X}} f(x, y_k)$.
2. $y_{k+1} := \operatorname{argmin}_{y \in \mathcal{Y}} f(x_{k+1}, y)$.

decreases the function error $f(x_k, y_k) - f(x_{\mathrm{OPT}}, y_{\mathrm{OPT}})$ by a factor of at least $1/24$ in each iteration.

Proof. We can apply Lemma 251 with the sets defined in Lemma 252, with $\sigma = 12$. On iteration k, consider picking the points $x^*, y^* = \frac{1}{2}(x_{k+1} + x_{\mathrm{OPT}}), \frac{1}{2}(y_k + y_{\mathrm{OPT}})$. Evidently, $x^* \in \mathcal{X}_{k+1}, y^* \in \mathcal{Y}_k$. Therefore, since $f(x_{k+1}, y_{k+1}) \ge f(x_{k+2}, y_{k+1})$,

$$f(x_{k+1}, y_k) - f(x_{k+2}, y_{k+1}) \ge f(x_{k+1}, y_k) - f(x_{k+1}, y_{k+1}) \ge \frac{1}{12}(f(x_{k+1}, y_k) - f(x^*, y^*)).$$

Furthermore, by convexity, we have

$$f(x_{k+1}, y_k) - f(x^*, y^*) \ge \frac{1}{2}(f(x_{k+1}, y_k) - f(x_{\mathrm{OPT}}, y_{\mathrm{OPT}})).$$

Finally, combining these two inequalities and rearranging,

$$\frac{23}{24}(f(x_{k+1}, y_k) - f(x_{\text{OPT}}, y_{\text{OPT}})) \geq f(x_{k+2}, y_{k+1}) - f(x_{\text{OPT}}, y_{\text{OPT}}).$$

Thus, by taking a y step and then an x step, we decrease the function error by a $1/24$ factor. □

Finally, we show that steps of the alternating minimization can be implemented in linear time.

Lemma 254. *For $f(x, y)$ defined in* (D.9)*, we can implement the steps*

1. $x_{k+1} := \text{argmin}_x f(x, y_k)$.
2. $y_{k+1} := \text{argmin}_y f(x_{k+1}, y)$.

restricted to the relevant domains, in time $O(n^2)$.

Proof. Recall $\mathbf{A}$ has n^2 nonzero entries, so a matrix-vector multiplication can be performed in this time. Computing x in linear time is straightforward: it is defined by

$$\text{argmin}_x \langle \gamma, x \rangle + \sum_{j \in [n]} x_j \log x_j \text{ such that } x \in \Delta^m, \gamma := \frac{1}{20 \|d\|_\infty} \xi + \frac{1}{10} \mathbf{A}^\top (y^2).$$

By examining the KKT conditions, it is clear that the minimizing x is proportional to $\exp(-\gamma)$; computing γ takes $O(n^2)$ time, as does the simplex projection. Similarly, computing y in linear time is simple for fixed x: it is

$$\text{argmin}_y \langle \eta, y \rangle + \left\langle 2 \|d\|_\infty \mathbf{A} x, y^2 \right\rangle \text{ such that } y \in [-1, 1]^{2n},$$

which is coordinate-wise decomposable as minimizing a quadratic over an interval. □

Theorem 83 (Complexity of alternating minimization)**.** *We can obtain an $\epsilon/2$-approximate minimizer to the proximal steps required by Algorithm 71 to $\epsilon/2$ accuracy, with the regularizer of* (D.6) *and $\kappa = 3$, in $O(\log \gamma)$ parallelizable iterations for $\gamma = \log n \cdot \|d\|_\infty \cdot \epsilon^{-1}$, and $O(n^2 \log \gamma)$ total work.*

Proof. By Lemmas 253 and 254, we can spend $O(n^2)$ parallelizable work to decrease the suboptimality gap by a $1/24$ factor, so it remains to argue that the initial error is at most $\text{poly}(\log n, \|d\|_\infty, \epsilon^{-1})$ to show that implementing the proximal steps to additive error $\epsilon/2$ can be done in $O(\log \gamma)$ iterations. We show that this is true for implementing the proximal step for z_t; a similar argument holds for w_t. To this end, note that by our setting of κ, for any z where we let $g(z) = (g^x(z), g^y(z))$,

$$\frac{1}{2\kappa} \|g^x(z)\|_\infty = \frac{1}{6} \left\| d + 2 \|d\|_\infty \mathbf{A}^\top y \right\|_\infty \leq \frac{\|d\|_\infty}{2},$$
$$\frac{1}{2\kappa} \|g^y(z)\|_1 = \frac{1}{6} \|2 \|d\|_\infty (b - \mathbf{A} x)\|_1 \leq \frac{4 \|d\|_\infty}{3}.$$

Therefore, for $s_t = (s_t^x, s_t^y)$, by the triangle inequality, and $t \le 12\Theta/\epsilon$ the bound on the number of steps required where Θ is the range of r, we have

$$\|s_t^x\|_\infty \le t \cdot \frac{1}{2\kappa} \|g^x(z)\|_\infty \le \frac{6\|d\|_\infty \Theta}{\epsilon},$$
$$\|s_t^y\|_1 \le t \cdot \frac{1}{2\kappa} \|g^y(z)\|_1 \le \frac{16\|d\|_\infty \Theta}{\epsilon}.$$

A simple calculation yields $\Theta = 20\|d\|_\infty \log n + 4\|d\|_\infty$ upper bounds the range of r. Finally, let x_t^*, y_t^* be the minimizer of the proximal objective,

$$\langle s_t^x, x\rangle + \langle s_t^y, y\rangle + r(x, y).$$

For any initialization $x_{\text{init}}, y_{\text{init}}$ to the alternating minimization, the suboptimality gap is given by

$$\langle s_t^x, x_{\text{init}} - x_t^*\rangle + \langle s_t^y, y_{\text{init}} - y_t^*\rangle + r(x_{\text{init}}, y_{\text{init}}) - r(x_t^*, y_t^*)$$
$$\le \|x_{\text{init}} - x_t^*\|_1 \|s_t^x\|_\infty + \|y_{\text{init}} - y_t^*\|_\infty \|s_t^y\|_1 + \Theta \le \left(\frac{44\|d\|_\infty}{\epsilon} + 1\right)\Theta.$$

Therefore, the total number of iterations required is bounded by $24\log\left(\left(\frac{88\|d\|_\infty}{\epsilon^2} + \frac{2}{\epsilon}\right)\Theta\right)$ as desired. □

Numerical precision. We also make a brief comment on bit-complexity issues which may arise when scaling exponentials. In particular, each of our alternating minimization steps of the form

$$\langle g, x\rangle + \sum_j x_j \log x_j \tag{D.13}$$

require exponentiating a potentially large vector $\log x - g$, and rescaling the vector to be on the simplex. The following lemma shows that we can implement this step with $O(\log n)$ bit-complexity, with a small polynomial (say n^{-90}) loss in the objective value. Because we may assume that ϵ^{-1} is bounded by say, n^2, else an interior point method achieves our stated runtime, the cumulative loss in objective value over all iterations due to limited precision is significantly less than ϵ, and does not affect our asymptotic convergence rate. More precisely, we maintain x implicitly through a vector v which is $\log x$ up to a scaling, and show that by truncating v to have its range of coordinates bounded by $O(\log n)$, the resulting simplex variable remains a high-precision minimizer to (D.13).

Lemma 255. *Let $v \in \mathbb{R}^m$, and let $x \in \Delta^m$ be such that $x \propto \exp(v)$. Consider the following operation: let $j^* = \text{argmin}_j v_j$, and j' be such that $v_{j'} < v_{j^*} - 100\log n$. Set $\hat{v} = v$ in every*

coordinate, except $\hat{v}_{j'} \leftarrow v_{j^*} - 100 \log n$. *Then, for* $\hat{x} \propto \exp(\hat{v})$ *in the simplex,*

$$\sum_j \hat{x}_j \log \hat{x}_j - \sum_j x_j \log x_j < n^{-95}, \langle g, \hat{x} - x \rangle < \|g\|_\infty n^{-95}.$$

Proof. Clearly, $\|\exp(v)\|_1 < \|\exp(\hat{v})\|_1$, since exp is monotone. Moreover,

$$\|\exp(\hat{v})\|_1 - \|\exp(v)\|_1 = \exp(v_{j^*} - 100 \log n) - \exp(v_{j'}) < n^{-100} \exp(v_{j^*}) < n^{-100} \|\exp(v)\|_1 .$$

Now, for every coordinate $j \neq j'$, this implies that $(1 - n^{-100}) x_j < \hat{x}_j < x_j$. Thus,

$$\hat{x}_j \log \hat{x}_j - x_j \log x_j < -n^{-100} x_j \log x_j.$$

Furthermore, we have

$$\hat{x}_{j'} \log \hat{x}_{j'} - x_{j'} \log x_{j'} < -x_{j'} \log x_{j'} < 100 n^{-100},$$

by $-x \log x$ is increasing for small x and $x_{j'}$ is bounded by n^{-100}. Combining these estimates,

$$\sum_j \hat{x}_j \log \hat{x}_j - \sum_j x_j \log x_j < (100 + \log n) n^{-100} < n^{-95}$$

for $n > 10$. Similarly,

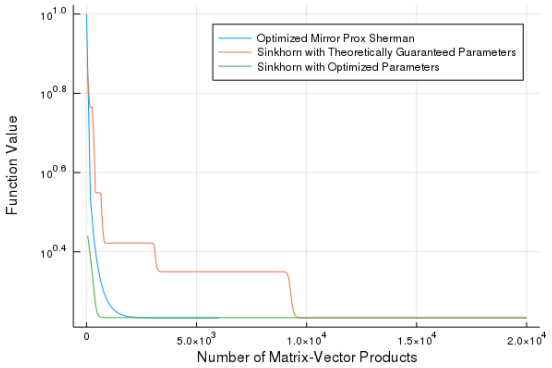
$$\langle g, \hat{x} - x \rangle \leq \|g\|_\infty \left(\hat{x}_{j'} + \sum_{j \neq j'} n^{-100} x_j \right) \leq \|g\|_\infty n^{-95}$$

for $n > 10$. □

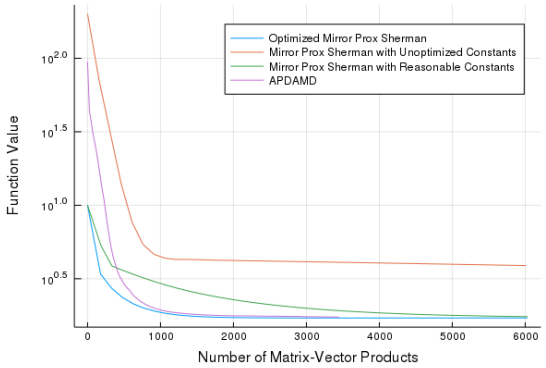
Thus, repeatedly applying Lemma 255 every time we need to truncate a coordinate of v due to finite bit precision, over the course of all iterations of the algorithm and all alternating minimization steps, the error incurred is negligible compared to the desired accuracy ϵ (where we also note $\|g\|_\infty$ for all g we encounter is bounded by a small polynomial in n).

D.1.6 Experiments

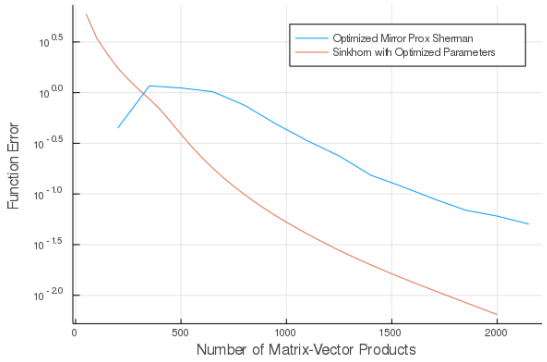
We show experiments illustrating the potential of our algorithm to be useful in practice, by considering its performance on computing optimal transport distances on the MNIST dataset and comparing against algorithms in the literature including APDAMD [365] and Sinkhorn iteration. All comparisons are based on the number of matrix-vector multiplications (rather than iterations, due to our algorithm's alternating subroutine), the main computational component of all algorithms considered.



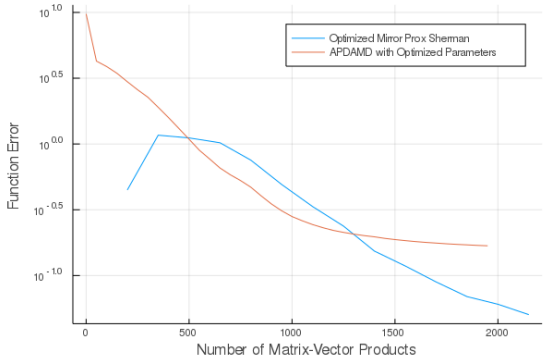
(a) Comparison with Sinkhorn iteration with different parameters.



(b) Comparison with APDAMD [365] with different parameters.



(a) Comparison with Sinkhorn iteration on 20 randomly chosen MNIST digit pairs.



(b) Comparison with APDAMD [365] on 20 randomly chosen MNIST digit pairs.

While our unoptimized algorithm performs poorly, slightly optimizing the size of the regularizer and step sizes used results in an algorithm with competitive performance to APDAMD, the first-order method with the best provable guarantees and observed practical performance. Sinkhorn iteration outperformed all first-order methods experimentally; however, an optimized version of our algorithm performed better than conservatively-regularized Sinkhorn iteration, and was more competitive with variants of Sinkhorn found in practice than other first-order methods.

As we discuss in our implementation details, we acknowledge that implementations of our algorithm illustrated are not the same as those with provable guarantees in our paper. However, we believe that our modifications are justifiable in theory, and consistent with those made in practice to existing algorithms. Further, we hope that studying the modifications we made (step size, using mirror prox [415] for stability considerations), as well as the consideration of other numerical speedups such as greedy updates [25] or kernel approximations [24], will become fruitful for understanding the potential of accelerated first-order methods in both the theory and practice of computational optimal transport.

Implementation details.

Dataset. For the first two figures, we had the following experimental setup. We randomly sampled a pair of digits from the MNIST dataset corresponding to the digit 1, and added a small amount of background noise for numerical stability, as is standard in the literature [25]. We downsampled the 28×28 pixel images to size 14×14 by skipping every other pixel to speed up experiments. Similar performances were observed across multiple random instances. Finally, the cost metric used was by Manhattan distance on the 2-dimensional grid.

For the second pair of figures, we randomly sampled 20 pairs of digits from the MNIST dataset where each pair corresponds to the same digit. As before we added a small amount of background noise for stability. As opposed to the previous comparison we ran all three algorithms on the true 28×28 pixel images. For each of the digit pairs we ran the Sinkhorn algorithm to high precision to obtain a baseline solution for comparison, and for each of the three algorithms tested we compared the value of the solutions obtained to this baseline. The plots compare the number of matrix-vector multiplies to the objective value error averaged over all 20 digit pairs. The metric is again the Manhattan distance over the 2-dimensional grid.

Objective value. For simplicity, in all cases we measured objective value by the overestimate presented in (D.4). By the proof of Lemma 247, this is an overestimate to the true objective after performing the rounding procedure in Algorithm 72. In practice, we observed that this overestimate was negligibly different from the objective after rounding.

Sinkhorn implementation details. We implemented the standard Sinkhorn algorithm, using different settings of η^{-1}. Sinkhorn iteration converges to an ϵ-approximate transportation plan in theory when η is very large, roughly $\log n/\epsilon$. However, in practice, it is observed that much smaller values of η suffice for rapid convergence. We tracked the convergence of Sinkhorn iteration for $\eta = 70$ and $\eta = 5$, which we considered close to a theoretically guaranteed parameter and a much less conservative practical parameter, respectively. The optimized Sinkhorn algorithm converged at rates much faster than the predicted ϵ^{-2} rate on all experiments, outperforming all other methods, which we believe merits further investigation. Significantly larger values of η led to numerical stability issues when computing $\exp(-\eta \mathbf{C})$.

APDAMD implementation details. We implemented the APDAMD algorithm (Algorithm 4 in [365]), with the quadratic regularizer (i.e. $\frac{1}{2\gamma} \|\lambda\|_2^2$). We observed that the amount of the quadratic regularizer added did not affect the practical convergence of the algorithm. A simple reason for this is because the algorithm builds in a more aggressive step-size strategy, because the pessimistic $\gamma = O(n)$ is often too conservative to be necessary in practice. The figure tracks APDAMD convergence with $\eta = 10^{-2}, \epsilon = 10^{-3}$.

Mirror prox. For numerical stability considerations, we implemented our algorithm as an instance of mirror prox [415], another extragradient method which takes local iterations rather than accumulating a dual operator and taking steps with respect to some $\bar{z}$ (i.e. dual extrapolation).

Although there is not a known proof of mirror prox convergence with an area-convex regularizer, we find this decision reasonable for several reasons. In general, variations of entropic mirror descent are well-known to be equivalent to their dual averaging versions; it is likely that a similar equivalence can be drawn between mirror prox and extragradient dual averaging, i.e. dual extrapolation. Furthermore, the standard proofs of dual extrapolation and mirror prox are quite similar; we believe it is likely that area-convexity results in convergence for mirror prox, although this merits further investigation.

Termination. We terminated our alternating minimization procedure when the movement of iterations in ℓ_1 was negligible. Typically, we observed that 3-5 alternating steps sufficed for convergence.

Step sizes. We varied two parameters in our experiments: the step size $\frac{1}{\kappa}$ used in our extragradient algorithm, and the amount of entropy used in our regularizer (in the paper, we used 10 times entropy compared to the quadratic component $x^\top \mathbf{A}^\top (y^2)$). One reason this may be reasonable in practice is similar to the observed behavior of the Sinkhorn iteration tuning the η^{-1} parameter, and APDAMD performing a more-aggressive line search for the observed amount of regularizer necessary. We note that the need to tune the amount of entropy used in the regularizer is likely due to the analysis not being tight in the constants, and with a tighter analysis, it may not be necessary to tune this parameter. To this end, we plotted the performance of three algorithm settings.

- In the "unoptimized constants", we set the constants to roughly those with theoretical guarantees, i.e. 10 times entropy and step size 1.
- In the "reasonably optimized constants", we set the amount of entropy to be 4, and the step size to be $\|d\|_\infty /3$, to offset the $\|d\|_\infty$ multiple of the regularizer used in our iterations. For smaller values of ϵ, these settings compared favorably with APDAMD.
- In the "optimized constants", we set the amount of entropy at 3, and the step size at $\|d\|_\infty$. This setting outperformed APDAMD and was more competitive with Sinkhorn iteration.

Discussion. We believe multiple interesting avenues of exploration arise from our experiments.

- Sinkhorn with aggressively chosen η outperformed all other methods we benchmarked against, and converged at rates faster than suggested by its known analyses. It may prove fruitful to study if further assumptions about practical instances explain this discrepancy.
- Directly accelerated methods such as APDAMD also exhibit ϵ^{-1} convergence rates, at the cost of a worse dependence on dimension. However, this worst-case dependence can be mitigated if the instance is favorable in practice, i.e. by choosing $\gamma \approx O(1)$. This was observed to be the case in our experiments for the MNIST dataset. It is interesting to see if a similar adaptive tuning applies to our method with provable guarantees.

- Our method did not exhibit instability when changing the amount of entropy in the regularizer, but it did exhibit vastly-improved convergence. It is possible that the amount of regularizer needed is not quite so large, perhaps through a more careful analysis.
- We did not benchmark against the greedy Sinkhorn method of [25], or consider numerical speedups such as those in [24]. It remains open to explore if these practical speedups are applicable to first-order methods such as ours as well.

D.1.7 Deferred proofs from Section 5.3

In this section, we give a proof of Proposition 14, which we restate for convenience. We recall the following definition: for convex-concave $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, we say a pair (x, y) is an ϵ-approximate saddle point if $f(x, y') - f(x', y) \leq \epsilon \ \forall x' \in \mathcal{X},\ y' \in \mathcal{Y}$. Before embarking on the proof, we require the following helper lemma for demonstrating correctness of Line 2 of Algorithm 13.

Lemma 256. *Consider* (5.1) *under the transformation described by Lines 1 and 2 of Algorithm 13. The approximate optimality of any point in Δ^m for the corresponding problem* (5.2) *is unaffected by this transformation (i.e. if some point is ϵ-approximately suboptimal for the new problem, it also has ϵ-approximate optimality for the old problem). Moreover, using $O(n)$ space and one pass we can implement this transformation for the remainder of any semi-streaming algorithm under the model of Definition 18.*

Proof. We discuss correctness of each line separately.

Correctness of Line 1. Let $i^* = \text{argmin}_{i \in [m]} c_i$. We claim that the minimizer of $\langle c, x \rangle + \left\| \mathbf{A}^\top x - b \right\|_1$ only puts nonzero values on coordinates $i \in [m]$ where $c_i \leq c_{i^*} + 2 \left\| \mathbf{A} \right\|_\infty$. To see this, consider any $x \in \Delta^m$ where $x_i \neq 0$ for some $c_i \leq c_{i^*} + 2 \left\| \mathbf{A} \right\|_\infty$, and consider taking $\tilde{x} = x$ in every coordinate, except $\tilde{x}_{i^*} = x_{i^*} + x_i$ and $\tilde{x}_i = 0$ (i.e. we zero out the coordinate i and shift its mass to i^*). Then,

$$\langle c, x - \tilde{x} \rangle > 2 \left\| \mathbf{A} \right\|_\infty x_i = \left\| \mathbf{A} \right\|_\infty \left\| \tilde{x} - x \right\|_1 \geq \left\| \mathbf{A}^\top (\tilde{x} - x) \right\|_1 \geq \left\| \mathbf{A}^\top \tilde{x} - b \right\|_1 - \left\| \mathbf{A}^\top x - b \right\|_1 .$$

Rearranging implies $\tilde{x}$ obtains better objective value for the problem (5.2). Hence, by ignoring all large coordinates, any ϵ-approximately suboptimal point in the new problem is also ϵ-approximately suboptimal for the old problem, because the objective values are the same and any feasible point in the new problem is also feasible for the old problem. Finally, shifting c by a multiple of the all-ones vector affects the problem globally by a constant, since x is in the simplex.

Correctness of Line 2. Because $c^\top x$ is unaffected, it suffices to discuss the effect on

$$\left\| \mathbf{A}^\top x - b \right\|_1 = \sum_{j \in [n]} \left| \left[\mathbf{A}^\top x \right]_j - b_j \right| .$$

For disambiguation call b' the result of the truncation, and b the vector in the original problem. Consider a coordinate where $b_j \neq b'_j$; suppose without loss that $b_j > \|\mathbf{A}\|_\infty$ (the other case is handled symmetrically). Then, since $\left[\mathbf{A}^\top x\right]_j \leq \|\mathbf{A}\|_\infty$ by definition,

$$\left|\left[\mathbf{A}^\top x\right]_j - b_j\right| = b_j - \left[\mathbf{A}^\top x\right]_j = (b_j - \|\mathbf{A}\|_\infty) + \left(\|\mathbf{A}\|_\infty - \left[\mathbf{A}^\top x\right]_j\right) = (b_j - \|\mathbf{A}\|_\infty) + \left|\left[\mathbf{A}^\top x\right]_j - b'_j\right|.$$

Hence the effect is a constant scalar shift, as desired. Finally, regarding the semi-streaming implementation, b is n-dimensional and hence we can store it in $O(n)$ space. We can also store $\|\mathbf{A}\|_\infty$ and $C_{\max}$ in one pass and constant space, and simulate the post-processed vector c by modifying any coordinate c_i encountered in a stream using these precomputed values. □

Proposition 14. *By taking $T = O(\frac{\|\mathbf{A}\|_\infty \log m}{\epsilon})$ for a sufficiently large constant, Algorithm 13 results in iterates $\{(x'_t, y'_t)\}_{0 \leq t < T}$ so that $\bar{x} := \frac{1}{T}\sum_{0 \leq t < T} x'_t$ is an ϵ-approximate minimizer to (5.2).*

Proof. First, the effects of Lines 1 and 2 of Algorithm 13 does not affect the problem (5.2) other than changing the objective value everywhere by a constant, by Lemma 5 of [147] and Lemma 256 respectively. Next, the claim on the number of iterations required by Algorithm 13 follows immediately from Corollary 59. It remains to show that the given value of K suffices for Algorithm 14 to solve the subproblems to δ additive accuracy.

For simplicity, we will prove this bound on K suffices for the computation of w_t; the bound for z_{t+1} follows analogously. Lemma 251 demonstrates that in solving the minimization subproblem of Algorithm 14, every iteration of alternating minimizaton decreases the suboptimality gap for the subproblem by a constant factor, so we only require a bound on the initial error of each subproblem in Line 5 of Algorithm 13. The subproblem is of the form: minimize over $z = (x, y) \in \Delta^m \times [-1, 1]^n$,

$$\langle s_t^{\mathsf{x}} + g_t^{\mathsf{x}}, x\rangle + \langle s_t^{\mathsf{y}} + g_t^{\mathsf{y}}, y\rangle + x^\top |\mathbf{A}|(y^2) + 10\|\mathbf{A}\|_\infty \sum_{i \in [m]} x_i \log \frac{x_i}{[x_t]_i} =: \langle \gamma, z\rangle + r(z). \tag{D.14}$$

Call the optimal solution to the above problem z^*. Note that the initial point used by Algorithm 14 in Line 8 is z_t, which by assumption was an $\frac{\epsilon}{2}$-approximate minimizer to an objective of the form (D.14) where γ is replaced by γ' such that by construction (and our pre-processing of c),

$$\|(\gamma - \gamma')^{\mathsf{x}}\|_\infty = O\left(\|\mathbf{A}\|_\infty\right), \ \|(\gamma - \gamma')^{\mathsf{y}}\|_1 = O\left(\max\left(\|\mathbf{A}\|_\infty, \|b\|_1\right)\right).$$

Hence,

$$\begin{aligned}
\langle \gamma, z_t\rangle + r(z_t) &= \langle \gamma', z_t\rangle + r(z_t) + \langle \gamma - \gamma', z_t\rangle \\
&\leq \langle \gamma', z^*\rangle + r(z^*) + \frac{\epsilon}{2} + \langle \gamma - \gamma', z_t\rangle \\
&= \langle \gamma, z^*\rangle + r(z^*) + \frac{\epsilon}{2} + \langle \gamma - \gamma', z_t - z^*\rangle.
\end{aligned}$$

In the only inequality, we used the suboptimality bound on z_t. Hence, z_t can have initial function error at most $O(\|\mathbf{A}\|_\infty + \|b\|_1 + \epsilon)$ for the objective (D.14), yielding the required bound on K.

To give the final result, we claim that any ϵ-approximate saddle point $(\bar{x}, \bar{y})$ to a convex-concave function f on the domain $\mathcal{X}, \mathcal{Y}$ has that $f_{\mathcal{X}}(\bar{x})$ approximates the value of the primal-only problem

$$\min_{x \in \mathcal{X}} f_{\mathcal{X}}(x), \text{ where } f_{\mathcal{X}}(x) := \max_{y \in \mathcal{Y}} f(x, y),$$

within an additive ϵ. To see this, let $(x^\star, y^\star)$ be the exact saddle point of f so that $f(x^\star, y^\star) = \text{OPT}$. By definition of $(\bar{x}, \bar{y})$,

$$\left(\max_{y \in \mathcal{Y}} f(\bar{x}, y) - \text{OPT}\right) + \left(\text{OPT} - \min_{x \in \mathcal{X}} f(x, \bar{y})\right) \leq \epsilon.$$

Moreover, both quantities on the left hand side are nonnegative by definition of OPT, i.e.

$$\text{OPT} = f(x^\star, y^\star) = \min_{x \in \mathcal{X}} \left(\max_{y \in \mathcal{Y}} f(x, y)\right) = \max_{y \in \mathcal{Y}} \left(\min_{x \in \mathcal{X}} f(x, y)\right).$$

Thus, by nonnegativity of the second quantity, we conclude that

$$f_{\mathcal{X}}(\bar{x}) = \max_{y \in \mathcal{Y}} f(\bar{x}, y) \leq \text{OPT} + \epsilon = f_{\mathcal{X}}(x^\star) + \epsilon.$$

□

D.2 Matching tools from the literature

In this section, we prove Lemma 71 and Proposition 15. We begin with the former.

Lemma 71. *The greedy algorithm can be implemented in one semi-streaming pass over a graph using $O(n)$ space and $O(m)$ work to return a matching of size M with $M \leq M^* \leq 2M$.*

Proof. It is folklore that the greedy algorithm (adding edges one at a time to a matching, if and only if neither endpoint is already matched) is a 2-approximation to the MCM, proving the approximation guarantee. Regarding the implementation, we can keep marking an array of vertex indices and check if a current edge violates a matched vertex, in $O(1)$ time per edge. □

We now proceed to Proposition 15. We first state our algorithm, a greedy iterative procedure which repeatedly adds a maximal matching from a maintained vertex set to the remaining vertices.

We require one key technical claim for analyzing Algorithm 73.

Lemma 257. *Fix an iteration $0 \leq i \leq N - 1$ of Algorithm 73, and suppose the MCM size of $G[V_i]$ is M_i. Then, the MCM size of $G[V_{i+1}]$ increases by at least $\frac{1}{3}(M^* - M_i)$.*

Algorithm 73: VertexReduction(G, ϵ)

Input: Bipartite graph $G = (V, E)$ with MCM size M^*;
Output: Vertex subset $V' \subseteq V$ with $|V'| = O(M^* \log(\epsilon^{-1}))$ and $G[V']$ that has MCM size $\geq (1-\epsilon)M^*$;
$V_0 \leftarrow$ vertices incident to some maximal matching in G;
$N \leftarrow \lceil 3\log(\epsilon^{-1}) \rceil$;
for $i \in [N]$ **do**
  $S \leftarrow$ vertices in $V \setminus V_{i-1}$ incident to some maximal matching between V_{i-1} and $V \setminus V_{i-1}$;
  $V_i \leftarrow V_{i-1} \cup S$;
Return: $V' \leftarrow V_N$

Proof. The symmetric difference between the maximum matchings of $G[V_i]$ and G can be decomposed into even cycles and paths (to see this, every vertex in the symmetric difference has degree at most 2). Hence, ignoring all even cycles, there is a vertex-disjoint augmenting path decomposition of size $k := M^* - M_i$; call these paths $P_1, P_2, \ldots P_k$. For each path P_j, $j \in [k]$, all of its edges except the first and last are contained in $G[V_i]$; call the first and last edges (x_j, x'_j) and (y_j, y'_j), where $x_j, y_j \in G[V_i]$. We consider a few cases, "processing" paths sequentially.

1. If both (x_j, x'_j) and (y_j, y'_j) are included in the maximal matching between V_i and $V \setminus V_i$, this increases the maximum matching size in V_{i+1} by one since we can use the augmenting path.

2. If both x_j and y_j are matched to other vertices $x''_j \neq x'_j$ and $y''_j \neq y'_j$, we can still use the resulting augmenting path in V_{i+1}. The other vertices x''_j, y''_j can only remove two other later paths from consideration in the process, where we consider paths sequentially. Note that if x''_j or y''_j is not an endpoint from a later path, this can only help the argument here.

3. If one of x_j and y_j is matched to its corresponding endpoint in P_j but the other is not, we can use the augmenting path and remove one other path from consideration.

Because we took a maximal matching between V_i and $S \setminus V_i$, at least one of the above cases must happen, so we increase the MCM size by at least $\frac{1}{3}k$. □

We conclude with our analysis of Algorithm 73.

Proposition 15 (Vertex size reduction)**.** *There is a procedure* VertexReduction *(Algorithm 73) which takes as input unweighted bipartite $G = (V, E)$ with MCM size M^* and with $O(\log \epsilon^{-1})$ passes, $O(M^* \log \epsilon^{-1})$ space, and $O(m \log \epsilon^{-1})$ work outputs a subset $V' \subseteq V$ of size $O(M^* \log \epsilon^{-1})$ such that the induced subgraph $G[V']$ has a MCM of size at least $(1-\epsilon)M^*$.*

Proof. The correctness follows immediately by applying Lemma 257, and the fact that our original maximal matching (contained in $G[V_0]$) is of size at least $\frac{1}{2}M^*$. We now prove correctness. For the first pass, the complexity follows directly from Lemma 71. For all additional iterations of the

algorithm, note that we can only add $O(M^*)$ additional vertices to the current set since it is a valid matching in the original graph, and the iteration can be implemented in a single pass analogously to the proof of Lemma 71. The space overhead is only the maintenance of the current set V_i. □

D.3 Cycle cancelling in low space

In this section, we provide implementation of a data structure which proves the following claim.

Proposition 16 (Cycle cancelling)**.** *Consider a (possibly weighted) matching problem on a bipartite graph $G = (V, E, w)$ (for MCM, we let $w = \mathbf{1}$). There is an algorithm that has the following property: given a stream of length L, consisting of edge-flow tuples (e, f_e) where $e \in E$ and $f_e \in \mathbb{R}_{\geq 0}$, define $x \in \mathbb{R}^E_{\geq 0}$ to be the sum of all $f_e \mathbf{1}_e$ in the stream where $\mathbf{1}_e$ is the 1-sparse indicator of edge e. Then the algorithm runs in $O(n)$ space and $O(L \log n)$ time, and outputs a flow $\tilde{x}$ supported on $O(n)$ edges forming a forest, so that $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$, and $\mathbf{B}^\top x = \mathbf{B}^\top \tilde{x}$.*

Specifically, we reduce its proof to demonstrating the existence of a *bipartite cycle-cancelling oracle*.

Definition 51 (Bipartite cycle-cancelling oracle)**.** *We call $\mathcal{O}$ a* bipartite cycle-cancelling oracle *(BCCO) if it is associated with a (weighted) bipartite graph $G = (V, E, w)$, and given any vector $x \in \mathbb{R}^E_{\geq 0}$ supported on $L \leq 2n$ edges, $\mathcal{O}$ outputs a vector $\tilde{x} \in \mathbb{R}^E_{\geq 0}$ such that $\mathbf{B}^\top x = \mathbf{B}^\top \tilde{x}$ and $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$, so that $\tilde{x}$ is supported on at most n edges.*

Lemma 258. *A BCCO $\mathcal{O}$ is implementable in $O(L)$ space and $O(L \log n)$ work.*

Proof of Proposition 16. If the stream length is $L \leq 2n$, applying $\mathcal{O}$ obtains the desired results.

Otherwise, it suffices to divide the stream into chunks of n edges, and repeatedly call $\mathcal{O}$. Specifically, we can input the first $2n$ edges of the stream into $\mathcal{O}$ and produce a set of n edges, then input these n edges with the next n edges of the stream, and so on. Inducting on the properties of $\mathcal{O}$, we obtain the value and feasibility guarantees on the final output. Regarding the space and work guarantees, it suffices to repeatedly apply Lemma 258, and note that we can reuse the space for $\mathcal{O}$. □

The remainder of this section is devoted to proving Lemma 258. Specifically, we demonstrate how to use the link/cut tree data structure of [488, 500], with a few small modifications, to provide the required oracle. In the following discussion, we state some preliminaries on link/cut trees which are known from prior work (namely a set of supported operations), and we put these components together at the end of this section to give an implementation which proves Lemma 258.

D.3.1 Link/cut tree description

The description in this section primarily follows the implementation of link/cut trees given in Chapter 5 of [500], with a few terminology changes following the later lecture notes of [169]. It is primarily a summary of prior work, where we state the supported operations that we will use.

Data structure representation. The link/cut tree implementation of [488, 500] we will use is maintained as a forest, where each (undirected, rooted) tree in the forest is decomposed into "preferred paths" and each path is stored as a splay tree, along with a parent pointer to the neighbor of the preferred path endpoint closer to the root. Every edge in the forest has an associated value (which can change). We refer to the tree representations of preferred paths as "auxiliary trees" and the tree representation of all connected vertices as a "main tree." The link/cut tree supports queries or modifications to main trees, including the following operations.

1. $\mathsf{Link}(v, w, C)$: add an edge of value C between v and w in the main tree (if they are in different trees, join them so that w is the parent of v).
2. $\mathsf{Cut}(v)$: remove the edge between v and its parent in the main tree.
3. $\mathsf{ChangeRoot}(r)$: change the root of the main tree to the vertex r.
4. $\mathsf{LCA}(v, w)$: return the least common ancestor of v and w in the main tree.
5. $\mathsf{Min}(v)$: return the smallest edge value between v and its root in the main tree.
6. $\mathsf{Add}(v, C)$: add C to all edge values between v and its root in the main tree.
7. $\mathsf{Sum}(v)$: return the sum of all edge values between v and its root in the main tree.

For a more complete and formal description, we refer the reader to [500, 169].

Space complexity of link/cut trees. Based on the representation described above, it is clear that as long as the amount of additional information we need to store in the auxiliary trees to implement path aggregation operations is a constant per vertex, the overall space complexity is $O(n)$ where n is an upper bound on the number of vertices of all main trees. The only remaining space overhead is storing all auxiliary trees, and the parent pointers of preferred paths in main trees. We include this discussion because it is not explicitly stated in the source material.

Work complexity of link/cut trees. We state the guarantees of the link/cut tree in the following claim, which follows by the analysis of the works [488, 500].

Proposition 57 (Main result of [488, 500])**.** *The amoritized cost of L calls to any of the operations* Link*,* Cut*,* ChangeRoot*,* LCA*,* Min*,* Add*, or* Sum *is bounded by $O(L \log n)$.*

D.3.2 Implementation of BCCO

We now prove Lemma 258. We first outline our approach in constructing the oracle, and then describe a concrete implementation by using link/cut trees.

Approach outline. The main goal is to show how to preserve an acyclic matching $\tilde{x}$ so that $\mathbf{B}^\top \tilde{x} = \mathbf{B}^\top x$ and $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$ after processing every edge, in a total of $O(n \log n)$ work using the link/cut tree. Inductively it is clear that the output will satisfy all the requirements of $\mathcal{O}$.

We now describe how to process a single input edge $e = (u, v)$ where $u \in L$ and $v \in R$ are on opposite sides of the bipartition. First, if u and v are both not in the data structure, create a new main tree consisting of just this edge (with say u as the root). If only one is in the data structure then create a new edge with value x_e appropriately using Link. If u and v are in different main trees, we can call $\mathsf{Link}(u, v, x_e)$. In these cases, no cycles are created and no edge values are changed.

The last case is when u and v are in the same main tree. The path between u and v in their main tree and the new edge (u, v) forms a cycle. In order to preserve $\mathbf{B}^\top x$, it suffices to alternate adding and subtracting some amount C from edges along the cycle. In order to make sure $\langle w, x \rangle$ is monotone, it suffices to compute the alternating sum of weights along the cycle to pick a parity, i.e. whether we add from every odd edge in the cycle and subtract from every even edge, or vice versa (since all cycles are even length, one of these will increase the weight). Once we pick a parity, we compute the minimum value amongst all edges of that parity in the cycle and alternate adding and subtracting this value; this will result in the tree becoming acyclic and preserve all invariants.

We finally remark that it suffices to divide the support of the input x into connected components, and remove cycles from each connected component. By designating a "root vertex" in each connected component and processing the edges in that component in BFS order from the root, it is clear that the tree corresponding to that component will never become disconnected (edges are only removed when a cycle is formed, and removing any edge on the cycle will not disconnect the tree). Thus, it suffices to discuss how to implement cycle cancelling for a single connected component.

Implementation. We discuss the implementation of cycle cancelling for a single connected component. It will use three link/cut trees, called W, T_+, and T_-; the topology of these three trees will always be the same (i.e. they are the same tree up to the edge values). Roughly speaking, W will contain edge weights with alternating signs (e.g. each edge's weight will be signed $(-1)^{\text{depth of edge}}$), and will be used to determine the direction of cycle cancelling to preserve $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$. Further, T_+ and T_- will each correctly contain half of the edge values of the current fractional matching (depending on their sign in W), and the other edge values will be set to a large quantity. They are used to compute the minimum (signed) edge value in cycles and to modify the fractional matching.

Consider processing a new edge $e = (u, v)$ with weight w_e and value x_e. If this edge is in any of the non-cycle-creating cases described in the outline, i.e. its endpoints do not belong in the same tree with a path between them, then we add a copy to each of W, T_+, and T_-. Depending on the depth of the edge in W (which we can check by looking at the sign of its parent edge), we either

give it value w_e or $-w_e$. If it is positive in W, then we give it value x_e in T_+ and value $n^2 \|x\|_\infty$ in T_-; otherwise, we swap these values. We choose the value $n^2 \|x\|_\infty$ for one copy of the edge, so that it never becomes the minimum value edge returned by Min throughout the algorithm.

Finally, we handle the case where we need to remove a cycle (when u and v are in the same tree, and the edge is not in the tree). We start by determining which half of edges in the cycle we want to remove value from, and which we should add to. To do this, we query $r = \mathsf{LCA}(u, v)$ and $\mathsf{ChangeRoot}(r)$ on the tree in W. Then, we compute the alternating sums of weights in each half of edges in the cycle. This allows us to determine a direction to preserve $\langle w, \tilde{x} \rangle \geq \langle w, x \rangle$. We also call $\mathsf{LCA}(u, v)$ and $\mathsf{ChangeRoot}(r)$ for both copies of the main tree in T.

We next determine the amount we wish to alternatingly add and subtract along the u-to-r and v-to-r paths by calling Min on the appropriate tree, T_- or T_+ (corresponding to the direction obtained from querying W). We then call Add on both copies of the main tree in T for these paths, with the value obtained by the Min queries or its negation appropriately. This zeros one edge, which we Cut from all three copies of the tree. Finally, we revert to the original root in all three trees to maintain correctness of signs. Overall, the number of link/cut tree operations per edge in the stream is a constant, so Proposition 57 bounds the total work by $O(n \log n)$.

D.4 Sampling for rounding linear programming solutions

In this section, we give a general procedure for rounding a fractional solution which is returned by our algorithm for box-simplex games in Section 5.3. This procedure applies to general box-simplex problems, beyond those with combinatorial structure, so we include this section for completeness. In particular, directly applying the sparsity bounds of this section to our matching problems directly imply $\widetilde{O}(n \cdot \text{poly}(\epsilon^{-1}))$ space bounds for the various matching-related applications in the paper, up to width parameters, which roughly match our strongest results up to the ϵ^{-1} dependence. As an example, we give an application of this technique to MCM at the end of this section.

Given streaming access to an approximate fractional solution x on the simplex (via an implicit representation), a natural way of constructing a low-space approximate solution is to randomly sample each entry of x_i and reweight to preserve expected objective value; this is the rounding strategy we analyze. We use the following Algorithm 2, parameterized by some prescribed $\{M_i\}_{i \in [m]}$.

D.4.1 Concentration bounds

For proofs in this section, as well as later, we crucially rely on well-known concentration properties of bounded random variables. We state here a few facts used repeatedly throughout.

Proposition 58 (Chernoff bound)**.** *For K independent scaled Bernoulli random variables $\{X_k\}_{k \in [K]}$*

Algorithm 2 RandomSample$(x, K, \{M_i\}_{i\in[m]})$

Input: Coordinates of $x \in \mathbb{R}^m_{\geq 0}$ in streaming fashion, sample count K, parameters $\{M_i\}_{i\in[m]}$

for $i \in [m]$ **do**

  Draw K independent random variables $\{X_i^k\}_{k\in[K]}$, where

$$X_i^k = \begin{cases} M_i & \text{with probability } \frac{x_i}{M_i} \\ 0 & \text{otherwise} \end{cases}.$$

  $\hat{x}_i \leftarrow \frac{1}{K}\sum_{k\in[K]} X_i^k$

return $\hat{x}$

satisfying $X_k = N_k$ with probability p_k, $0 < N_k \leq 1$ for all $k \in [K]$, and all $0 < \delta < 1$,

$$\Pr\left(\left|\sum_{k\in[K]} X_k - \sum_{k\in[K]} \mathbb{E}X_k\right| \geq \delta \sum_{k\in[K]} \mathbb{E}X_k\right) \leq 2\exp\left(-\frac{\delta^2 \sum_{k\in[K]} \mathbb{E}X_k}{3}\right)$$

We give a simple generalization of Proposition 58 to the case where the scaled Bernoulli variables are allowed to take on negative values.

Corollary 60 (Generalized Chernoff bound)**.** *For K independent scaled Bernoulli random variables $\{X_k\}_{k\in[K]}$ satisfying $X_k = N_k$ with probability p_k, $0 < |N_k| \leq 1$ for all $k \in [K]$, and all $0 < \delta < 1$,*

$$\Pr\left(\left|\sum_{k\in[K]} X_k - \sum_{k\in[K]} \mathbb{E}X_k\right| \geq \delta \sum_{k\in[K]} \mathbb{E}|X_k|\right) \leq 4\exp\left(-\frac{\delta^2 \sum_{k\in[K]} \mathbb{E}|X_k|}{3}\right)$$

Proof. Divide the set $[K] = \mathcal{K}^+ \cup \mathcal{K}^-$, where we define $\mathcal{K}^+ := \{k \in [K] | N_k \geq 0\}$ and $\mathcal{K}^+ := \{k \in [K] | N_k < 0\}$. Applying Proposition 58 to $\sum_{k\in\mathcal{K}^+} X_k$ and $\sum_{k\in\mathcal{K}^-} -X_k$ yields the result. □

Furthermore, the following one-sided Chernoff bound holds when $0 < N_k \leq 1$ and $\delta \geq 1$.

Proposition 59 (One-sided Chernoff bound)**.** *For K independent scaled Bernoulli random variables $\{X_k\}_{k\in[K]}$ satisfying $X_k = N_k$ with probability p_k, $0 < N_k \leq 1$ for all $k \in [K]$, and all $\delta > 0$,*

$$\Pr\left(\sum_{k\in[K]} X_k - \sum_{k\in[K]} \mathbb{E}X_k \geq \delta \sum_{k\in[K]} \mathbb{E}X_k\right) \leq \exp\left(-\frac{\delta^2 \sum_{k\in[K]} \mathbb{E}X_k}{2+\delta}\right)$$

Proposition 60 (Bernstein's inequality)**.** *For K independent random variables $\{X_k\}_{k\in[K]}$ satisfying $|X_k| \leq C$ with probability one, let $V = \sum_{k\in[K]} \mathrm{Var}[X_k]$. Then for all $t \geq 0$,*

$$\Pr\left(\left|\sum_{k\in[K]} X_k - \sum_{k\in[K]} \mathbb{E}X_k\right| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2V + 2Ct/3}\right).$$

D.4.2 Random sampling guarantees

We first give a general guarantee on the approximation error incurred by random sampling via Algorithm 2. While the guarantees are a bit cumbersome to state, they become significantly simpler in applications. For instance, in all our applications, all binary random variables are scaled with M_i such that $\max_{i\in[n]} M_i \leq 1$ (see Lemma 259 for definition), and the bounds on $\mathbf{A}^\top \hat{x} - \mathbf{A}^\top x$ become standard multiplicative error approximations when the matrix $\mathbf{A}$ is all-positive.

Lemma 259. *Consider an instance of problem* (5.1) *parameterized by* $\mathbf{A}$*,* b*,* c*. For some* $x \in \Delta^m$ *whose coordinates can be computed in streaming fashion, define for all* $j \in [n]$*,*

$$B_j = \left[|\mathbf{A}|^\top x \right]_j.$$

Let $\hat{x}$ *be the output of Algorithm 2 on input* x *with*

$$M_i = \min_{j\in[n]} \frac{B_j}{|\mathbf{A}_{ij}|} \text{ and } K = \frac{12\log(mn)}{\epsilon^2}.$$

With probability at least $1-(mn)^{-1}$*,* $\hat{x}$ *satisfies the following properties for* $B := \max_{i\in[m]} M_i$*:*

$$\left| \left[\mathbf{A}^\top \hat{x} - \mathbf{A}^\top x \right]_j \right| \leq \epsilon B_j \text{ for all } j \in [n], \quad |\, \|\hat{x}\|_1 - \|x\|_1 \,| \leq \epsilon \max(1, B),$$
$$\left| c^\top \hat{x} - c^\top x \right| \leq \epsilon \|c\|_\infty \max(1, B),$$
$$\|\hat{x}\|_0 = O\left(\left(1 + \sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j} \right) \cdot \frac{\log(mn)}{\epsilon^2} \right).$$

Proof. First note that it is immediate by definition of the B_j to see that all $\frac{x_i}{M_i} = \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|x_i}{B_j} \leq 1$ are valid sampling probabilities for all $i \in [m]$. Also, recall that entrywise

$$\hat{x}_i = \frac{1}{K} \sum_{k\in[K]} X_i^k.$$

We show the first property; fix some $j \in [n]$ and consider $[\mathbf{A}^\top \hat{x}]_j - [\mathbf{A}^\top x]_j$. Applying Corollary 60

to the random variables $\{\frac{1}{B_j}\mathbf{A}_{ij}X_i^k\}_{i\in[m],k\in[K]}$ with $\delta=\epsilon$, we see by definition of K that

$$\begin{aligned}
\sum_{i\in[m],k\in[K]}\frac{1}{B_j}\mathbf{A}_{ij}X_i^k &= \frac{K}{B_j}\left[\mathbf{A}^\top\hat{x}\right]_j,\\
\sum_{i\in[m],k\in[K]}\mathbb{E}\left[\frac{1}{B_j}\mathbf{A}_{ij}X_i^k\right] &= \frac{K}{B_j}[\mathbf{A}^\top x]_j,\\
\sum_{i\in[m],k\in[K]}\mathbb{E}\left|\frac{1}{B_j}\mathbf{A}_{ij}X_i^k\right| &= \frac{K}{B_j}[|\mathbf{A}|^\top x]_j,\\
\implies \Pr\left(\frac{K}{B_j}\left|[\mathbf{A}^\top\hat{x}]_j-[\mathbf{A}^\top x]_j\right|\geq\delta\frac{K}{B_j}[|\mathbf{A}|^\top x]_j\right) &\leq 4\exp\left(-\frac{\delta^2K[|\mathbf{A}|^\top x]_j}{3B_j}\right)\leq\frac{4}{(nm)^4},
\end{aligned}$$

where for the last inequality we use definitions of δ, K, and that $B_j=[|\mathbf{A}|^\top x]_j$, for all $j\in[n]$.

This conclusion for a coordinate $j\in[n]$ is equivalent to $\left|[\mathbf{A}^\top\hat{x}]_j-[\mathbf{A}^\top x]_j\right|\leq\epsilon B_j$. Union bounding over all $j\in[n]$, we thus have with probability at least $1-\frac{4}{n^3m^4}$

$$\left|\left[\mathbf{A}^\top\hat{x}-\mathbf{A}^\top x\right]_j\right|\leq\epsilon B_j,\quad\forall j\in[n].$$

Now we show the second and third properties. Given $B:=\max_{i\in[m]}M_i$, we first apply Proposition 60 on the sum $\sum_{i\in[m]}Kc_i\hat{x}_i=\sum_{i\in[m],k\in[K]}c_iX_i^k$, using the bound

$$\begin{aligned}
\sum_{i\in[m],k\in[K]}\operatorname{Var}\left[c_iX_i^k\right] &\leq K\left\|c\right\|_\infty^2\sum_{i\in[m]}\operatorname{Var}\left[X_i^k\right]\\
&\leq K\left\|c\right\|_\infty^2\sum_{i\in[m]}x_iM_i\leq K\left\|c\right\|_\infty^2B.
\end{aligned}$$

Thus, we can choose parameters $C=\left\|c\right\|_\infty B$, $V=K\left\|c\right\|_\infty^2B$ to obtain the following bounds for $K\geq\frac{12\log(mn)}{\epsilon^2}$ depending on whether $B>1$ or $B\leq1$. For $B>1$, we have

$$\begin{aligned}
\Pr\left(K\left|\sum_{i\in[m]}c_i\hat{x}_i-\sum_{i\in[m]}c_ix_i\right|\geq\epsilon B\left\|c\right\|_\infty K\right) &\leq 2\exp\left(-\frac{\epsilon^2B^2\left\|c\right\|_\infty^2K^2}{2K\left\|c\right\|_\infty^2B+2\left\|c\right\|_\infty^2B^2K\epsilon/3}\right)\\
&\leq\frac{1}{4(mn)^2}.
\end{aligned}$$

For $B\leq1$, we have

$$\begin{aligned}
\Pr\left(K\left|\sum_{i\in[m]}c_i\hat{x}_i-\sum_{i\in[m]}c_ix_i\right|\geq\epsilon\left\|c\right\|_\infty K\right) &\leq 2\exp\left(-\frac{\epsilon^2\left\|c\right\|_\infty^2K^2}{2K\left\|c\right\|_\infty^2B+2\left\|c\right\|_\infty^2BK\epsilon/3}\right)\\
&\leq\frac{1}{4(mn)^2}.
\end{aligned}$$

Altogether this implies the third conclusion, and the second conclusion follows as a special case when specifically picking $c = \mathbf{1}_n$. Each holds with probability $\geq 1 - \frac{1}{4}(mn)^{-2}$.

Finally, for all $i \in [m]$, let $Y_i = 1$ if $\hat{x}_i \neq 0$ and $Y_i = 0$ otherwise. It is straightforward to see that $Y_i = 1$ with probability

$$1 - \left(1 - \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|x_i}{B_j}\right)^K \leq K \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|x_i}{B_j}$$

$$\implies \sum_{i\in[m]} \mathbb{E}[Y_i] \leq K \sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j} \leq K \max\left(\sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j}, 1\right).$$

Thus, by applying the one-sided Chernoff bound in Proposition 59 (where we consider the cases where $\sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j} \geq 1$ and $\sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j} \leq 1$ separately),

$$\Pr\left(\sum_{i\in[m]} Y_i \geq 2K + 2K\left(\sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j}\right)\right) \leq \frac{1}{4(mn)^2}.$$

Finally, applying a union bound, all desired events hold with probability $1 - \frac{1}{mn}$. □

Note that in the semi-streaming model, one can choose $B_j := \left[|\mathbf{A}|^\top x\right]_j$ and compute all such values in one pass when x is of the form in Lemma 69; however, occasionally we will choose larger values of B_j to obtain improved sparsity guarantees. Thus, we provide the following one-sided guarantee, which holds when $\mathbf{A}_{ij} \geq 0$, $\forall i, j$ and $B_j \geq [\mathbf{A}^\top x]_j$.

Corollary 61. *Consider an instance of problem* (5.1) *parameterized by* $\mathbf{A}$ *with* $\mathbf{A}_{ij} \geq 0$ $\forall$ *i,j, and b, c. For some* $x \in \Delta^m$ *whose coordinates can be computed in streaming fashion, suppose we have bounds for all* $j \in [n]$*,*

$$B_j \geq \left[\mathbf{A}^\top x\right]_j.$$

Let $\hat{x}$ *be the output of Algorithm 2 on input x with*

$$M_i = \min_{j\in[n]} \frac{B_j}{|\mathbf{A}_{ij}|}, \text{ and } K = \frac{12\log(mn)}{\epsilon^2}.$$

With probability at least $1 - (mn)^{-1}$*,* $\hat{x}$ *satisfies the following properties for* $B := \max_{i\in[m]} M_i$*:*

$$\left[\mathbf{A}^\top \hat{x} - \mathbf{A}^\top x\right]_j \leq \epsilon B_j \text{ for all } j \in [n], \quad |\,\|\hat{x}\|_1 - \|x\|_1\,| \leq \epsilon \max(1, B),$$

$$\left|c^\top \hat{x} - c^\top x\right| \leq \epsilon \|c\|_\infty \max(1, B), \quad \|\hat{x}\|_0 = O\left(\left(1 + \sum_{i\in[m]} x_i \max_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{B_j}\right) \cdot \frac{\log(mn)}{\epsilon^2}\right).$$

Note that the first and fourth properties of Corollary 61 are one-sided in that we only wish to

upper bound a property of $\hat{x}$. The proof is identical to Lemma 259, except that we use Proposition 59 with $\delta = \epsilon \cdot \frac{B_j}{[|\mathbf{A}|^\top x]_j} > 0$ instead of Proposition 58 in cases where $\delta \geq 1$, which suffices for the one-sided bound of the first property. Similarly, when $\delta \geq 1$ as in the proof of the fourth property, the one-sided Chernoff bound only helps concentration. Next, we give an end-to-end guarantee on turning Algorithm 13 into a solver for the fractional problem, via applying Algorithm 2 on the output.

Lemma 260. *Given (x, y), an $\frac{\epsilon}{2}$-approximate saddle point of (5.1), let*

$$B_j = [|\mathbf{A}|^\top x]_j, \quad B = \max_{i \in [m]} \min_{j \in [n]} \frac{B_j}{|\mathbf{A}_{ij}|}$$

and $\hat{x} = \mathsf{RandomSample}(x, K, \{M_i\}_{i \in [m]})$ with

$$M_i = \min_{j \in [n]} \frac{B_j}{|\mathbf{A}_{ij}|}, \text{ and } K = O\left(\frac{\log(mn) \left(\left(\|c\|_\infty^2 + \|\mathbf{A}\|_\infty^2 \right) (1+B)^2 + \left(\sum_{j \in [n]} B_j \right)^2 \right)}{\epsilon^2} \right).$$

With probability $1 - (mn)^{-1}$, $(\frac{\hat{x}}{\|\hat{x}\|_1}, y)$ is an ϵ-approximate saddle point to (5.1). Moreover, the total space complexity of Algorithm 13 and Algorithm 2 to compute and store the output $(\frac{\hat{x}}{\|\hat{x}\|_1}, y)$ is

$$O\left(\log n \left(\sum_{i \in [m]} x_i \max_{j \in [n]} \frac{|\mathbf{A}_{ij}|}{B_j} \right) \cdot \frac{\left(\left(\|c\|_\infty^2 + \|\mathbf{A}\|_\infty^2 \right) (1+B)^2 + \left(\sum_{j \in [n]} B_j \right)^2 \right)}{\epsilon^2} + \frac{n \|\mathbf{A}\|_\infty}{\epsilon} \right).$$

Proof. Our choice of K is with respect to an accuracy parameter on the order of

$$\frac{\epsilon}{(\|c\|_\infty + \|\mathbf{A}\|_\infty)(1+B) + \sum_{j \in [n]} B_j}.$$

Recall that an ϵ-approximate saddle point to a convex-concave function f satisfies

$$\max_{\bar{y} \in \mathcal{Y}} f(x, \bar{y}) - \min_{\bar{x} \in \mathcal{X}} f(\bar{x}, y) \leq \epsilon.$$

For our output $(\frac{\hat{x}}{\|\hat{x}\|_1}, y)$, since we keep the same box variable y, it suffices to show that the side of the duality gap due to x and $\frac{\hat{x}}{\|\hat{x}\|_1}$ does not change significantly. Namely, we wish to show

$$\max_{\bar{y} \in \mathcal{Y}} f\left(\frac{\hat{x}}{\|\hat{x}\|_1}, \bar{y} \right) - \max_{\bar{y} \in \mathcal{Y}} f(x, \bar{y}) = \left(c^\top \frac{\hat{x}}{\|\hat{x}\|_1} + \left\| \mathbf{A}^\top \frac{\hat{x}}{\|\hat{x}\|_1} - b \right\|_1 \right) - \left(c^\top x + \left\| \mathbf{A}^\top x - b \right\|_1 \right) \leq \frac{\epsilon}{2}.$$

By the triangle inequality, it equivalently suffices to show that

$$c^\top \hat{x} - c^\top x \le \frac{\epsilon}{8},\ c^\top \hat{x}\left(\frac{1}{\|\hat{x}\|_1} - 1\right) \le \frac{\epsilon}{8},$$
$$\left\|\mathbf{A}^\top \hat{x} - \mathbf{A}^\top x\right\|_1 \le \frac{\epsilon}{8},\ \left\|\mathbf{A}^\top \hat{x}\right\|_1 \left|\frac{1}{\|\hat{x}\|_1} - 1\right| \le \frac{\epsilon}{8}.$$

The first and third conclusions hold by applying Lemma 259 for the choice of K. The second and fourth hold by $|c^\top \hat{x}| \le \|c\|_\infty \|\hat{x}\|_1$ and $\left\|\mathbf{A}^\top \hat{x}\right\|_1 \le \|\mathbf{A}\|_\infty \|\hat{x}\|_1$ and then applying Lemma 259 for the choice of K. Finally, the desired sparsity follows by combining the space bound of the output (via Lemma 259) and the space complexity of implicitly representing the average iterate. □

D.4.3 Application: rounding MCM solutions

We give a simple application of our random sampling framework to computing an explicit low-space approximate MCM. While the space complexity does not match our strongest results based on a low-space cycle cancelling implementation, we hope it is a useful example of how to more generally sparsify fractional solutions of box-simplex games without explicit combinatorial structure.

Following the reductions of Section 5.4, we assume in this section that we have a simplex variable $x \in \Delta^m$ and a matching size $\bar{M} \in [1, n]$ such that

$$\mathbf{B}^\top(\bar{M}x) \le \mathbf{1}_V,\ \bar{M} \ge (1-\epsilon)M^*,$$

where M^* is the maximum matching size. We now show how to apply our random sampling procedure, Algorithm 2, to sparsify the support of the matching without significant loss.

Corollary 62. *Suppose for an MCM problem, $x \in \Delta^E$ satisfies $\mathbf{B}^\top x \le \frac{1}{\bar{M}}\mathbf{1}_V$, and $\epsilon \in (0,1)$. Let $\hat{x}$ be the output of Algorithm 2 on input x with $m_e = \frac{1}{\bar{M}}$ for all $e \in E$, and $K := \frac{12\log n}{\epsilon^2}$. Then with probability at least $1 - n^{-2}$, the output $\hat{x}$ satisfies the following properties:*

$$\mathbf{B}^\top \hat{x} \le \frac{1+\epsilon}{\bar{M}}\mathbf{1}_V, \quad |\,\|\hat{x}\|_1 - 1| \le \epsilon, \quad \|\hat{x}\|_0 = O\left(\frac{\bar{M}\log n}{\epsilon^2}\right).$$

Proof. It is straightforward to see that by the assumptions, we can take $\mathbf{A} = \bar{M}\mathbf{B}$ and $B_v = 1$ for all $v \in V$, and choose the accuracy level to be ϵ in Lemma 259. Thus, by the conclusions of Lemma 259, noting that $M_e = \frac{1}{\bar{M}} \le 1$ holds for all $e \in E$, we conclude that with probability at least $1 - n^{-2}$, all

desired guarantees hold:

$$\bar{M}\mathbf{B}^\top \hat{x} \leq \bar{M}\mathbf{B}^\top x + \epsilon \mathbf{1}_V \leq (1+\epsilon)\mathbf{1}_V,$$
$$|\,\|\hat{x}\|_1 - 1| \leq \epsilon,$$
$$\|\hat{x}\|_0 = O\left(\left(\sum_{e\in E} x_e \max_{v\in V} \frac{|\mathbf{A}_{ev}|}{B_v}\right)\cdot \frac{\log(mn)}{\epsilon^2}\right) = O\left(\frac{\bar{M}\log n}{\epsilon^2}\right).$$

□

D.5 Approximate MCM via box-constrained Newton's method

The goal of this section is to prove the following Theorem 84. In particular, we give an alternate second-order method to compute $(1-\epsilon)$-approximate maximum matchings in unweighted graphs using techniques developed by [145] for solving matrix scaling and balancing problems.

Theorem 84. *For a MCM problem on bipartite $G = (V, E)$ with $|V| = n$, $|E| = m$ and optimal value M^*, Algorithm 74 with parameter[6] $\epsilon \in [\Theta\left(\frac{\log(mn)}{n}\right), \frac{1}{2}]$ obtains a matching of size at least $(1-\epsilon)M^*$ using $\widetilde{O}(n)$ space, $\widetilde{O}(\epsilon^{-1})$ passes, and $\widetilde{O}(m)$ work per pass.*

We first recall the dual (vertex cover) formulation of the standard bipartite matching linear program

$$\min_{v\geq 0, \mathbf{B}v\geq \mathbf{1}} \mathbf{1}^\top v, \tag{D.15}$$

where $\mathbf{B}$ is the unoriented incidence matrix. Our method (which is based on the box-constrained Newton's method of [145]) uses a relaxation of this linear program which makes use of oriented incidence matrices and Laplacians, which we now define.

Definition 52. *Let $G = (V, E, w)$ be a weighted undirected bipartite graph with bipartition L, R and nonnegative edge weights w. We define the unoriented incidence matrix $\mathbf{B} \in \mathbb{R}^{E\times V}$ as the matrix where for any edge $e = (i, j), i \in L, j \in R$ the row corresponding to e in $\widetilde{\mathbf{B}}$ has $\widetilde{\mathbf{B}}_{ei} = \widetilde{\mathbf{B}}_{ej} = 1$, and all other entries set to 0. Additionally, we define the* oriented *incidence matrix $\widetilde{\mathbf{B}} \in \mathbb{R}^{E\times V}$ as*

$$\widetilde{\mathbf{B}} = \begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix} \mathbf{B},$$

and the graph Laplacian $\mathcal{L}_G = \widetilde{\mathbf{B}}^\top \mathbf{diag}\,(w)\,\widetilde{\mathbf{B}}$.

When the graph is obvious we drop the subscript from $\mathcal{L}_G$. The orientation of the edges in $\widetilde{\mathbf{B}}$ are typically chosen to be arbitrary in the literature, but we specify edge orientation as our algorithm

[6] This lower bound on ϵ is without loss of generality as otherwise we can use the algorithm in Section 5.5.1 which computes an exact MCM with a larger value of ϵ.

will distinguish the vertices in L and R, and denote a vector x on these vertices as x_L and x_R (or sometimes $[x]_L$, $[x]_R$ for clarity). We will first describe our regularization scheme for this LP and prove that approximate minimizers for the regularized objective yield approximate fractional vertex covers. We then prove some stability properties on the Hessian of our regularized objective and show that a second-order method can be implemented in $\widetilde{O}(n)$ space and $\widetilde{O}(\epsilon^{-1})$ passes. Our choice of regularization and notion of stability are heavily based on [145], which employed a similar regularization scheme to solve matrix scaling and balancing problems. We nevertheless give a self-contained description of these details for completeness.

Lemma 261 (Properties of regularized vertex cover)**.** *Let G be an unweighted bipartite graph with n nodes and m edges, and let L, R be the vertices on either side of its bipartition. Let M^* be the size of the maximum matching in G. Let $\varepsilon \geq \frac{8\log(mn)}{n}$ be a parameter, and set $\mu = \frac{\varepsilon}{4\log(mn)}$. Consider*

$$f_\mu(x) := \mathbf{1}^\top x + \mu\left(\sum_{e\in E} e^{\frac{1}{\mu}(1-[\mathbf{B}x]_e)} + \sum_{i\in V} e^{-\frac{1}{\mu}(x_i+\frac{\varepsilon}{4})}\right). \tag{D.16}$$

f_μ has the following properties.

- *For any x with $f_\mu(x) < m$ we have $\mathbf{B}x \geq (1-\frac{\varepsilon}{2})\mathbf{1}$ and $x \geq -\frac{\varepsilon}{2}$*
- *If $x^\star$ is a feasible minimizer to D.15, $x := (1+\frac{\varepsilon}{2})x^\star$ has $f_\mu(x) \leq (1+\frac{2\varepsilon}{3})M^*$.*
- *$f_\mu(x) \geq M^* - \varepsilon n$ for any x.*
- *For any x with $f_\mu(x) < m$, let $x' = \min\{x, 2\cdot\mathbf{1}\}$ Then $f_\mu(x') \leq f_\mu(x)$.*

Proof. We prove the claims in order. For the first claim, let i be the index of the smallest entry of x. If $x_i \leq -\frac{\varepsilon}{2}$, then since $x_i + \frac{\varepsilon}{4} \leq \frac{x_i}{2}$ and exp is always nonnegative,

$$f_\mu(x) \geq \mathbf{1}^\top x + \mu\exp\left(-\frac{x_i}{2\mu}\right) \geq nx_i + \mu\exp\left(-\frac{x_i}{2\mu}\right) \geq m.$$

The second inequality followed from the definition of x_i and the third from monotonicity of the expression in x_i. Thus $x \geq -\frac{\varepsilon}{2}\mathbf{1}$ as claimed. This also implies that $\mathbf{1}^\top x \geq -\frac{\varepsilon n}{2}$, so it is straightforward to verify that if some $1-[\mathbf{B}x]_i \geq \frac{\varepsilon}{2}$, then we would have $f_\mu(x) \geq -\frac{\varepsilon n}{2} + (mn)^2\mu \geq m$, completing the first claim.

Next, since $x^\star$ is a feasible minimizer to (D.15) we have $\mathbf{1}^\top x^\star = M^*$, $x^\star \geq 0$, and $\mathbf{B}x^\star \geq 1$. Therefore by construction, $\mathbf{1}^\top x = (1+\frac{\varepsilon}{2})M^*$, $x \geq 0$, and $\mathbf{B}x = (1+\frac{\varepsilon}{2})\mathbf{B}x^\star \geq (1+\frac{\varepsilon}{2})$. Plugging

these into $f_\mu(x)$ yields (since $M^* \geq 1$)

$$\begin{aligned} f_\mu(x) &\leq \left(1+\frac{\varepsilon}{2}\right) M^* + \mu\left(m\exp\left(\frac{-\varepsilon}{2\mu}\right) + n\exp\left(\frac{-\varepsilon}{4\mu}\right)\right) \\ &\leq \left(1+\frac{\varepsilon}{2}\right) M^* + \mu\frac{m+n}{mn} \leq \left(1+\frac{2\varepsilon}{3}\right) M^*. \end{aligned}$$

For the third claim, suppose x satisfied $f_\mu(x) \leq M^* - \varepsilon n$. As $M^* \leq n$ we may apply the first claim and obtain $x \geq -\frac{\varepsilon}{2}$ and $\mathbf{B}x \geq \mathbf{1} - \frac{\varepsilon}{2}$. Let $v = x + \frac{\varepsilon}{2}\mathbf{1}$, so $v \geq 0$ and $\mathbf{B}v \geq \mathbf{1}$. This is a contradiction: as v is feasible for (D.15) we have $M^* \leq \mathbf{1}^\top v = \mathbf{1}^\top x + \frac{\varepsilon n}{2} < f_\mu(x) + \frac{\varepsilon n}{2} < M^*$.

Finally, note that $f_\mu(x) < m$ combined with the first property of f_μ implies that $\mathbf{B}x \geq (1-\frac{\varepsilon}{2})\mathbf{1}$, $x \geq -\frac{\varepsilon}{2}$. Assume x has $x_i = \alpha > 2$ for some i: we will show that decreasing this coordinate to 2 decreases f_μ. The only terms affected by changing x_i are $\mathbf{1}^\top x$ (which decreases by $\alpha - 2$), $\mu\sum_{j\in V} e^{-\frac{1}{\mu}(x_j+\frac{\varepsilon}{4})}$, and $\mu\sum_{e\in E} e^{\frac{1}{\mu}(1-[\mathbf{B}x]_e)}$. The first of these sums increases by at most

$$\mu\exp\left(-\frac{2+\frac{\varepsilon}{4}}{\mu}\right) - \mu\exp\left(-\frac{\alpha+\frac{\varepsilon}{4}}{\mu}\right) \leq (\alpha-2)e^{\frac{2}{\mu}} \leq \frac{\alpha-2}{mn},$$

while each of the m terms in the second sum increases by at most

$$\mu\exp\left(-\frac{-1-[x_R]_j}{\mu}\right) - \mu\exp\left(-\frac{1-\alpha-[x_R]_j}{\mu}\right) \leq (\alpha-2)e^{\frac{1-\varepsilon}{\mu}} \leq \frac{\alpha-2}{mn},$$

for some x_j: we use our lower bound on x here. Thus the total change in $f_\mu(x)$ is at most $-(\alpha - 2)\left(1 - \frac{1}{mn} - \frac{1}{n}\right) < 0$. □

We next show that derivatives of f satisfy some useful stability properties, which we now define.

Definition 53. *Let f be a convex function. We say that f is r-second order robust if for any vectors x, y where $\|x-y\|_\infty \leq r$ we have (where $\preceq$ is the Loewner order on the positive semidefinite cone)*

$$e^{-1}\nabla^2 f(x) \preceq \nabla^2 f(y) \preceq e\nabla^2 f(x).$$

Lemma 262. *Let*

$$g_\mu(x) = f_\mu\left(\begin{bmatrix}\mathbf{I}_L & 0\\ 0 & -\mathbf{I}_R\end{bmatrix} x\right)$$

where f_μ is defined in (D.16)*. The gradient and Hessian of g_μ are*

$$\nabla g_\mu(x) = \begin{bmatrix}\mathbf{1}_L - z_L\\ -\mathbf{1}_R + z_R\end{bmatrix} - \widetilde{\mathbf{B}}^\top y,\ \nabla^2 g_\mu(x) = \frac{1}{\mu}\left(\mathbf{diag}\,(z) + \widetilde{\mathbf{B}}^\top\ \mathbf{diag}\,(y)\,\widetilde{\mathbf{B}}\right)$$

where $y := \exp\left(\frac{1}{\mu}\left(\mathbf{1} - \widetilde{\mathbf{B}}x\right)\right)$, $z_L := \exp\left(-\frac{1}{\mu}(x_L + \frac{\varepsilon}{4}\mathbf{1})\right)$, $z_R := \exp\left(\frac{1}{\mu}(x_R - \frac{\varepsilon}{4}\mathbf{1})\right)$, and $z :=$

$[z_L, z_R]$[7]. *For all x, $\nabla^2 g_\mu(x)$ is a nonnegative diagonal matrix plus a weighted graph Laplacian matrix. Further, g_μ is convex and $\frac{\mu}{2}$-second order robust. Finally, for any x, δ we have*

$$\delta^\top \nabla^2 g_\mu(x)\delta \leq \frac{2}{\mu} \cdot \delta^\top \left(\mathbf{diag}\left(\mathbf{B}^\top y\right) + \mathbf{diag}\left(z\right)\right)\delta.$$

Proof. The gradient and Hessian claims may be verified directly as we have via the chain rule

$$\nabla g_\mu(x) = \begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix} \nabla f_\mu \left(\begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix} x \right)$$

and

$$\nabla^2 g_\mu(x) = \begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix} \nabla^2 f_\mu \left(\begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix} x \right) \begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix}.$$

It is clear that $\frac{z}{\mu}$ is nonnegative and $\widetilde{\mathbf{B}}^\top \mathbf{diag}\left(\frac{y}{\mu}\right)\widetilde{\mathbf{B}}$ is a weighted graph Laplacian. Convexity of g_μ follows from $\nabla^2 g_\mu(x) \succeq 0$ everywhere. To prove second order robustness, for any x, x' with $\|x - x'\|_\infty \leq \frac{\mu}{2}$,

$$\left\|\widetilde{\mathbf{B}}x - \widetilde{\mathbf{B}}x'\right\|_\infty \leq \left\|\widetilde{\mathbf{B}}\right\|_\infty \|x - x'\|_\infty \leq 2 \cdot \frac{\mu}{2} \leq \mu.$$

Thus every matrix forming the Hessian of g_μ changes by at most a factor of e multiplicatively, and so g_μ is second order robust. For the final claim, let $|\delta|$, δ^2 be applied entrywise. Then,

$$\begin{aligned}\delta^\top \widetilde{\mathbf{B}}^\top \mathbf{diag}\left(y\right)\widetilde{\mathbf{B}}\delta &= \sum_{i\in[m]} y_i[\widetilde{\mathbf{B}}\delta]_i^2 \leq \sum_{i\in[m]} y_i[\mathbf{B}|\delta|]_i^2 \\ &\leq \sum_{i\in[m]} y_i[\mathbf{B}\delta^2]_i[\mathbf{B}\mathbf{1}]_i = 2\sum_{i\in[m]} y_i[\mathbf{B}\delta^2]_i = 2\delta^\top \mathbf{diag}\left(\mathbf{B}^\top y\right)\delta.\end{aligned}$$

The second line used Cauchy-Schwarz and $\mathbf{B}\mathbf{1} = 2 \cdot \mathbf{1}$. Since $\mathbf{diag}\left(z\right) \succeq 0$, this yields the claim. □

It remains to implement a low-pass and low-space optimization procedure to minimize g_μ. We make use of a variant of the box-constrained Newton's method of [145] stated below.

Definition 54. *We say that a procedure $\mathcal{O}$ is an (r, k)-oracle for a class of matrices $\mathcal{M}$ if, for any $\mathbf{A} \in \mathcal{M}$ and vector b, $\mathcal{O}(\mathbf{A}, b)$ returns a vector z where $\|z\|_\infty \leq rk$ and*

$$z^\top b + \frac{1}{2} z^\top \mathbf{A} z \leq \frac{1}{2}\left(\min_{\|y\|_\infty \leq r} y^\top b + \frac{1}{2} y^\top \mathbf{A} y\right).$$

The proof of the following is patterned from [145], but tolerates multiplicative error in the Hessian.

[7]Here, we use the notation x_L to refer to the vector x restricted to the entries in $L \subseteq V$.

Lemma 263. *Let f be r-second order robust with minimizer $x^\star$. For some x, let $\mathbf{M}$ be such that $\frac{1}{2}\nabla^2 f(x) \preceq \mathbf{M} \preceq 2\nabla^2 f(x)$. Let $\mathcal{O}$ be an (r,k)-oracle for a class of matrices $\mathcal{M}$ where $\mathbf{M} \in \mathcal{M}$. Then if $\max\{\|x^\star\|_\infty, \|x\|_\infty\} \leq R$ and $R \geq r$, for any vector x, $x' = x + \frac{1}{k}\mathcal{O}\left(\frac{1}{k}\nabla f(x), \frac{2e}{k^2}\mathbf{M}\right)$ satisfies*

$$f(x') - f(x^\star) \leq \left(1 - \frac{r}{160kR}\right)(f(x) - f(x^\star)).$$

Proof. We note that for any y where $\|y - x\|_\infty \leq r$ we have

$$f(y) = f(x) + \nabla f(x)^\top (y - x) + \int_0^1 \int_0^\beta (y - x)^\top \nabla^2 f(\alpha x + (1 - \alpha)y)(y - x) d\alpha d\beta.$$

As the integral is over a triangle of area $\frac{1}{2}$, the second order robustness of f yields

$$\begin{aligned} f(x) + \nabla f(x)^\top (y - x) + \frac{1}{4e}(y - x)^\top \mathbf{M}(y - x) &\leq f(y), \\ f(y) &\leq f(x) + \nabla f(x)^\top (y - x) + e(y - x)^\top \mathbf{M}(y - x). \end{aligned} \tag{D.17}$$

Let

$$\hat{\delta} = \operatorname{argmin}_{\|\delta\|_\infty \leq r} \frac{1}{k}\nabla f(x)^\top \delta + \frac{e}{k^2}\delta^\top \mathbf{M}\delta.$$

Also, let $\Delta = \mathcal{O}(\frac{1}{k}\nabla f(x), \frac{2e}{k^2}\mathbf{M})$. By definition of $\mathcal{O}$ we have $\|\Delta\|_\infty \leq rk$ and

$$\frac{1}{k}\nabla f(x)^\top \Delta + \frac{e}{k^2}\Delta^\top \mathbf{M}\Delta \leq \frac{1}{2}\left(\frac{1}{k}\nabla f(x)^\top \hat{\delta} + \frac{e}{k^2}\hat{\delta}^\top \mathbf{M}\hat{\delta}\right).$$

Thus since $x' = x + \frac{1}{k}\Delta$ we see $\|x' - x\|_\infty \leq r$, and hence

$$f(x') \leq f(x) + \frac{1}{k}\nabla f(x)^\top \Delta + \frac{e}{k^2}\Delta^\top \mathbf{M}\Delta \leq f(x) + \frac{1}{2}\left(\frac{1}{k}\nabla f(x)^\top \hat{\delta} + \frac{e}{k^2}\hat{\delta}^\top \mathbf{M}\hat{\delta}\right). \tag{D.18}$$

Define $\tilde{x} = \frac{r}{2R}(x^\star - x)$, we note that $\|\tilde{x}\|_\infty \leq \frac{r}{2R}(\|x^\star\|_\infty + \|x\|_\infty) \leq r$. By the minimality of $\hat{\delta}$ we observe that $\hat{\delta}$ achieves a smaller value than $c\widetilde{x}$ for any $c \leq 1$. For the choice $c = \frac{1}{4e^2}$ this implies

$$\begin{aligned} \frac{1}{2}\left(\frac{1}{k}\nabla f(x)^\top \hat{\delta} + \frac{e}{k^2}\hat{\delta}^\top \mathbf{M}\hat{\delta}\right) &\leq \frac{1}{2}\left(\frac{c}{k}\nabla f(x)^\top \tilde{x} + \frac{c^2 e}{k^2}\tilde{x}^\top \mathbf{M}\tilde{x}\right) \\ &= \frac{1}{8e^2}\left(\frac{1}{k}\nabla f(x)^\top \tilde{x} + \frac{1}{4ek^2}\tilde{x}^\top \mathbf{M}\tilde{x}\right) \\ &\leq \frac{1}{8e^2}\left(f\left(x + \frac{1}{k}\tilde{x}\right) - f(x)\right) \\ &\leq -\frac{r}{16ke^2 R}\left(f(x) - f(x^\star)\right), \end{aligned}$$

where the third line used (D.17) and the last used convexity of f. Plugging this into (D.18) yields

$$f(x') - f(x^\star) \leq \left(1 - \frac{r}{16ke^2R}\right)(f(x) - f(x^\star)).$$

□

Next, the class of matrices which are Laplacians plus a nonnegative diagonal admit an efficient $\mathcal{O}$.

Lemma 264 (Theorem 5.11, [145]). *Let $\mathcal{M}$ be the class of matrices which consist of a Laplacian matrix plus a nonnegative diagonal. Let $\mathbf{A} \in \mathcal{M}$ be a matrix with m nonzero entries. There is an algorithm which runs in $\widetilde{O}(m)$ time and space and implements an $(r, O(\log n))$-oracle for $\mathbf{A}$.*

We complement this with a known semi-streaming spectral approximation for Laplacians.

Lemma 265 (Section 2.2, [388]). *Let $G = (V, E, w)$ be a weighted undirected graph, given as an insertion-only stream. There is an algorithm which for any $\varepsilon \in (0, 1)$ takes one pass and returns a graph H with $O(\varepsilon^{-2} n \log^3(n))$ edges such that $\mathcal{L}_H \preceq \mathcal{L}_G \preceq (1+\varepsilon)\mathcal{L}_H$ using $\widetilde{O}(m)$ work and $O(\varepsilon^{-2} n \log^3 n)$ space.*

Algorithm 74: BoxConstrainedVC$(G, \varepsilon, \mathcal{O})$

Input: Bipartite graph $G = (V, E)$ with vertex partition $V = L \cup R$ and oriented edge-incidence matrix $\widetilde{\mathbf{B}}$ given as a stream, $\varepsilon > 0$, $(O(\frac{\varepsilon}{\log(n)}), k)$-oracle $\mathcal{O}$ for symmetric diagonal dominant matrices with nonpositive off diagonal ;
Output: v fractional vertex cover for G;
$\mathbf{R} := \begin{bmatrix} \mathbf{I}_L & 0 \\ 0 & -\mathbf{I}_R \end{bmatrix}$;
$x_0 \leftarrow \mathbf{R}\mathbf{1}$, $\mu = \frac{\varepsilon}{4\log(mn)}$, $T = O(\frac{\log \mu^{-1}}{\mu k})$;
$f_\mu(x) := \mathbf{1}^\top x + \mu \left(\sum_{e \in E} e^{\frac{1}{\mu}(1 - [\mathbf{B}x]_e)} + \sum_{i \in V} e^{-\frac{1}{\mu}(x_i + \frac{\varepsilon}{4})} \right)$;
$g_\mu(x) := f_\mu(\mathbf{R}x)$;
for $0 \leq t < T$ **do**
  Compute $\nabla g_\mu(x_t)$ and $\mathbf{M}$, a 2-approximation of $\nabla^2 g_\mu(x_t)$ with Lemma 265;
  $x'_{t+1} = x_t + \frac{1}{k}\mathcal{O}\left(\frac{1}{k}\nabla g_\mu(x_t), \frac{2e}{k^2}\mathbf{M}\right)$;
  $x_{t+1} = \max\{\min\{x'_{t+1}, 2 \cdot \mathbf{1}\}, -2 \cdot \mathbf{1}\}$ entrywise;
Return: $v = \mathbf{R}x_T + \frac{\varepsilon}{2}\mathbf{1}$;

We now assemble these claims to prove the correctness of our algorithm.

Proposition 61. *Let $G = (V, E)$ be an unweighted bipartite graph with bipartition L, R. Given access to a $(O(\frac{\varepsilon}{\log n}), k)$-oracle $\mathcal{O}$ for $\nabla^2 g_\mu$, Algorithm 74 computes v, a feasible vertex cover of size $M^* + \varepsilon n$. For $y := \exp\left(\frac{1}{\mu}(1 - \mathbf{B}x)\right)$, there exists w with $\|w\|_1 \leq \mu n$ so $y - w$ is a feasible matching*

with $\mathbf{1}^\top y \geq M^* - 5\varepsilon n$. *Algorithm 74 requires* $\widetilde{O}(n)$ *auxiliary space and* $O(\frac{k \log \mu^{-1}}{\mu})$ *passes, each requiring* $\widetilde{O}(m)$ *work, plus the work and space required by one* $\mathcal{O}$ *call.*

As the space used by $\mathcal{O}$ can be reused between runs, the space overhead will be $\widetilde{O}(n)$ throughout.

Proof. By Lemmas 261 and 262, $\mathcal{O}$ applies to a matrix family containing the Hessian $\nabla^2 g_\mu$. In addition, we see that ∇g_μ may be computed in a single pass since it equals

$$\nabla g_\mu(x) = \mathbf{R}\left(\mathbf{1} - z\right) - \widetilde{\mathbf{B}}^\top y$$

for $z = [z_L, z_R]$ and z_L, z_R, y as in Lemma 262. The first term may be computed directly, while the second can be computed analogously to Lemma 69. Further, we can obtain a 2-spectral approximation of $\nabla^2 g_\mu(x)$ in semi-streaming fashion: we compute z in one pass, and sparsify the Laplacian using Lemma 265 while computing y coordinatewise in one pass. Thus in each iteration we perform one pass and one call to $\mathcal{O}$: as there are $O(\frac{k \log \mu^{-1}}{\mu})$ iterations: the claimed space, pass, and work bounds follow.

We now prove the correctness of the algorithm. Let $x^\star$ be the minimizer of g_μ. By construction, $g_\mu(x_0) = f_\mu(\mathbf{R}x_0) \leq 2n \leq m$ and the function is monotone decreasing. We note that the point x_t has $\|x_t\|_\infty \leq 2$ by construction for all t, and that $\|x^\star\|_\infty \leq 2$ by the fourth condition of Lemma 261. Further, by Lemma 262 we see that is r-second order robust with $r = \frac{\mu}{2}$. By Lemma 263 we obtain

$$g_\mu(x'_{t+1}) - g_\mu(x^\star) \leq \left(1 - \frac{\mu}{640k}\right)\left(g_\mu(x_t) - g_\mu(x^\star)\right)$$

for any t. As the fourth condition of Lemma 261 implies $g_\mu(x_{t+1}) \leq g_\mu(x'_{t+1})$, the final x_T has

$$g_\mu(x_T) - g_\mu(x^\star) \leq \left(1 - \frac{\mu}{640k}\right)^T \left(g_\mu(x_0) - g_\mu(x^\star)\right) \leq \frac{\mu^3 n}{9e},$$

where the second inequality uses $g_\mu(x_0) \leq 2n$ and $g_\mu(x^\star) \geq 0$. Thus if M^* is the size of the minimum vertex cover of G, we obtain by Lemma 261 that $g_\mu(x_T) = f_\mu(\mathbf{R}x_T) \leq (1 + \frac{2\varepsilon}{3})M^* + \frac{\mu^3 n}{9e} \leq M^* + \frac{\varepsilon}{2}n$, since $M^* \leq \frac{n}{2}$ and $\mu \leq \varepsilon$. To complete the proof, the first condition of Lemma 261 implies $v = \mathbf{R}x + \frac{\varepsilon}{2}\mathbf{1}$ is nonnegative with $\mathbf{B}v \geq 1$ and $\mathbf{1}^\top v \leq M^* + \frac{\varepsilon}{2}n + \frac{\varepsilon}{2}n = M^* + \varepsilon n$: it is a feasible fractional matching as claimed.

For the second claim, we observe that for any δ with $\|\delta\|_\infty \leq \frac{\mu}{2}$ and $\hat{x} = x_T + \delta$,

$$\begin{aligned}
\frac{\mu^3 n}{9e} &\geq g_\mu(x_T) - g_\mu(x^\star) \geq g_\mu(x_T) - g_\mu(\hat{x}) \\
&\geq -\nabla g_\mu(x_T)^\top \delta - e\delta^\top \nabla^2 g_\mu(x_T)\delta \\
&\geq -\nabla g_\mu(x_T)^\top \delta - \frac{2e}{\mu}\delta^\top \left(\mathbf{diag}\left(\mathbf{B}^\top y\right) + \mathbf{diag}\left(z\right)\right)\delta;
\end{aligned}$$

the first line used optimality of $x^\star$, the second used second order robustness of g_μ, and the third used Lemma 262. We choose $\delta = -\frac{\mu^2}{4e}\mathbf{sign}\left(\nabla g_\mu(x_T)\right)$: this satisfies $\|\delta\|_\infty \leq \frac{\mu}{2}$ so

$$\frac{\mu^2}{4e}\left\|\nabla g_\mu(x_T)\right\|_1 - \frac{\mu^3}{8e}\left\|\mathbf{B}^\top y + z\right\|_1 \leq \frac{\mu^3 n}{9e}.$$

Now note

$$\begin{aligned}\left\|\mathbf{B}^\top y + z\right\|_1 &= \left\|[\widetilde{\mathbf{B}}^\top y]_L + z_L\right\|_1 + \left\|-[\widetilde{\mathbf{B}}^\top y]_R + z_R\right\|_1 \\ &\leq \left\|[\widetilde{\mathbf{B}}^\top y]_L + z_L - \mathbf{1}_L\right\|_1 + \left\|-[\widetilde{\mathbf{B}}^\top y]_R + z_R - \mathbf{1}_R\right\|_1 + n = \left\|\nabla g_\mu(x_T)\right\|_1 + n.\end{aligned}$$

Plugging this in to the above expression and rearranging, we obtain

$$\frac{\mu^2}{4e}\left\|\nabla g_\mu(x_T)\right\|_1 \leq \frac{\mu^3 n}{9e} + \frac{\mu^3 n}{8e} + \frac{\mu^3}{8e}\left\|\nabla g_\mu(x_T)\right\|_1,$$

so $\left\|\nabla g_\mu(x_T)\right\|_1 \leq \mu n$. This implies the existence of w with $\|w\|_1 \leq \mu n$ where $\mathbf{1} + w = \mathbf{B}^\top y + z$. Since z is nonnegative, $\mathbf{B}^\top y \leq \mathbf{1} + w$ with $\|w\|_1 \leq \mu n$ as desired. We finally lower bound $\mathbf{1}^\top y$. Let $v' = \mathbf{R}x - \mu\mathbf{1}$. Taking the inner product of $\mathbf{1} + w = \mathbf{B}^\top y + z$ with v' and rearranging gives

$$\mathbf{1}^\top x_L - \mathbf{1}^\top x_R - \mu n - \mathbf{1}^\top y + w^\top v' = y^\top(\widetilde{\mathbf{B}}x - \mathbf{1}) - \mu\mathbf{1}^\top y + x^\top\mathbf{R}z - \mu\mathbf{1}^\top z,$$

or

$$f_\mu(\mathbf{R}x) - \mu n + w^\top v' = y^\top(\widetilde{\mathbf{B}}x - 1) + \mathbf{1}^\top y + x^\top\mathbf{R}z.$$

Next, $w^\top v' \geq -\|w\|_1\|v'\|_\infty \geq -2\mu n$. Further, since $y = \exp\left(\frac{1}{\mu}(1 - \widetilde{\mathbf{B}}x)\right)$, for any i either $[\widetilde{\mathbf{B}}x]_i - 1 \leq \frac{\varepsilon}{2}$ or $y_i \leq \exp\left(-\frac{1}{\mu}\frac{\varepsilon}{2}\right) = \frac{\varepsilon}{2mn}$. Thus $y^\top(\widetilde{\mathbf{B}}x - 1) \leq \varepsilon\mathbf{1}^\top y + \frac{\varepsilon}{2}M^*$. Similarly, we obtain $x^\top\mathbf{R}z \leq \varepsilon n$. Plugging these in, and using $\mu \leq \frac{\varepsilon}{4}$ and $M^* \leq n$ yields

$$f_\mu(\mathbf{R}x) - 3\mu n - \varepsilon n - \frac{\varepsilon}{2}M^* \leq (1+\varepsilon)\mathbf{1}^\top y \implies \mathbf{1}^\top y \geq \frac{1}{1+\varepsilon}f_\mu(\mathbf{R}x) - 3\varepsilon n.$$

Here we used $\mu \leq \frac{\varepsilon}{4}$ and $M^* \leq n$. The claim follows from $f_\mu(\mathbf{R}x) \geq M^* - \varepsilon n$ via Lemma 261. □

Proof of Theorem 84. Let M^* be the size of the maximum matching in G. We first preprocess the graph G using Proposition 15 within the pass, space, and work budgets to reduce to $n = O(M^*\log(\epsilon^{-1}))$. Applying Algorithm 74 to $\widetilde{G}$ with $\varepsilon = \frac{\epsilon}{12\log(\epsilon^{-1})}$, by the second claim in Proposition 61 we obtain $y \in \mathbb{R}^E_{\geq 0}$ with $\mathbf{1}^\top y \geq (1 - \frac{\epsilon}{2})M^* - 5\varepsilon n \geq (1 - O(\epsilon))M^*$, and

$$\sum_{j\in V}\max\left([\mathbf{B}^\top y]_j - 1, 0\right) \leq \mu n \leq \frac{\epsilon}{12}M^*.$$

The conclusion follows upon applying Propositions 16 and 73 to round the approximate matching y

to be sparse and feasible, and using the implementation of $\mathcal{O}$ from Lemma 258. □

Appendix E

Deferred proofs from Chapter 6

E.1 Proofs for Section 6.3

E.1.1 Proofs for Section 6.3.1

Proposition 18. *Let $\epsilon \in (0,1)$ and $M \geq 0$. Given a family of (ϵ, β)-CROs $\{f_{M,E}\}$ for $G = (V, E_0)$, and an $(\epsilon, \mathcal{T})$-canonical solver for the family, Algorithm 20 satisfies the following.*

1. *When $M_{\text{est}} > \frac{1}{4}M$ on Line 7, where M_{est} estimates $\text{MCM}(E_k)$: at any point in the loop of Lines 9 to 10, $8M\tilde{x}^{E_k}_{E_k \setminus E_{\text{del}}}$ is an ϵ-approximate matching of $G(V, E_k \setminus E_{\text{del}})$.*
2. *When $M_{\text{est}} \leq \frac{1}{4}M$ on Line 7, where M_{est} estimates $\text{MCM}(E)$: $\text{MCM}(E) \leq \frac{1}{2}\text{MCM}(E_0)$.*

The runtime of the algorithm is $O(m + (\mathcal{T} + m) \cdot \frac{M}{\beta\epsilon})$.

Proof. Throughout this proof, let the edge sets recomputed by Line 11 be denoted $E_1, E_2, \ldots, E_K$ and assume the termination condition Line 7 breaks the loop for E_{K+1}, where E_0 is the original edge set of the graph. Before proving the two claims, we first observe that for all $k \in [K]$, $\text{MCM}(E_k) \geq \frac{M}{8}$. The base case of $k = 0$ clearly holds because of the approximation guarantee of the greedy algorithm, which yields $M \geq \frac{1}{2}\text{MCM}(E)$. Now, for any $k \in [K]$, let M^k_{est} be the greedily computed estimate of $\text{MCM}(E_k)$. We conclude by

$$\text{MCM}(E_k) \geq M^k_{\text{est}} > \frac{1}{4}M \geq \frac{1}{8}\text{MCM}(E).$$

Matching approximation. For any $0 \leq k \leq K$, by item 1 of the CRO definition applied to E_k, we have $(1 + \frac{\epsilon}{8})\text{MCM}(E_k) \geq -\nu^{E_k} \geq (1 - \frac{\epsilon}{8})\text{MCM}(E_k)$, and hence combining with (6.11a) and (6.12a)

implies

$$-f_{M,E_k}(\hat{x}^{E_k}) \geq -\left(1-\frac{\epsilon}{8}\right)\nu^{E_k} \geq \left(1-\frac{\epsilon}{8}\right)^2 \frac{1}{4}M \geq \frac{3}{16}M, \tag{E.1}$$

$$-f_{M,E_k}(\tilde{x}^{E_k}) \geq -\left(1-\frac{\epsilon}{8}\right)\nu^{E_k} \geq \left(1-\frac{\epsilon}{8}\right)^2 \frac{1}{4}M \geq \frac{3}{16}M. \tag{E.2}$$

By the definition of the CRO in (6.10) and feasiblity of the matching $8M\tilde{x}^{E_k}$ (which follows by the assumed guarantees on Round), we have

$$8M\left\|\tilde{x}^{E_k}\right\|_1 \geq -f_{M,E_k}\left(\tilde{x}^{E_k}\right) - \frac{\epsilon}{128}M \geq \frac{5}{32}M \implies \left\|\tilde{x}^{E_k}\right\|_1 \geq \frac{5}{8\cdot 32}. \tag{E.3}$$

Also, following the above calculation and (E.2) we have that

$$8M\left\|\tilde{x}^{E_k}\right\|_1 \geq -f_{M,E_k}\left(\tilde{x}^{E_k}\right) - \frac{\epsilon}{128}M \geq \left(1-\frac{\epsilon}{8}\right)^2 \mathrm{MCM}(E_k) - \frac{\epsilon}{32}M \geq \left(1-\frac{\epsilon}{2}\right)\mathrm{MCM}(E_k)$$

where for the last inequality we use $\mathrm{MCM}(E_k) \geq \frac{1}{4}M$, as shown in the first part of this proof. Consequently, considering any E between recomputations satisfying $E_{k+1} \subset E \subseteq E_k$ such that $E = E_k \setminus E_{\mathrm{del}}$, we have by the condition on Line 9 that

$$\begin{aligned} 8M\left\|\tilde{x}^{E_k}_{E_k\setminus E_{\mathrm{del}}}\right\|_1 &= 8M\left\|\tilde{x}^{E_k}\right\|_1 - 8M\left\|\tilde{x}^{E_k}_{E_{\mathrm{del}}}\right\|_1 \geq \left(1-\frac{\epsilon}{8}\right)8M\left\|\tilde{x}^{E_k}\right\|_1 \\ &\geq \left(1-\frac{\epsilon}{8}\right)\left(1-\frac{\epsilon}{2}\right)\mathrm{MCM}(E_k) \geq (1-\epsilon)\,\mathrm{MCM}(E_k\setminus E_{\mathrm{del}}), \end{aligned}$$

which combined with feasibility of $8M\tilde{x}^{E_k}$ (and hence, feasibility of any restriction) implies that $8M\tilde{x}^{E_k}_{E_k\setminus E_{\mathrm{del}}}$ is always an ϵ-approximate matching of current graph.

Multiplicative matching decrease. When the algorithm terminates after $K+1$ loops, using the termination condition, and the guarantee of M_{est}, we have

$$\mathrm{MCM}(E_{K+1}) \leq 2M_{\mathrm{est}} \leq \frac{1}{2}M \leq \frac{1}{2}\mathrm{MCM}(E_0).$$

Hence, the MCM value decreases by a factor of at least 2.

Iteration bound. We next show that the algorithm will stop after $K+1$ loops (from Line 9

to Line 10) for bounded K. For some loop $0 \le k \le K-1$, letting $E_{k+1} = E_k \setminus E_{\mathrm{del}}$,

$$\begin{aligned}
&f_{M,E_{k+1}}\left(x^{E_{k+1}}\right) - f_{M,E_k}\left(x^{E_k}\right) \overset{(i)}{\ge} \beta V^H_{x^{E_k}}\left(x^{E_{k+1}}\right) \\
&\overset{(ii)}{\ge} \beta \sum_{i \in E_{\mathrm{del}}} \left([x^{E_{k+1}}]_i \log[x^{E_{k+1}}]_i - [x^{E_k}]_i \log[x^{E_k}]_i - \left(1 + \log[x^{E_k}]_i\right) \cdot \left([x^{E_{k+1}}]_i - [x^{E_k}]_i\right)\right) \\
&\overset{(iii)}{=} \beta \sum_{i \in E_{\mathrm{del}}} [x^{E_k}]_i \overset{(iv)}{\ge} \beta \left(\left\|\hat{x}^{E_k}_{E_{\mathrm{del}}}\right\|_1 - \left\|\hat{x}^{E_k} - x^{E_k}\right\|_1\right) \\
&\overset{(v)}{\ge} \beta \left(\left\|\tilde{x}^{E_k}_{E_{\mathrm{del}}}\right\|_1 - \left\|\hat{x}^{E_k} - x^{E_k}\right\|_1\right),
\end{aligned} \tag{E.4}$$

where we used (i) the assumption (6.9), (ii) convexity of the scalar function $c \log c$ allowing us to restrict the divergence to a subset, (iii) the property that $[x^{E_{k+1}}]_i = 0$ on all deleted edges by Item 2 in Definition 23, (iv) the triangle inequality, and (v) the monotonicity property (6.12b).

We now proceed to bound $\|\tilde{x}^{E_k}_{E_{\mathrm{del}}}\|_1$ and $\left\|\hat{x}^{E_k} - x^{E_k}\right\|_1$. By the termination condition on Line 9, $\|\tilde{x}^{E_k}_{E_{\mathrm{del}}}\|_1 \ge \frac{\epsilon}{8}\left\|\tilde{x}^{E_k}\right\|_1$. Moreover,

$$\left\|\hat{x}^{E_k} - x^{E_k}\right\|_1 \le \frac{\epsilon}{1000} \le \frac{\epsilon}{16}\left\|\tilde{x}^{E_k}\right\|_1,$$

where we used the ℓ_1 bound in (6.11b) and the lower bound on $\left\|\tilde{x}^{E_k}\right\|_1$ implied by (E.3).

Plugging these two bounds back in (E.4), and once again using (E.3), we thus have

$$f_{M,E_{k+1}}\left(x^{E_{k+1}}\right) - f_{M,E_k}\left(x^{E_k}\right) \ge \frac{\beta\epsilon}{16}\left\|\tilde{x}^{E_k}\right\|_1 = \Omega(\beta\epsilon).$$

At the beginning of the algorithm, recall by $(1+\frac{1}{8})\mathrm{MCM}(E_0) \ge -\nu^{E_k} = -f_{M,E_0}(x^{E_0})$ due to Definition 23 we have $-f_{M,E_0}(x^{E_0}) \le 2.25M$. Similarly, due to Definition 23 we also have for E_K, it holds that $-\nu^{E_K} \ge (1-\frac{1}{8})\mathrm{MCM}(E_K) \ge \frac{7}{32}M$. Thus, this shows $K \le O\left(\frac{M}{\beta\epsilon}\right)$ and thus the algorithm must terminate within $O(\frac{M}{\beta\epsilon})$ loops. This implies the runtime bound after combining with the cost of the greedy approximation. □

E.1.2 DDBM via regularized box-simplex games

Throughout this section, let $\epsilon \in (0,1)$, and assume that the DDBM problem is initialized with bipartite $G = (V, E_0)$ with unsigned incidence matrix $\mathbf{B} \in \{0,1\}^{E \times V}$; we denote $n := |V|$ and $m := |E_0|$. For $E \subseteq E_0$ and M, an 8-approximation to $\mathrm{MCM}(E)$, we consider the following

regularized box-simplex game approximating the matching problem:

$$\min_{(x,\xi)\in\Delta^{E+1}} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y) := -\mathbf{1}_E^\top(8Mx) - y^\top\left(8M\mathbf{B}^\top x - \mathbf{1}\right) + \gamma^{\mathsf{x}} H(x,\xi) + \gamma^{\mathsf{y}}\left(y^2\right)^\top \mathbf{B}^\top x,$$
$$\text{where } \gamma^{\mathsf{x}} = \frac{\epsilon M}{256\log(m)},\ \gamma^{\mathsf{y}} = \frac{\epsilon M}{256}. \tag{E.5}$$

We recall a rounding procedure from [39]; we restate its guarantees below for completeness.

Lemma 266 (Rounding)**.** *Given bipartite (V,E),* RemoveOverflow *(Algorithm 4, [39]) takes $\ell \in \mathbb{R}^E_{\geq 0}$, runs in $O(|E|)$ time, and outputs $\tilde{\ell}$ satisfying $\tilde{\ell} \leq \ell$ entrywise, $\mathbf{B}^\top\tilde{\ell} \leq \mathbf{1}$, and*

$$\left\|\tilde{\ell}\right\|_1 \geq \|\ell\|_1 - \sum_{i\in V}\left([\mathbf{B}^\top\ell - 1]_i\right)_+.$$

Next, we show that the set of $f_{M,E}(x) = \min_{\xi|(x,\xi)\in\Delta^{E+1}} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y)$ forms a family of $(\epsilon,\gamma^{\mathsf{x}})$-CROs.

Lemma 267 (CROs from regularized box-simplex games)**.** *Define (overloading notation)*

$$f_{M,E}(x) := \min_{\xi|(x,\xi)\in\Delta^{E+1}} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y).$$

Then, the family of $f_{M,E}(x)$ is a family of $(\epsilon,\gamma^{\mathsf{x}})$-CROs.

Proof. We prove each property in turn. Item 2 (restricting coordinates to zero) is immediate from the problem definition, so we focus on showing the other three.

Multiplicative value approximation (Item 1). By assumption, M is an 8-multiplicative approximation to $\mathrm{MCM}(E)$, so there is a maximum matching $8Mx^E_\star$ such that $\left\|x^E_\star\right\|_1 \leq 1$. For such $x^E_\star$ and $\xi := 1 - \left\|x^E_\star\right\|_1$, and because the entropic and quadratic regularization terms are nonpositive,

$$\begin{aligned}\nu^E &\leq \max_{y\in[0,1]^V} f_{M,E}(x^E_\star,\xi,y)\\ &\leq \max_{y\in[0,1]^V} -\mathbf{1}_E^\top(8Mx^E_\star) - y^\top\left(8M\mathbf{B}^\top x^E_\star - \mathbf{1}\right) = -\mathrm{MCM}(E).\end{aligned}$$

The last equality holds because we assumed $8Mx^E_\star$ was a maximum matching. For the other direction, we proceed by contradiction. Suppose we have a feasible (x,ξ) with

$$\max_y f_{M,E}(x,\xi,y) < -\left(1+\frac{\epsilon}{4}\right)\mathrm{MCM}(E).$$

Applying RemoveOverflow on the approximate matching $\ell := 8Mx$ yields $\tilde{\ell} := 8M\tilde{x}$ such that

$$\begin{aligned}-\mathrm{MCM}(E) \le -\left\|8M\tilde{x}\right\|_1 &\le -\left\|8Mx\right\|_1 + \left\|8M\mathbf{B}^\top x - \mathbf{1}\right\|_1 \\ &\overset{(i)}{\le} \max_{y\in[0,1]^V} f_{M,E}(x,\xi,y) + \frac{\epsilon M}{128} \\ &\overset{(ii)}{<} -\left(1+\frac{\epsilon}{4}\right)\mathrm{MCM}(E) + \frac{\epsilon}{16}\mathrm{MCM}(E) < -\mathrm{MCM}(E),\end{aligned}$$

where we used (i) the definition of $f_{M,E}$ and bounds on γ^{x}, γ^{y} leading to an additive range of at most $\frac{\epsilon M}{128}$, and (ii) that $M \le 8\mathrm{MCM}(E)$ by assumption, leading to a contradiction. Hence, combining these bounds yields Item 1.

Relative strong convexity (Item 3). For simplicity, we drop the subscript M. Let (x^E, ξ^E, y^E) and $(x^{E'}, \xi^{E'}, y^{E'})$ be the unique optimal solutions of f_E and $f_{E'}$ respectively. By the first order optimality condition of (x^E, ξ^E), we have

$$f_{E'}(x^{E'}, \xi^{E'}, y^E) - f_E(x^E, \xi^E, y^E) \ge \int_0^1 (1-\alpha) v^\top \nabla^2_{(x,\xi)(x,\xi)} f(x_\alpha, \xi_\alpha, y^E) v d\alpha,$$

where we let $v := (x^{E'} - x^E, \xi^{E'} - \xi^E)$, $x_\alpha := \alpha x^{E'} + (1-\alpha)x^E$, and $\xi_\alpha := \alpha\xi^{E'} + (1-\alpha)\xi^E$. By joint convexity of f as a function of (x,ξ) (with y fixed),

$$\begin{aligned}\int_0^1 (1-\alpha) v^\top \nabla^2_{(x,\xi)(x,\xi)} f(x_\alpha, \xi_\alpha, y^E) v d\alpha &\ge \int_0^1 (1-\alpha) v^\top \nabla^2_{xx} f(x_\alpha, \xi_\alpha, y^E) v d\alpha \\ &= \gamma^{\mathsf{x}} V^H_{x_E}(x^{E'}).\end{aligned}$$

Here we let $\nabla^2_{xx} f$ zero out blocks of $\nabla^2_{(x,\xi)(x,\xi)} f$ appropriately. The conclusion (6.9) follows since $f_{E'}(x^{E'}, \xi^{E'}, y^{E'}) \ge f_{E'}(x^{E'}, \xi^{E'}, y^E)$ by optimality of $y^{E'}$.

Additive approximation (Item 4). It suffices to note that for any feasible matching $8Mx$, the value of $f_{M,E}$ without the added entropic and quadratic regularization terms is exactly $-8M\left\|x_E\right\|_1$, since $8M\mathbf{B}^\top x \le \mathbf{1}$ entrywise. Using standard bounds on the ranges of $H(x,\xi) \in [-\log(m+1), 0)$, $\left(y^2\right)^\top \mathbf{B}^\top x \in [0,1]$, we have the additive approximation (6.10) due to the values of γ^{x} and γ^{y}. □

We now apply our high-accuracy box-simplex solver RegularizedBS (Theorem 41), together with our rounding procedure RemoveOverflow, to develop a $(\epsilon, \widetilde{O}(m\epsilon^{-1}))$-canonical solver for the CRO family induced by the regularized objectives (E.5).

Lemma 268 (Canonical solver for regularized box-simplex games)**.** *For $\epsilon = \Omega(m^{-3})$, there is an $(\epsilon, \mathcal{T})$-canonical solver for the family of $f_{M,E}$ defined in Lemma 267, using* RegularizedBS *(Algorithm 23) as* Solve *and* RemoveOverflow *as* Round*, achieving $\mathcal{T} = O\left(m\epsilon^{-1}\log^3 n\right)$.*

Proof. We define x^E as in (6.8). Solve applies the procedure in Theorem 41 to solve $\frac{1}{16M} f_{M,E}$ to

additive accuracy $O(\frac{\epsilon^3}{\log m})$ in the stated runtime, for a sufficiently small constant. By the multiplicative approximation guarantee on ν^E and MCM(E) in Definition 23, and since MCM(E) is 8-approximated by M, we obtain the guarantee (6.11a). The ℓ_1 bound (6.11b) then follows since the objective is $\Omega(\frac{\epsilon}{\log m})$-strongly convex in the x coordinates.

To show that RemoveOverflow is a valid choice of Round, the monotonicity and feasibility conditions follow immediately from Lemma 266. Moreover, for $\ell \leftarrow 8M\hat{x}^E$, the definition of $f_{M,E}$ and the guarantees of Lemma 266 imply

$$\begin{aligned}\left\|\tilde{\ell}\right\|_1 &\geq \|\ell\|_1 - \sum_{i\in V}\left(\left[\mathbf{B}^\top \ell - 1\right]\right)_+ \\ &= \min_{y\in[0,1]^V} \mathbf{1}_E^\top(8M\hat{x}^E) - y^\top\left(8M\mathbf{B}^\top\hat{x}^E - \mathbf{1}\right) \geq -\nu^E - O\left(\frac{\epsilon^3 M}{\log m}\right) - \frac{\epsilon M}{128}.\end{aligned}$$

Combined with our approximation guarantees between ν^E, MCM(E), and M via Items 1 and 4 of Definition 23, this implies (6.12a), showing all required properties of RemoveOverflow. □

Theorem 38. *Let $G = (V, E)$ be bipartite and let $\epsilon \in [\Omega(m^{-3}), 1)$. There is a deterministic algorithm for the DDBM problem which maintains an ϵ-approximate matching, based on solving regularized box-simplex games, running in time $O\left(m\epsilon^{-3}\log^5 n\right)$.*

Proof. It suffices to combine Lemma 267, Lemma 268, and Corollary 17. □

E.1.3 DDBM via matrix scaling and box-constrained Newton's method

We follow the same notational conventions as in Appendix E.1.2, and further assume $|L| \leq |R|$ without loss of generality. Moreover, assume for simplicity that both L and R have at least one vertex which has no adjacent edges, denoted $v_L^\star$ and $v_R^\star$; this clearly will not affect any matching sizes if we add such vertices. We will further expand the graph $G = (V, E)$ to a new graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, with bipartition $\widetilde{V} = \widetilde{L} \cup \widetilde{R}$. In particular, $\widetilde{L}$ consists of the original left vertices L, a new dummy vertex set L_0 such that $|L| + |L_0| = |R|$, and an extra dummy vertex v_L^{dum}. Similarly, $\tilde{R}$ consists of the original right vertices R and a new dummy vertex v_R^{dum}. The new edge set is

$$\widetilde{E} := E \cup \left(\bigcup_{v\in L\cup L_0}(v, v_R^{\mathsf{dum}})\right) \cup \left(\bigcup_{v'\in R}(v', v_L^{\mathsf{dum}})\right) \cup \left(v_L^{\mathsf{dum}}, v_R^{\mathsf{dum}}\right).$$

We define $\widetilde{\mathbf{B}} \in \{0,1\}^{\widetilde{E}\times\widetilde{V}}$ such that $\widetilde{\mathbf{B}}_{e,v} = 1$ whenever $v \in e$. To instantiate our matrix scaling solver for approximating Sinkhorn distance in Appendix E.3.2, we construct a demand vector $d \in \mathbb{R}^{\widetilde{V}}_{\geq 0}$ and a cost vector $c \in \mathbb{R}^{\widetilde{E}}$ as follows.

1. Set $d_v = 1$ for all $v \in L \cup L_0 \cup R$.
2. Set $d_{v_L^{\mathsf{dum}}} = d_{v_R^{\mathsf{dum}}} = |R|$.

3. Set $c_e = -1$ for all $e \in E$, and $c_e = 0$ for all $\widetilde{E} \setminus E$.

For a vector in $\mathbb{R}^{\widetilde{E}}_{\geq 0}$, we refer to its restrictions to the sets E and $\widetilde{E} \setminus E$ by x and x^{dum} when clear from context. It is clear that any matching $2|R|x \in \mathbb{R}^{E}_{\geq 0}$ on the original graph (V, E) can be extended to a matching

$$2|R|x^{\mathsf{tot}} = 2|R| \begin{pmatrix} x \\ x^{\mathsf{dum}} \end{pmatrix} \in \mathbb{R}^{\widetilde{E}}_{\geq 0}$$

such that $\|x^{\mathsf{tot}}\|_1 = 1$, and $2|R|\widetilde{\mathbf{B}}^\top x^{\mathsf{tot}} = d$, by placing additional flow on the edges in $\widetilde{E} \setminus E$. Specifically, we will extend any flow on $(v, v') \in E$ to put the same amount of flow on (v', v_L^{dum}), (v, v_R^{dum}), and $(v_L^{\mathsf{dum}}, v_R^{\mathsf{dum}})$, and then for any additional demand not routed to a vertex $v \in L \cup L_0 \cup R$ by the original edges E, we route it arbitrarily over the additional edges $\widetilde{E}$ (similarly extending this flow to v_L^{dum} and v_R^{dum}). This is always feasible, as we can route to $v_L^\star$ and $v_R^\star$. The total ℓ_1 norm of d is $4|R|$ by construction, and every unit of x^{tot} contributes two units of demand, thus $2|R| \|x^{\mathsf{tot}}\| = 2|R|$ and rearrangement gives the ℓ_1 guarantee. We then consider the Sinkhorn distance objective

$$\min_{x^{\mathsf{tot}} = \binom{x}{x^{\mathsf{dum}}} \in \mathbb{R}^{\widetilde{E}}_{\geq 0} | \widetilde{\mathbf{B}}^\top (2|R|x^{\mathsf{tot}}) = d} f^{\mathsf{sink}}_{M,E}(x^{\mathsf{tot}}) := c^\top \left(2|R|x^{\mathsf{tot}}\right) + \gamma H\left(x^{\mathsf{tot}}\right) \text{ where } \gamma = \Theta\left(\frac{\epsilon M}{\log(m)}\right). \tag{E.6}$$

We show such objectives also form a family of CROs.

Lemma 269 (CROs from Sinkhorn distances)**.** *Define (overloading notation)*

$$f^{\mathsf{sink}}_{M,E}(x) = \min_{x^{\mathsf{dum}} \in \mathbb{R}^{E \setminus E_0}_{\geq 0}} f^{\mathsf{sink}}_{M,E}(x, x^{\mathsf{dum}}).$$

Then, the family of $f^{\mathsf{sink}}_{M,E}(x)$ is a family of (ϵ, γ)-CROs.

Proof. We focus on proving Item 1; Item 2 is immediate from the problem definition, and Items 3 and 4 follow analogously to the proof in Lemma 267.

Multiplicative value approximation (Item 1). By construction, any optimal matching $2|R|x_\star$ on the edge set E can be extended to $x^{\mathsf{tot}}_\star$ such that $\|x^{\mathsf{tot}}_\star\|_1 = 1$ and $\widetilde{\mathbf{B}}^\top(2|R|x^{\mathsf{tot}}) = d$. By nonpositivity of entropy and the definition of c, we obtain

$$\nu^E \leq f^{\mathsf{sink}}_{M,E}(x^E_\star) = -\mathrm{MCM}(E).$$

For the other direction, we proceed by contradiction. Suppose we have a feasible x^{tot} with $x := x^{\mathsf{tot}}_E$, so that $f^{\mathsf{sink}}_{M,E}(x^{\mathsf{tot}}) < -\left(1 + \frac{\epsilon}{4}\right)\mathrm{MCM}(E)$. This implies $2|R|x$ is a feasible matching, and then an analogous argument to the one used in Lemma 267 yields the contradiction. □

We next construct a canonical solver for the family of $f^{\mathsf{sink}}_{M,E}$, which makes use of the matrix scaling solver in Proposition 63 of Appendix E.3.2.

Lemma 270 (Canonical solver for Sinkhorn objectives)**.** *For $\epsilon = \Omega(m^{-3})$, there is an $(\epsilon, \mathcal{T})$-canonical solver for the family of $f^{\mathsf{sink}}_{M,E}$ defined in Lemma* (269)*, using the matrix scaling solver from [145] as* Solve *and truncation to E as* Round*, running in time $\widetilde{O}(n^2/\epsilon)$.*

Proof. Let r be the restriction of $\frac{d}{2|R|}$ to the vertices in $R \cup \{v_R^{\mathsf{dum}}\}$, and let ℓ be the restriction of $\frac{d}{2|R|}$ to the vertices in $L \cup L_0 \cup \{v_L^{\mathsf{dum}}\}$. Finally, let $\mathbf{K}$ be the $(|R|+1) \times (|R|+1)$ matrix with $\mathbf{K}_{ij} = \exp(-\frac{2|R|}{\gamma} c_{ij})$ for all indices corresponding to $(i,j) \in E$, and $\mathbf{K}_{ij} = 0$ otherwise. Lemma 2 of [151] shows that solving the (r,c)-matrix scaling problem on $\mathbf{K}$ computes the minimizer to (E.6).

We next apply Proposition 63, similarly to the proof of Theorem 86. Clearly $s_{\mathbf{K}} \le m$, and by Lemma 10 of [82], we may bound B by

$$O\left(\log(n) + \frac{|R|}{\gamma}\right) = \widetilde{O}\left(\frac{|R|}{\epsilon M}\right) = \widetilde{O}\left(\frac{n^2}{\epsilon m}\right).$$

Here, we used that $M = \Omega(\frac{m}{n})$ which follows from the fact that the minimum vertex cover size is the same as the maximum matching size. If there was a vertex cover of size $o(\frac{m}{n})$, this would be a contradiction since each vertex can cover at most $n-1$ edges. The runtime then follows from Proposition 63, where the required accuracy to satisfy (6.11a) and (6.11b) is a polynomial in problem parameters, following the proofs of Theorem 86 and Lemma 268 (see Appendix E.3.2 for more details). The requirements of Round are clear from inspection and the additive range of entropy. □

Theorem 39. *Let $G = (V, E)$ be bipartite and $\epsilon \in [\Omega(m^{-3}), 1)$. There is a randomized algorithm for the DDBM problem which maintains an ϵ-approximate matching with probability $1 - n^{-\Omega(1)}$, based on matrix scaling solver of [145], running in time $\widetilde{O}\left(n^2\epsilon^{-3}\right)$.*

Proof. It suffices to combine Lemma 269, Lemma 270, and Corollary 17. □

E.1.4 DDBM via Sinkhorn objective solver in [124]

In this section we give an alternative $m^{1+o(1)}\epsilon^{-2}$-time algorithm for decremental bipartite matching, leveraging a recent breakthrough of a $m^{1+o(1)}$-time algorithm for general graph flow problems in [124]. We follow the same notational conventions as in previous sections.

Our approach is to simply leverage the recent algorithm of [124] to solve the regularized subproblems (E.6) from the previous section. We recall the following result from [124].

Proposition 62 (Theorem 10.16 from [124])**.** *Given a graph $G = (V, E)$, demands $d \in \mathbb{R}^V$, costs $c \in \mathbb{R}^E$, and weights $w \in \mathbb{R}^E_{\ge 0}$, all entrywise bounded by $\exp(\log^{O(1)} m)$, let $\mathbf{B}$ be its unsigned adjacency matrix, and $h(f) = \sum_{e \in E} c_e f_e + w_e f_e \log f_e$. Then in $m^{1+o(1)}$ time we can find a flow f with $\mathbf{B}^\top f = d$, $f \ge 0$, and for any constant $C > 0$*

$$h(f) \le \min_{B^\top f^\star = d, f^\star \ge 0} h(f^\star) + m^{-C}.$$

Lemma 271 (Canonical solver for Sinkhorn objectives)**.** *For $\epsilon = \Omega(m^{-3})$, there is an $(\epsilon, \mathcal{T})$-canonical solver for the family of $f^{\mathsf{sink}}_{M,E}$ defined in Lemma 269, using Proposition 62 as* Solve *and truncation to E as* Round*, running in $m^{1+o(1)}$ time.*

Proof. To implement Solve, we apply Proposition 62 to the problem (E.6) and solve the subproblem to accuracy $O(\frac{\epsilon^3}{m})$, for a sufficiently small constant. We observe that (E.6) is a problem exactly of the form where Proposition 62 applies: by the multiplicative guarantee on ν^E and MCM(E) and since MCM(E) is 8-approximated by M, (6.11a) holds immediately. We additionally obtain (6.11b), as the function $f^{\mathsf{sink}}_{M,E}$ is $\Omega(\frac{\epsilon}{\log m})$-strongly convex in the coordinates of x. The requirements of Round are clear by inspection and by the additive range of entropy, as shown in Lemma 270. □

With this result, we follow Theorem 39 and obtain an improved runtime for DDBM.

Theorem 40. *Let $G = (V, E)$ be bipartite and $\epsilon \in [\Omega(m^{-3}), 1)$. There is a randomized algorithm for the DDBM problem which maintains an ϵ-approximate matching with probability $1 - n^{-\Omega(1)}$, based on the Sinkhorn objective solver of [124], running in time $m^{1+o(1)}\epsilon^{-2}$.*

Proof. It suffices to combine Lemma 269, Lemma 271, and Corollary 17. □

E.2 Proofs for Section 6.4

We first remark on the assumptions made throughout Section 6.4, restated as below: For some $\delta > 0$, we have

1. upper bounds on entries: $\|\mathbf{A}\|_\infty \le 1$, $\|b\|_\infty \le B_{\max}$, $\|c\|_\infty \le C_{\max}$. For simplicity, we assume $B_{\max} \ge C_{\max} \ge 1$, as otherwise we set $C_{\max} \leftarrow \max(1, C_{\max})$ and $B_{\max} \leftarrow \max(C_{\max}, B_{\max})$.
2. lower bounds on matrix column entries: $\max_i |\mathbf{A}_{ij}| \ge \delta$ for every $j \in [n]$.

The second assumption can be satisfied by padding entries of $\mathbf{A}$ by δ, which at most incurs $O(\delta) \ll O(\epsilon)$ error in the objective value for any $\delta \ll \epsilon$. Our runtimes extend to general $\mathbf{A}$ in a scale-invariant way (i.e. by scaling the whole problem by a factor of $\frac{1}{\|\mathbf{A}\|_\infty}$ and then scaling up the resulting error), so the first assumption (on $\|\mathbf{A}\|_\infty$) is for simplicity. All of our runtimes depend logarithmically on the quantity $B_{\max}$, which is polynomially bounded in all our applications.

Finally, regarding $C_{\max}$, it is clear we can assume without loss of generality that $c \le \mathbf{1}$ entrywise, since shifting c by a multiple of the all-ones vector changes the objective value in (6.16). In the unregularized case, i.e. $f_{\mu,\epsilon}$ with $\mu = \epsilon = 0$, [39] shows that we can also *lower bound* the vector c entrywise by -1. In the regularized case, we provide the following lemma which shows that by truncating the entries of c, we do not lose too much in objective value (see Appendix E.2).

Lemma 272. *Let $\tau \geq 0$, and define $S_\tau = \{i \in [m] \mid c_i \geq \min_{i\in[m]} c_i + \tau\}$ to be the large entries of c. Define $\mathcal{X}' := \{x \in \Delta^m \mid x_i = 0 \text{ for all } i \in S_\tau\}$. Then,*

$$\min_{x\in\mathcal{X}'} \max_{y\in[0,1]^n} f_{\mu,\epsilon}(x,y) \leq \min_{x\in\Delta^m} \max_{y\in[0,1]^n} f_{\mu,\epsilon}(x,y) + \mu m \exp\left(-\frac{\tau-3}{\mu}\right).$$

In other words, Lemma 272 shows that if our goal is to find an $x \in \Delta^m$ which approximately minimizes $\max_{y\in[0,1]^n} f_{\mu,\epsilon}(x,y)$, we may restrict ourselves to considering only x supported on coordinates where c is polylogarithmically bounded, without much loss in the error. In particular, setting $\tau = \Theta(\log \frac{m}{\sigma})$ above, where the final desired error is σ, yields this claim for $\mu \leq 1$. Our methods will have runtimes depending linearly on $C_{\max}$, which upon applying the preprocessing of Lemma 272, is an overall polylogarithmic dependence on the final target accuracy.

Now we prove Lemma 272. Before we do so, we state two simple helper lemmas used in its proof.

Lemma 273. *Let $\mathcal{X}, \mathcal{Y}$ be compact and convex, let $\mathcal{X}' \subseteq \mathcal{X}$ also be compact and convex, and let $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be convex-concave. Suppose for some $\Delta > 0$ it is the case that for all $y \in \mathcal{Y}$,*

$$\min_{x\in\mathcal{X}'} f(x,y) \leq \min_{x\in\mathcal{X}} f(x,y) + \Delta.$$

Then,

$$\min_{x\in\mathcal{X}'} \max_{y\in\mathcal{Y}} f(x,y) \leq \min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x,y) + \Delta.$$

Proof. Let $(x_\star, y_\star)$ be the optimizer to f over $\mathcal{X} \times \mathcal{Y}$, and let $(x'_\star, y'_\star)$ be the optimizer to f over $\mathcal{X}' \times \mathcal{Y}$. We conclude with the following sequence of inequalities:

$$f(x'_\star, y'_\star) \leq \min_{x\in\mathcal{X}} f(x, y'_\star) + \Delta \leq f(x_\star, y'_\star) + \Delta \leq f(x_\star, y_\star) + \Delta.$$

The three inequalities respectively used our assumption, $x_\star \in \mathcal{X}$, and $y_\star = \operatorname{argmax}_{y\in\mathcal{Y}} f(x_\star, y)$. □

Lemma 274. *For $\mu > 0$, define the function $\operatorname{smin}_\mu : \mathbb{R}^m \to \mathbb{R}$ by*

$$\operatorname{smin}_\mu(v) := -\mu \log\left(\sum_{i\in[m]} \exp\left(-\frac{1}{\mu} v_i\right)\right).$$

Fix v, and let $S \subseteq [m]$ be a set such that for all $i \in [m] \setminus S$, we have $v_i \geq \min_{i\in[m]} v_i + T$ for some $T > 0$. Further let $v' \in \mathbb{R}^S$ be the restriction of v to the set S. Then,

$$\operatorname{smin}_\mu(v') := -\mu \log\left(\sum_{i\in S} \exp\left(-\frac{1}{\mu} v'_i\right)\right) \leq \operatorname{smin}_\mu(v) + \mu m \exp\left(-\frac{T}{\mu}\right).$$

Proof. Let $v_{\min} := \min_{i\in[m]} v_i$. Note that

$$\sum_{i\in[m]\setminus S} \exp\left(-\frac{1}{\mu}v_i\right) \le m\exp\left(-\frac{1}{\mu}(v_{\min}+T)\right) \le m\exp\left(-\frac{T}{\mu}\right)\sum_{i\in S}\exp\left(-\frac{1}{\mu}v_i\right).$$

Here we used that some element in S achieves $v_i = v_{\min}$. Hence,

$$\sum_{i\in[m]} \exp\left(-\frac{1}{\mu}v_i\right) \le \left(1+m\exp\left(-\frac{T}{\mu}\right)\right)\sum_{i\in S}\exp\left(-\frac{1}{\mu}v_i\right).$$

Taking logarithms and scaling by $-\mu$ yields the claim, where we use $\log(1+c)\le c$ for $c\ge 0$. □

Combining these lemmas allow us to prove Lemma 272 formally.

Proof of Lemma 272. By Lemma 273, it suffices to first fix $y\in[0,1]^n$, and prove that

$$\min_{x\in\mathcal{X}'} f_{\mu,\epsilon}(x,y) \le \min_{x\in\Delta^m} f_{\mu,\epsilon}(x,y) + \mu m\exp\left(-\frac{\tau-3}{\mu}\right). \tag{E.7}$$

Clearly the term $b^\top y$ cancels from both sides. Next, define

$$v := \mathbf{A}y + c - \frac{\epsilon}{2}|\mathbf{A}|(y^2).$$

Let v' be the restriction of v to the indices in S_τ. By explicitly minimizing over $x\in\Delta^m$ and $x\in\mathcal{X}'$ respectively, (E.7) is equivalent to proving

$$\operatorname{smin}_\mu(v') \le \operatorname{smin}_\mu(v) + \mu m\exp\left(-\frac{\tau-3}{\mu}\right).$$

Since $\mathbf{A}y - \frac{\epsilon}{2}|\mathbf{A}|(y^2)$ has a range of at most 3, every truncated entry of v' must be at least $(\tau-3)$ larger than the smallest entry of v'. We conclude by applying Lemma 274 with $T=\tau-3$. □

E.2.1 Proofs for Section 6.4.1

Proposition 19 (Convergence of Algorithm 21)**.** *Given regularizer r with range at most Θ, suppose g is ν-strongly-monotone with respect to r (see (6.19)), and is α-relatively-Lipschitz with respect to r (see (6.20)). Let z_K be the output of Algorithm 21. Then, $V^r_{z_K}(z^\star) \le \left(\frac{\alpha}{\nu+\alpha}\right)^K \Theta + \frac{\varepsilon}{\nu}$.*

Proof. Fix an iteration k. We have

$$\begin{aligned}\nu V_{z_{k-1/2}}(z^\star) &\overset{(i)}{\le} \left\langle g(z_{k-1/2}) - g(z^\star), z_{k-1/2} - z^\star\right\rangle \overset{(ii)}{\le} \left\langle g(z_{k-1/2}), z_{k-1/2} - z^\star\right\rangle \\ &= \left\langle g(z_{k-1/2}), z_k - z^\star\right\rangle + \left\langle g(z_{k-1/2}), z_{k-1/2} - z_k\right\rangle,\end{aligned} \tag{E.8}$$

where we used (i) the definition of ν-strong monotonicity, and (ii) optimality of $z^\star$.

Next,

$$\begin{aligned}\left\langle g(z_{k-1/2}), z_k - z^\star\right\rangle - \frac{\varepsilon}{2} &\overset{(i)}{\leq} -\left\langle \alpha\nabla V_{z_{k-1}}(z_k) + \nu\nabla V_{z_{k-1/2}}(z_k), z_k - z^\star\right\rangle \\ &\overset{(ii)}{\leq} \nu\Big(V_{z_{k-1/2}}(z^\star) - V_{z_k}(z^\star) - V_{z_{k-1/2}}(z_k)\Big) \qquad \text{(E.9)} \\ &\quad + \alpha\Big(V_{z_{k-1}}(z^\star) - V_{z_k}(z^\star) - V_{z_{k-1}}(z_k)\Big) \\ &\leq \alpha V_{z_{k-1}}(z^\star) - (\nu+\alpha)V_{z_k}(z^\star) + \nu V_{z_{k-1/2}}(z^\star) - \alpha V_{z_{k-1}}(z_k), \qquad \text{(E.10)}\end{aligned}$$

and similarly,

$$\begin{aligned}&\left\langle g(z_{k-1/2}), z_{k-1/2} - z_k\right\rangle - \frac{\varepsilon}{2} \\ &\overset{(i)}{\leq} -\langle \alpha\nabla V_{z_{k-1}}(z_{k-1/2}), z_{k-1/2} - z_k\rangle + \left\langle g(z_{k-1/2}) - g(z_{k-1})), z_{k-1/2} - z_k\right\rangle \\ &\overset{(ii)}{\leq} \alpha\Big(V_{z_{k-1}}(z_k) - V_{z_{k-1/2}}(z_k) - V_{z_{k-1}}(z_{k-1/2})\Big) + \left\langle g(z_{k-1/2}) - g(z_{k-1})), z_{k-1/2} - z_k\right\rangle \\ &\overset{(iii)}{\leq} \alpha V_{z_{k-1}}(z_k), \qquad \text{(E.11)}\end{aligned}$$

where we used (i) the approximate proximal oracle optimality conditions for z_k and $z_{k-1/2}$, (ii) the three-point property of Bregman divergence (6.7), and (iii) the relative Lipschitzness conddition of g. In the last inequality, we also use nonnegativity of divergence term V.

Now, combining (E.10) and (E.11) with (E.8) yields

$$\nu V_{z_{k-1/2}}(z^\star) \leq \alpha V_{z_{k-1}}(z^\star) - (\nu+\alpha)V_{z_k}(z^\star) + \nu V_{z_{k-1/2}}(z^\star) + \varepsilon.$$

Rearranging terms then yields

$$V_{z_k}(z^\star) \leq \frac{\alpha}{\nu+\alpha} V_{z_{k-1}}(z^\star) + \frac{\varepsilon}{\nu+\alpha}.$$

Applying this bound recursively K times and using that $V_{z_0}(u) \leq r(u) - r(z_0) \leq \Theta$ for z_0 the minimizer of r, we have

$$V_{z_K}(z^\star) \leq \left(\frac{\alpha}{\nu+\alpha}\right)^K \Theta + \sum_{k=0}^{K-1}\left(\frac{\alpha}{\nu+\alpha}\right)^k\left(\frac{\varepsilon}{\nu+\alpha}\right) \leq \left(\frac{\alpha}{\nu+\alpha}\right)^K \Theta + \frac{\varepsilon}{\nu}.$$

□

Corollary 18 (Convergence of Algorithm 22)**.** *Let $\delta, \varepsilon \in (0,1)$, $\rho \geq 1$. Suppose we are given*

$\gamma \in \mathcal{Z}_* = \mathcal{X}_* \times \mathcal{Y}_*$ with $\max(\|\gamma^{\mathsf{x}}\|_\infty, \|\gamma^{\mathsf{y}}\|_1) \le B$, and define the proximal subproblem solution

$$x_{\mathsf{OPT}}, y_{\mathsf{OPT}} = \operatorname{argmin}_{x \in \Delta^m} \operatorname{argmin}_{y \in [0,1]^n} f(x,y) := \langle \gamma^x, x \rangle + \langle \gamma^y, y \rangle + \theta r(x,y) \quad \text{for some } \theta > 0.$$

If the Hessian condition in Lemma 81 holds with a constant $\kappa > 0$, and all simplex iterates x of Algorithm 22 satisfy $x \ge \delta$ elementwise, then the algorithm finds a $\frac{\varepsilon}{2}$-approximate solution to the proximal oracle within $T = O\left(\log\left(\frac{\rho(B+mn\theta)^2}{\delta\varepsilon\theta}\right)\right)$ iterations.

To prove Corollary 18, we first provide technical lemma that shows how to transfer the accuracy desired for transferring from function error in Algorithm 22 guarantees to the duality gap error desired by an approximate extragradient step in Algorithm 21, in a controllable sense.

Lemma 275. *Let $\delta, \varepsilon \in (0,1)$, $\rho \ge 1$. Suppose we are given $\gamma \in \mathcal{Z}_* = \mathcal{X}_* \times \mathcal{Y}_*$ satisfying $\max(\|\gamma^{\mathsf{x}}\|_\infty, \|\gamma^{\mathsf{y}}\|_1) \le B$, and define the proximal oracle subproblem solution*

$$x_{\mathsf{OPT}}, y_{\mathsf{OPT}} = \operatorname{argmin}_{x \in \Delta^m} \operatorname{argmin}_{y \in [0,1]^n} f(x,y) := \langle \gamma^x, x \rangle + \langle \gamma^y, y \rangle + \theta r(x,y) \quad \text{for some } \theta > 0.$$

Let $z = (x,y)$ be an approximate solution satisfying $f(x,y) \le f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}}) + \frac{\varepsilon^2\delta^2\theta}{64\rho(8n\theta+B)^2}$, and suppose $x, x_{\mathsf{OPT}} \ge \delta$ holds elementwise. For any $w \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$,

$$\langle \gamma + \theta \nabla r(z), z - w \rangle \le \frac{\varepsilon}{2}.$$

Proof. Let $z_{\mathsf{OPT}} = (x_{\mathsf{OPT}}, y_{\mathsf{OPT}})$. Under the assumption that $f(z) - f(z_{\mathsf{OPT}}) \le A := \frac{\varepsilon^2\delta^2\theta}{64\rho(8n\theta+B)^2}$, using the lower bound in (6.23) with $|\mathbf{A}|^\top x \ge \delta^2$ entrywise by the assumption $\max_i |\mathbf{A}_{ij}| \ge \delta$ for every column j, and since entropy is strongly convex in the ℓ_1 norm, we have

$$\begin{aligned} &\frac{\theta\rho}{4}\|x - x_{\mathsf{OPT}}\|_1^2 + \frac{\delta^2\theta}{2\rho}\|y - y_{\mathsf{OPT}}\|_\infty^2 \le f(z) - f(z_{\mathsf{OPT}}) \le A \\ \implies \quad &\|x - x_{\mathsf{OPT}}\|_1 \le \frac{\varepsilon}{4\left(B + \frac{8n\rho\theta}{\delta}\right)} \text{ and } \|y - y_{\mathsf{OPT}}\|_\infty \le \frac{\varepsilon}{4\left(B + \frac{8n\theta}{\rho}\right)}. \end{aligned} \tag{E.12}$$

For $w = (u,v)$, by definition of z_{OPT}, $\langle \gamma + \theta\nabla r(z_{\mathsf{OPT}}), z_{\mathsf{OPT}} - w \rangle \le 0$. Hence, we bound

$$\begin{aligned} \langle \gamma + \theta\nabla r(z), z - w \rangle &\le \langle \gamma + \theta\nabla r(z), z - w \rangle - \langle \gamma + \theta\nabla r(z_{\mathsf{OPT}}), z_{\mathsf{OPT}} - w \rangle \\ &= \langle \gamma, z - z_{\mathsf{OPT}} \rangle + \theta \langle \nabla r(z) - \nabla r(z_{\mathsf{OPT}}), z_{\mathsf{OPT}} - w \rangle + \theta \langle \nabla r(z), z - z_{\mathsf{OPT}} \rangle . \end{aligned} \tag{E.13}$$

We now bound the terms on the right-hand side of (E.13):

$$\begin{aligned}
\langle \gamma, z - z_{\mathsf{OPT}} \rangle &\leq \|\gamma^{\mathsf{x}}\|_\infty \|x - x_{\mathsf{OPT}}\|_1 + \|\gamma^{\mathsf{y}}\|_1 \|y - y_{\mathsf{OPT}}\|_\infty \\
&\leq B \|x - x_{\mathsf{OPT}}\|_1 + B \|y - y_{\mathsf{OPT}}\|_\infty , \\
\langle \nabla_x r(z) - \nabla_x r(z_{\mathsf{OPT}}), x_{\mathsf{OPT}} - u \rangle &\leq \|x_{\mathsf{OPT}} - u\|_1 \|\nabla_x r(z) - \nabla_x r(z_{\mathsf{OPT}})\|_\infty \\
&\leq 2\rho \left\| \log \frac{x}{x_{\mathsf{OPT}}} \right\|_\infty + \frac{2}{\rho} \| |\mathbf{A}|((y + y_{\mathsf{OPT}}) \circ (y - y_{\mathsf{OPT}})) \|_\infty \\
&\leq \frac{2\rho}{\delta} \|x - x_{\mathsf{OPT}}\|_1 + \frac{4}{\rho} \|y - y_{\mathsf{OPT}}\|_\infty , \\
\langle \nabla_y r(z) - \nabla_y r(z_{\mathsf{OPT}}), y_{\mathsf{OPT}} - v \rangle &\leq \|y_{\mathsf{OPT}} - v\|_\infty \|\nabla_y r(z) - \nabla_y r(z_{\mathsf{OPT}})\|_1 \\
&\leq \frac{2n}{\rho} \|y - y_{\mathsf{OPT}}\|_\infty + \frac{2n}{\rho} \|x - x_{\mathsf{OPT}}\|_1 , \\
\langle \nabla r(z), z - z_{\mathsf{OPT}} \rangle &\leq \|\nabla_x r(z)\|_\infty \|x - x_{\mathsf{OPT}}\|_1 + \|\nabla_y r(z)\|_1 \|y - y_{\mathsf{OPT}}\|_\infty \\
&\leq \left(\rho \left(1 + \log \left(\frac{1}{\delta} \right) \right) + \frac{1}{\rho} \right) \|x - x_{\mathsf{OPT}}\|_1 + \frac{2}{\rho} \|y - y_{\mathsf{OPT}}\|_\infty .
\end{aligned} \tag{E.14}$$

For the second bound, we used $\log(1+c) \leq c$ for whichever of $c = \frac{x_j}{[x_{\mathsf{OPT}}]_j}$ or $c = \frac{[x_{\mathsf{OPT}}]_j}{x_j}$ is larger, and the entrywise lower bounds on x and x_{OPT}. For the third bound, we used

$$\begin{aligned}
\left\| \mathbf{diag}\left(y_{\mathsf{OPT}}\right) |\mathbf{A}|^\top x_{\mathsf{OPT}} - \mathbf{diag}\left(y\right) |\mathbf{A}|^\top x \right\|_1 &\leq \sum_{i \in [m], j \in [n]} |\mathbf{A}|_{ij} \left| [x_{\mathsf{OPT}}]_i [y_{\mathsf{OPT}}]_j - x_i y_j \right| \\
&\leq \sum_{i \in [m], j \in [n]} |\mathbf{A}|_{ij} \left(\left| [y_{\mathsf{OPT}}]_j - y_j \right| + |[x_{\mathsf{OPT}}]_i - x_i| \right) \\
&\leq n(\|y_{\mathsf{OPT}} - y\|_\infty + \|x_{\mathsf{OPT}} - x\|_1).
\end{aligned}$$

Plugging the bounds of (E.14) back in (E.13), we can thus conclude that for $\rho \geq 1$, $\delta \leq 1$

$$\langle \gamma + \theta \nabla r(z), z - w \rangle \leq \left(B + \frac{8n\rho\theta}{\delta} \right) \|x - x_{\mathsf{OPT}}\|_1 + \left(B + \frac{8n\theta}{\rho} \right) \|y - y_{\mathsf{OPT}}\|_\infty \leq \frac{\varepsilon}{2},$$

where for the last inequality we use conditions in (E.12). □

We use this lemma to prove Corollary 18 formally below.

Proof of Corollary 18. We first bound $f(x^{(1)}, y^{(0)}) - f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}}) \leq \max_{z \in \mathcal{Z}} f(z) - f(z_{\mathsf{OPT}}) \leq 2\|\gamma^{\mathsf{x}}\|_\infty + 2\|\gamma^{\mathsf{y}}\|_1 + \theta \left(\rho \log m + \frac{2}{\rho} \right) \leq 4B + \theta \left(\rho \log m + \frac{2}{\rho} \right)$ where the last inequality uses ℓ_1-ℓ_∞ Hölder, the domain definitions $\mathcal{X} = \Delta^m$, $\mathcal{Y} = [0,1]^n$, and the definition of $r = r_{\mu,\epsilon}$ as in (6.18).

Now applying Lemma 81, after $O\left(\log\left(\frac{\rho(B+mn\theta)^2}{\delta\varepsilon\theta}\right)\right)$ iterations for a sufficiently large constant,

$$f(x^{(T+1)}, y^{(T)}) - f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}}) \le \left(1 - \frac{1}{2\kappa}\right)^T \left(f(x^{(1)}, y^{(0)}) - f(x_{\mathsf{OPT}}, y_{\mathsf{OPT}})\right)$$
$$\le \left(1 - \frac{1}{2\kappa}\right)^T \left(4B + \theta\left(\rho \log m + \frac{2}{\rho}\right)\right) \le \frac{\varepsilon^2\delta^2\theta}{64\rho(8n\theta + B)^2},$$

which by Lemma 275 in turn implies it implements an approximate proximal oracle to $\frac{\varepsilon}{2}$ accuracy. □

E.2.2 Proofs for Section 6.4.2

We give a helpful variant of the Cauchy-Schwarz inequality in appropriate "local norms" and a consequence about approximating the Hessian of our regularizer. These find uses in proving Lemma 86.

Lemma 82 (Bounds on regularizer)**.** *Suppose* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *has* $\|\mathbf{A}\|_\infty \le 1$*. For any* $z = (x, y) \in \Delta^m \times [0,1]^n$*,* $r = r_{\mu,\epsilon}$ *defined as in* (6.18)*, and* $\bar{x} \in \mathbb{R}^m_{>0}$*,* $\langle x, \mathbf{A}y\rangle \le \|x\|_{\mathbf{diag}\left(\frac{1}{\bar{x}}\right)} \|y\|_{\mathbf{diag}\left(|\mathbf{A}|^\top \bar{x}\right)}$*. Further, if* $\rho \ge 3$*, the matrix*

$$\mathbf{D}(x) := \begin{pmatrix} \frac{\rho}{2}\mathbf{diag}\left(\frac{1}{x}\right) & \mathbf{0} \\ \mathbf{0} & \frac{1}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x\right) \end{pmatrix} \tag{6.22}$$

satisfies the following relationship with the Hessian matrix of $r(z)$*:*

$$\mathbf{D}(x) \preceq \nabla^2 r(z) = \begin{pmatrix} \rho \cdot \mathbf{diag}\left(\frac{1}{x}\right) & \frac{2}{\rho}\mathbf{A}\mathbf{diag}\left(y\right) \\ \frac{2}{\rho}\mathbf{diag}\left(y\right)\mathbf{A}^\top & \frac{2}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x\right) \end{pmatrix} \preceq 4\mathbf{D}(x). \tag{6.23}$$

Proof. For the first property, it suffices to square both sides and use Cauchy-Schwarz:

$$\left(\sum_{i\in[m]}\sum_{j\in[n]} |\mathbf{A}_{ij}u_i v_j|\right)^2 \le \left(\sum_{i\in[m]}\sum_{j\in[n]} \frac{|\mathbf{A}_{ij}|}{[\bar{x}]_i} u_i^2\right)\left(\sum_{i\in[m]}\sum_{j\in[n]} |\mathbf{A}_{ij}|\,[\bar{x}]_i v_j^2\right)$$
$$\le \left(\sum_{i\in[m]} \frac{\|\mathbf{A}\|_\infty}{[\bar{x}]_i} u_i^2\right)\left(\sum_{j\in[n]} \left[|\mathbf{A}|^\top \bar{x}\right]_j v_j^2\right)$$
$$\le \|u\|^2_{\mathbf{diag}\left(\frac{1}{\bar{x}}\right)} \|v\|^2_{\mathbf{diag}\left(|\mathbf{A}|^\top \bar{x}\right)}.$$

Next, given any $w = (u, v)$, we have

$$\begin{aligned} w^\top \nabla^2 r(x,y) w &= \rho \|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{2}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} + \frac{4}{\rho} u^\top \mathbf{Adiag}(y)\, v \\ &\le \rho \|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{2}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} + \frac{4}{\rho}\|u\|_{\mathbf{diag}(\frac{1}{x})} \|\mathbf{diag}(y)\, v\|_{\mathbf{diag}(|\mathbf{A}|^\top x)} \\ &\le \left(\rho + \frac{2}{\rho}\right)\|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{4}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} \\ &\le 4\left(\frac{\rho}{2}\|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{1}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)}\right) = 4 w^\top \mathbf{D}(x) w. \end{aligned}$$

Similarly, on the other side we have

$$\begin{aligned} w^\top \nabla^2 r(x,y) w &= \rho \|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{2}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} + \frac{4}{\rho} u^\top \mathbf{Adiag}(y)\, v \\ &\ge \rho \|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{2}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} - \frac{4}{\rho}\|u\|_{\mathbf{diag}(\frac{1}{x})} \|\mathbf{diag}(y)\, v\|_{\mathbf{diag}(|\mathbf{A}|^\top x)} \\ &\ge \left(\rho - \frac{4}{\rho}\right)\|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \left(\frac{2}{\rho} - \frac{1}{\rho}\right)\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} \\ &\ge \frac{\rho}{2}\|u\|^2_{\mathbf{diag}(\frac{1}{x})} + \frac{1}{\rho}\|v\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x)} = w^\top \mathbf{D}(x) w. \end{aligned}$$

□

Lemma 83 (Error of padding, cf. Lemma 6, [112])**.** *For $\delta > 0$ and $\bar{z} = (\bar{x}, y) \in \Delta^m \times [0,1]^n$ let $z = (x, y) \in \Delta^m \times [0,1]^n$ where $x = \mathcal{O}_\delta(\bar{x})$ (Definition 26), then for r in (6.18), and any $w \in \mathcal{Z} = \Delta^m \times [0,1]^n$, $V_z^r(w) - V_{\bar{z}}^r(w) \le \left(\rho + \frac{8}{\rho}\right) m\delta$.*

Proof. We write $r(x,y) = \rho H(x) + \frac{1}{\rho} Q(x,y)$ where

$$H(x) = \sum_{i \in [m]} x_i \log x_i,\ Q(x,y) = \left(y^2\right)^\top |\mathbf{A}|^\top x,$$

and bound the Bregman divergences induced by H and Q respectively.

First note that by definition of the padding oracle we have $\|x_k - \bar{x}_k\|_1 \le \|x_k - \hat{x}_k\|_1 + \|\bar{x}_k - \hat{x}_k\|_1 \le 2m\delta$, and that x_k is a $m\delta$-padding of $\bar{x}_k$ by the definition of padding in [112] (cf. Definition 2). Thus using Lemma 6 of [112], we have

$$V_z^H(w) - V_{\bar{z}}^H(w) \le m\delta.$$

For the quadratic part, letting $w = (u, v)$, we have

$$\begin{aligned} V_z^Q(w) - V_{\bar{z}}^Q(w) &= Q(\bar{z}) - Q(z) + \langle \nabla Q(\bar{z}), w - \bar{z}\rangle - \langle \nabla Q(z), w - z\rangle \\ &\le \|x - \bar{x}\|_1 + \left\langle |\mathbf{A}| y^2, x - \bar{x}\right\rangle + \left\langle \mathbf{diag}(y)\, |\mathbf{A}|^\top (\bar{x} - x), v - y\right\rangle \\ &\le 4\|x - \bar{x}\|_1 \le 8m\delta, \end{aligned}$$

where the first inequality used $\bar{y} = y$, and both the first and second used the assumed bounds $\|y\|_\infty \le 1$, $\|\mathbf{A}\|_\infty \le 1$. Combining these bounds proves the statement. □

Lemma 84 (Iterate stability in Algorithm 22)**.** *Suppose $\epsilon \le 1$, $\rho \ge 6$, and $\alpha \ge \frac{36}{\rho}(\mu \log \frac{4}{\delta} + 3C_{\max})$. Let (x_k, y_k) denote blocks of z_k, the k^{th} iterate of Algorithm 21. In any iteration k of Algorithm 21, calling Algorithm 22 to implement Line 6, if $x_{k-1} \ge \frac{\delta}{2}$ entrywise, $x^{(t+1)} \in x_{k-1} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]$, for all $t \in [T]$. Calling Algorithm 22 to implement Line 7, if $x_{k-1/2} \ge \frac{\delta}{4}$ entrywise, $x^{(t+1)} \in x_{k-1}^{\frac{\alpha}{\alpha+\nu}} \circ x_{k-1/2}^{\frac{\nu}{\alpha+\nu}} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]$ for all $t \in [T]$.*

Proof. We first handle the case of Line 6 in Algorithm 21. Using Algorithm 22 to implement this step, we observe that the iterates satisfy

$$\begin{aligned} x^{(t+1)} &\leftarrow \text{argmin}_{x \in \Delta^m} \langle \gamma^{\mathsf{x}}, x \rangle + \theta r(x, y^{(t)}) \ \text{ where } \ \theta = \alpha, \\ \text{and } \ \gamma^{\mathsf{x}} &= g^{\mathsf{x}}(x_{k-1}, y_{k-1}) - \alpha \nabla_x r(z_{k-1}) \\ &= \mathbf{A} y_{k-1} + c + \mu(\mathbf{1} + \log(x_{k-1})) - \frac{\epsilon}{2} |\mathbf{A}| \left(y_{k-1}^2\right) - \alpha\rho(1 + \log x_{k-1}) - \frac{\alpha}{\rho}|\mathbf{A}| y_{k-1}^2, \end{aligned}$$

which implies

$$x^{(t+1)} \propto x_{k-1} \circ \exp\left(\frac{1}{\alpha\rho}\left(-\frac{\alpha}{\rho}|\mathbf{A}|\left(y^{(t)}\right)^2 - \mathbf{A} y_{k-1} - c + \frac{\epsilon}{2}|\mathbf{A}| y_{k-1}^2 + \frac{\alpha}{\rho}|\mathbf{A}| y_{k-1}^2 - \mu \log(x_{k-1})\right)\right).$$

Consequently, under the given assumptions on x_{k-1}, ρ, and α, and using $\|c\|_\infty \le C_{\max}$,

$$\begin{aligned} \left|\log\left(\frac{x^{(t+1)}}{x_{k-1}}\right)\right| \le \frac{1}{\alpha\rho}\left(1 + C_{\max} + \frac{\alpha}{\rho} + \frac{\epsilon}{2} + \mu \log\frac{2}{\delta}\right) \le \frac{1}{18} \\ \implies x^{(t+1)} \in x_{k-1} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]. \end{aligned}$$

Next, we handle the case of Line 7 in Algorithm 22. Here, the iterates of Algorithm 22 satisfy

$$\begin{aligned} x^{(t+1)} &\leftarrow \text{argmin}_{x \in \Delta^m} \langle \gamma^{\mathsf{x}}, x \rangle + \theta r(x, y^{(t)}) \ \text{ where } \ \theta = \alpha + \nu, \\ \text{and } \ \gamma^{\mathsf{x}} &= g^{\mathsf{x}}(x_{k-1/2}, y_{k-1/2}) - \alpha \nabla_x r(z_{k-1}) - \nu \nabla_x r(z_{k-1/2}) \\ &= \mathbf{A} y_{k-1/2} + c + \mu(\mathbf{1} + \log(x_{k-1/2})) - \frac{\epsilon}{2} |\mathbf{A}| \left(y_{k-1/2}^2\right) \\ &\quad - \alpha\rho(1 + \log x_{k-1}) - \frac{\alpha}{\rho}|\mathbf{A}| y_{k-1}^2 - \nu\rho(1 + \log x_{k-1/2}) - \frac{\nu}{\rho}|\mathbf{A}| y_{k-1/2}^2. \end{aligned}$$

Hence,

$$\begin{aligned} x^{(t+1)} \propto \quad & x_{k-1}^{\frac{\alpha}{\alpha+\nu}} \circ x_{k-1/2}^{\frac{\nu}{\alpha+\nu}} \circ \tau^{\mathsf{x}} \\ \text{where } \tau^{\mathsf{x}} = \quad & \exp\Bigg(\frac{1}{(\alpha+\nu)\rho}\Bigg(-\frac{\alpha}{\rho}|\mathbf{A}|\left(y^{(t)}\right)^2 - \frac{\nu}{\rho}|\mathbf{A}|\left(y^{(t)}\right)^2 - \mathbf{A}y_{k-1/2} - c \\ & + \frac{\epsilon}{2}|\mathbf{A}|y_{k-1/2}^2 + \frac{\alpha}{\rho}|\mathbf{A}|y_{k-1}^2 + \frac{\nu}{\rho}|\mathbf{A}|y_{k-1/2}^2 - \mu\log(x_{k-1/2})\Bigg)\Bigg). \end{aligned}$$

Consequently, under a similar calculation as before,

$$\begin{aligned} \exp\left(-\frac{1}{18}\right) \leq \tau^{\mathsf{x}} \leq \exp\left(\frac{1}{18}\right) \text{ entrywise} \\ \implies x^{(k+1)} \in x_{k-1}^{\frac{\alpha}{\alpha+\nu}} \circ x_{k-1/2}^{\frac{\nu}{\alpha+\nu}} \cdot \left[\exp\left(-\frac{1}{9}\right), \exp\left(\frac{1}{9}\right)\right]. \end{aligned}$$

□

E.2.3 Proofs for Section 6.4.3

Lemma 85 (Strong monotonicity)**.** *Let $\mu \geq \frac{\epsilon}{2}$ and $\rho := \sqrt{\frac{2\mu}{\epsilon}}$. The gradient operator $g_{\mu,\epsilon}$* (6.17) *is $\nu := \frac{1}{2}\sqrt{\frac{\mu\epsilon}{2}}$-strongly monotone (see* (6.19)*) with respect to $r_{\mu,\epsilon}$ defined in* (6.18)*.*

Proof. Throughout the proof, let $g := g_{\mu,\epsilon}$, $f := f_{\mu,\epsilon}$, and $r := r_{\mu,\epsilon}$ for notational simplicity. In order to show the desired bound

$$\langle g(w) - g(z), w - z\rangle \geq \frac{1}{3}\sqrt{\frac{\mu\epsilon}{2}}\langle \nabla r(w) - \nabla r(z), w - z\rangle, \tag{E.15}$$

we begin by putting (E.15) into a more convenient form. Letting $\mathbf{J}(z)$ be the Jacobian of g at the point z, we have by direct integration (where $z_t := (1-t)z + tw$ for all $t \in [0,1]$)

$$\langle g(w) - g(z), w - z\rangle = \int_0^1 (w-z)^\top \mathbf{J}(z_t)(w-z)dt.$$

Since quadratic forms through a square matrix $\mathbf{M}$ are preserved by replacing $\mathbf{M}$ with its symmetric part $\frac{1}{2}(\mathbf{M} + \mathbf{M}^\top)$, it suffices to understand the symmetric part of $\mathbf{J}(z_t)$. The restriction of $\mathbf{J}_\mu$ to the xy and yx blocks is skew-symmetric, since the blocks are respectively $\nabla^2_{xy}f(x,y)$ and $-\nabla^2_{yx}f(x,y)$.

Similarly, its restrictions to its xx and yy blocks are symmetric. Hence, we have

$$\begin{aligned}\langle g(w) - g(z), w - z\rangle &= \int_0^1 (w-z)^\top \begin{pmatrix} \nabla_{xx}^2 f(z_t) & \mathbf{0} \\ \mathbf{0} & -\nabla_{yy}^2 f(z_t) \end{pmatrix} (w-z)dt \\ &= \int_0^1 (w-z)^\top \begin{pmatrix} \mu \cdot \mathbf{diag}\left(\frac{1}{x_t}\right) & \mathbf{0} \\ \mathbf{0} & \epsilon \cdot \mathbf{diag}\left(|\mathbf{A}|^\top x_t\right) \end{pmatrix} (w-z)dt.\end{aligned} \tag{E.16}$$

In the last line, we denoted $z_t := (x_t, y_t)$ for all $t \in [0,1]$. Next, to bound the right hand side of (E.15), integrating once more yields

$$\begin{aligned}\langle \nabla r(w) - \nabla r(z), w - z\rangle &= \int_0^1 (w-z)^\top \nabla^2 r(z_t)(w-z)dt \\ &= \int_0^1 (w-z)^\top \begin{pmatrix} \rho \cdot \mathbf{diag}\left(\frac{1}{x_t}\right) & \frac{2}{\rho}|\mathbf{A}|\,\mathbf{diag}\,(y_t) \\ \frac{2}{\rho}\mathbf{diag}\,(y_t)\,|\mathbf{A}|^\top & \frac{2}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x_t\right) \end{pmatrix} (w-z)dt \\ &\le \int_0^1 (w-z)^\top \begin{pmatrix} 2\rho \cdot \mathbf{diag}\left(\frac{1}{x_t}\right) & \mathbf{0} \\ \mathbf{0} & \frac{4}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x_t\right) \end{pmatrix} (w-z)dt.\end{aligned} \tag{E.17}$$

The last line used Lemma 82 for all $(x_t, y_t) \in \Delta^m \times [0,1]^n$. Combining the bounds (E.16) and (E.17) yields the conclusion. □

Corollary 19 (Iterate stability in Algorithm 23)**.** *Assume the same parameter bounds as Lemma 84, and that $\delta \in (0, m^{-1})$. In the k^{th} outer loop of Algorithm 23, $x_{k-1} \ge \frac{\delta}{2}$ entrywise. Further, for all iterates $x^{(t+1)}$ computed in Line 7 to Line 8 and x_{OPT} as defined in* (6.21) *with $\theta = \alpha$, $\frac{1}{2}x_{k-1} \le x^{(t+1)}, x_{\mathsf{OPT}} \le 2x_{k-1}$, and $x^{(t+1)}, x_{\mathsf{OPT}} \ge \frac{\delta}{4}$, entrywise. Similarly, for all iterates $x^{(t+1)}$ computed in Line 9 to Line 10 and x_{OPT} as defined in* (6.21) *with $\theta = \alpha+\nu$, $\frac{1}{2}x_{k-1/2} \le x^{(t+1)}, x_{\mathsf{OPT}} \le 2x_{k-1/2}$ and $x^{(t+1)}, x_{\mathsf{OPT}} \ge \frac{\delta}{4}$, entrywise.*

Proof. It suffices to prove $x_{k-1} \ge \frac{\delta}{2}$ entrywise as all other conclusions are then immediate consequences of Lemma 84 (the conclusions about x_{OPT} follow from taking a limit). When $k = 1$, $x_{k-1} \ge \frac{\delta}{2}$ holds by the assumption that $\delta \le \frac{1}{2m}$. When $k > 1$, x_{k-1} is the result of padding with parameter δ. We note for any $x \in \Delta^m$ we have the desired

$$1 \le \|\max(x,\delta)\|_1 \le 1 + \delta m \implies \mathcal{O}_\delta(x) \ge \frac{\delta}{\|\max(x,\delta)\|_1} \ge \frac{\delta}{1+\delta m} \ge \frac{\delta}{2}.$$

□

Lemma 86 (Relative Lipschitzness)**.** *Assume the same parameter bounds as in Lemma 84. In the k^{th} outer loop of Algorithm 23, let $\bar{z}_k \leftarrow (x^{(T+1)}, y^{(T)})$ from Line 10 be z_k before the padding*

operation. Then, $x_{k-1/2}, \bar{x}_k \in [\frac{1}{2}x_{k-1}, 2x_{k-1}]$ *elementwise and*

$$\left\langle g(z_{k-1/2}) - g(z_{k-1}), z_{k-1/2} - \bar{z}_k \right\rangle \le \alpha \left(V_{z_{k-1}}(z_{k-1/2}) + V_{z_{k-1/2}}(\bar{z}_k) \right) \quad \textit{for} \quad \alpha = 4 + 32\sqrt{\frac{\mu\epsilon}{2}}.$$

Proof. The conclusion that $x_{k-1/2}, \bar{x}_k \in [\frac{1}{2}x_{k-1}, 2x_{k-1}]$ elementwise follows from Corollary 19. Next, we have for $z_\beta = z_{k-1} + \beta(z_{k-1/2} - z_{k-1})$, and $\rho \ge 6$,

$$\begin{aligned} V_{z_{k-1}}(z_{k-1/2}) &= \int_0^1 (1-\beta) \|z_{k-1/2} - z_{k-1}\|^2_{\nabla^2 r(z_{k-1} + \beta(z_{k-1/2} - z_{k-1}))} d\beta \\ &\ge \frac{1}{4} \left\| z_{k-1/2} - z_{k-1} \right\|^2_{D(x_{k-1})}, \end{aligned}$$

following the fact that $\nabla^2 r(z_{k-1} + \beta(z_{k-1/2} - z_{k-1})) \succeq \mathbf{D}(x_{k-1} + \beta(x_{k-1/2} - x_{k-1})) \succeq \frac{1}{2}\mathbf{D}(x_{k-1})$ from Lemma 82 and $x_{k-1/2} \in [\frac{1}{2}x_{k-1}, 2x_{k-1}]$. Similarly, we also have

$$V_{z_{k-1/2}}(\bar{z}_k) \ge \frac{1}{4} \left\| z_{k-1/2} - \bar{z}_k \right\|^2_{\mathbf{D}(x_{k-1})}.$$

On the other hand, we directly compute

$$\begin{aligned} &\left\langle g(z_{k-1/2}) - g(z_{k-1}), z_{k-1/2} - \bar{z}_k \right\rangle \\ &\quad = \left\langle \mathbf{A}(y_{k-1/2} - y_{k-1}), x_{k-1/2} - \bar{x}_k \right\rangle + \left\langle -\mathbf{A}^\top (x_{k-1/2} - x_{k-1}), y_{k-1/2} - \bar{y}_k \right\rangle \\ &\qquad + \left\langle \mu \log\left(\frac{x_{k-1/2}}{x_{k-1}} \right) - \frac{\epsilon}{2} |\mathbf{A}| (y^2_{k-1/2} - y^2_{k-1}), x_{k-1/2} - \bar{x}_k \right\rangle \\ &\qquad + \left\langle \epsilon \cdot \mathbf{diag}\left(y_{k-1/2} \right) |\mathbf{A}|^\top x_{k-1/2} - \epsilon \cdot \mathbf{diag}\left(y_{k-1} \right) |\mathbf{A}|^\top x_{k-1}, y_{k-1/2} - \bar{y}_k \right\rangle. \end{aligned} \tag{E.18}$$

We now bound the terms on the right-hand side of (E.18). By Lemma 82,

$$\begin{aligned} &\left\langle \mathbf{A}(y_{k-1/2} - y_{k-1}), x_{k-1/2} - \bar{x}_k \right\rangle \\ &\qquad \le \frac{1}{2\rho} \|y_{k-1/2} - y_{k-1}\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x_{k-1})} + \frac{\rho}{2} \|x_{k-1/2} - \bar{x}_k\|^2_{\mathbf{diag}\left(\frac{1}{x_{k-1}}\right)}, \\ &\left\langle \mathbf{A}^\top (x_{k-1/2} - x_{k-1}), \bar{y}_k - y_{k-1/2} \right\rangle \\ &\qquad \le \frac{1}{2\rho} \|y_{k-1/2} - \bar{y}_k\|^2_{\mathbf{diag}(|\mathbf{A}|^\top x_{k-1})} + \frac{\rho}{2} \|x_{k-1/2} - x_{k-1}\|^2_{\mathbf{diag}\left(\frac{1}{x_{k-1}}\right)}. \end{aligned} \tag{E.19}$$

For the rest of the terms we have, letting $(x_\beta, y_\beta) = z_\beta = (1-\beta)z_{k-1} + \beta z_{k-1/2}$ for all $\beta \in [0,1]$,

$$\begin{aligned}
&\left\langle \mu \log\left(\frac{x_{k-1/2}}{x_{k-1}}\right) - \frac{\epsilon}{2}|\mathbf{A}|(y_{k-1/2}^2 - y_{k-1}^2), x_{k-1/2} - \bar{x}_k \right\rangle \\
&+ \left\langle \epsilon \cdot \mathbf{diag}\left(y_{k-1/2}\right)|\mathbf{A}|^\top x_{k-1/2} - \epsilon \cdot \mathbf{diag}\left(y_{k-1}\right)|\mathbf{A}|^\top x_{k-1}, y_{k-1/2} - \bar{y}_k \right\rangle \\
=&\int_0^1 (z_{k-1/2} - z_{k-1})^\top \begin{pmatrix} \mu \cdot \mathbf{diag}\left(\frac{1}{x_\beta}\right) & -\epsilon|\mathbf{A}|\,\mathbf{diag}\left(y_\beta\right) \\ \epsilon \mathbf{diag}\left(y_\beta\right)|\mathbf{A}|^\top & \epsilon \mathbf{diag}\left(|\mathbf{A}|^\top x_\beta\right) \end{pmatrix} (z_{k-1/2} - \bar{z}_k) d\beta \qquad \text{(E.20)} \\
=&\sqrt{\frac{\mu\epsilon}{2}} \int_0^1 (z_{k-1/2} - z_{k-1})^\top \begin{pmatrix} \rho \cdot \mathbf{diag}\left(\frac{1}{x_\beta}\right) & -\frac{2}{\rho}|\mathbf{A}|\,\mathbf{diag}\left(y_\beta\right) \\ \frac{2}{\rho}\mathbf{diag}\left(y_\beta\right)|\mathbf{A}|^\top & \frac{2}{\rho}\mathbf{diag}\left(|\mathbf{A}|^\top x_\beta\right) \end{pmatrix} (z_{k-1/2} - \bar{z}_k) dt \\
\leq& 8\sqrt{\frac{\mu\epsilon}{2}} \left(\|z_{k-1/2} - z_{k-1}\|^2_{\mathbf{D}(x_{k-1})} + \|z_{k-1/2} - \bar{z}_k\|^2_{\mathbf{D}(x_{k-1})} \right),
\end{aligned}$$

where for the last inequality we use similar arguments as in Lemma 82 to bound the matrix by $4\mathbf{D}(x_\beta) \preceq 8\mathbf{D}(x_{k-1})$ following the assumption that $x_{k-1/2} \in [\frac{1}{2}x_{k-1}, 2x_{k-1}]$ elementwise. Putting (E.19) and (E.20) into (E.18), we conclude that the relative Lipschitz condition (6.20) holds with the stated α. □

Corollary 20 (Inner loop convergence in Algorithm 23)**.** *Assume the same parameter bounds as in Lemma 84. For γ defined in Line 7, suppose for an appropriate constant $T = \Omega\left(\log \frac{mnB_{\max}\alpha\rho}{\delta\varepsilon}\right)$. Then, for all k iterate $z_{k-1/2} = (x_{k-1/2}, y_{k-1/2})$ of Line 8 satisfies*

$$\left\langle \nabla g(z_{k-1}) + \alpha \nabla V_{z_{k-1}}(z_{k-1/2}), z_{k-1/2} - w \right\rangle \leq \frac{\nu\varepsilon}{4}, \text{ for all } w \in \mathcal{Z}.$$

Similarly, for γ defined in Line 9, iterate $\bar{z}_k = (x_{(T+1)}, y_{(T)})$ of Line 10 satisfies

$$\left\langle \nabla g(z_{k-1}) + \alpha \nabla V_{z_{k-1}}(\bar{z}_k) + \nu \nabla V_{z_{k-1/2}}(\bar{z}_k), \bar{z}_k - w \right\rangle \leq \frac{\nu\varepsilon}{4}, \text{ for all } w \in \mathcal{Z}.$$

Proof. We first claim Lemma 81 holds with $\kappa = 8$. This follows from the fact that for any $x' \in \Delta^m$, $y' \in [0,1]^n$ satyisfying $x' \geq \frac{1}{2}x$, we have

$$\nabla^2 r(x', y') \succeq \mathbf{D}(x') \succeq \frac{1}{2}\mathbf{D}(x) \succeq \frac{1}{8}\nabla^2 r(x,y) \succeq \frac{1}{8}\nabla^2_{yy} r(x,y),$$

where we use both of the Hessian bounds from Lemma 82 and the fact that restrictions to blocks only decrease a quadratic form.

For each $\gamma = (\gamma^{\mathsf{x}}, \gamma^{\mathsf{y}})$ defined in Algorithm 23 and the fact that $x_{k-1}, x_{k-1/2} \geq \frac{\delta}{4}$ for all $k \in [K]$,

we have by the assumptions $\|\mathbf{A}\|_\infty \le 1$, $\|c\|_\infty \le C_{\max}$, and $\|b\|_1 \le nB_{\max}$ that it suffices to take

$$B = \max\left(\|\gamma^{\mathsf{x}}\|_\infty, \|\gamma^{\mathsf{y}}\|_1\right) = O\left((\alpha\rho + \mu)\log\frac{1}{\delta} + nB_{\max}\right)$$

in Corollary 18. Hence, the stated number of iterations T suffices. □

Proposition 20 (Convergence of Algorithm 23)**.** *Assume the same parameter bounds as in Lemma 84, and that $\delta \le \frac{\varepsilon}{4\rho\alpha m}$. Algorithm 23 returns z_K satisfying $V^r_{z_K}(z^\star) \le \frac{3\varepsilon}{\nu}$, letting (for an appropriate constant) $K = \Omega(\frac{\alpha}{\nu}\log(\frac{\nu\log m}{\varepsilon}))$.*

Proof. Our proof is based on the proof of Proposition 19. Lemma 85 and Lemma 86 showed that our operator-regularizer pair satisfies ν-strong monotonicity and α-relative Lipschitzness. Under these conditions, and letting $\bar{z}_k$ be the unpadded version of z_k, recall that the proof of Proposition 19 showed that

$$(\nu+\alpha)V_{\bar{z}_k}(z^\star) \le \alpha V_{z_{k-1}}(z^\star) + \varepsilon.$$

Moreover, the padding guarantee of Lemma 83 shows

$$(\nu+\alpha)\left(V_{z_k}(z^\star) - V_{\bar{z}_k}(z^\star)\right) \le (\nu+\alpha)\left(\rho + \frac{8}{\rho}\right)m\delta \le \varepsilon,$$

by the assumption on δ. Following the proof of Proposition 19, we conclude

$$V_{z_K}(z^\star) \le \left(\frac{\alpha}{\nu+\alpha}\right)^K \Theta + \frac{2\varepsilon}{\nu}.$$

Since $\Theta = O(\rho\log m)$ for our choice of regularizer, the claim follows. □

Finally, we prove the main theorem in this section on the convergence of our regularized box-simplex solver for (6.16). We first state two helper lemmas used to prove the theorem.

Lemma 276 (Entropy bounded by ℓ_1-distance)**.** *For any $x, \tilde{x} \in \Delta^m$, defining $H(x) = \sum_{i\in[m]} x_i \log x_i$,*

$$H(\tilde{x}) - H(x) \le 33\log(m)\|\tilde{x} - x\|_1 + m^{-30}.$$

Proof. We define a partition of the coordinates of $\tilde{x}$,

$$L := \left\{i \in [m] \mid \tilde{x}_i \ge m^{-32}\right\} \text{ and } S := \left\{i \in [m] \mid \tilde{x}_i < m^{-32}\right\}.$$

First, we have by convexity of entropy that, letting $v_L \in \mathbb{R}^{|L|}$ denote the $|L|$-dimensional restriction

of a vector $v \in \mathbb{R}^m$ to the coordinates of $L \subseteq [m]$,

$$\begin{aligned}\sum_{i\in L} \tilde{x}_i \log \tilde{x}_i - \sum_{i\in L} x_i \log x_i &\leq \langle \log \tilde{x}_L + \mathbf{1}_L, \tilde{x}_L - x_L \rangle \\ &\leq \|\log \tilde{x}_L + \mathbf{1}_L\|_\infty \|\tilde{x}_L - x_L\|_1 \\ &\leq (32\log(m) + 1)\|\tilde{x} - x\|_1 \leq 33\log(m)\|\tilde{x}_L - x_L\|_1,\end{aligned}$$

for sufficiently large m. In the second line, we used the ℓ_1-ℓ_∞ Hölder's inequality. Moreover, let $S' := \{i \in [m] \mid x_i < m^{-32}\}$ be the subset of $[m]$ where x is small. By nonpositivity of entropy,

$$\begin{aligned}\sum_{i\in S} \tilde{x}_i \log \tilde{x}_i - \sum_{i\in S} x_i \log x_i \leq -\sum_{i\in S} x_i \log x_i &= -\sum_{i\in S'} x_i \log x_i - \sum_{i\in S\setminus S'} x_i \log x_i \\ &\leq \frac{32|S'|\log(m)}{m^{32}} + 32\log(m) \sum_{i\in S\setminus S'} x_i \\ &\leq \frac{32|S|\log(m)}{m^{32}} + 32\log(m) \sum_{i\in S\setminus S'} \left(x_i - m^{-32}\right) \\ &\leq m^{-30} + 32\log(m) \sum_{i\in S\setminus S'} (x_i - \tilde{x}_i) \leq m^{-30} + 32\log(m)\|\tilde{x}_S - x_S\|_1.\end{aligned}$$

The second line used that $-c\log c$ is increasing in the range $(0, m^{-32})$, and the fourth used that $m^{-32} > \tilde{x}_i$ for all $i \in S$. We conclude by combining the above two displays. □

Lemma 277. *Let $(x^\star, y^\star)$ be the optimizer to the regularized box-simplex game in (6.16). Suppose $\rho \geq 6$, $\mu \leq 1$, $\varepsilon \geq m^{-10}$ and that $z = (x, y)$ satisfies $V_z(z^\star) \leq \frac{\varepsilon^2}{C_{\max}^2 \log^4 m}$. Then, x is an ε-approximate minimizer to*

$$f^{\times}_{\mu,\epsilon}(x, y) := \max_{y\in[0,1]^n} f_{\mu,\epsilon}(x, y).$$

Proof. We first claim that the given divergence condition implies that

$$\|x - x^\star\|_1 \leq \frac{\varepsilon}{C_{\max}\log^2 m}. \tag{E.21}$$

To see this, we observe that due to the Hessian bound in Lemma 82, the divergence induced by $r(x, y) - \frac{\rho}{2}H(x)$ is nonnegative, and hence

$$\|x^\star - x\|_1^2 \leq 2V_x^H(x^\star) \leq \frac{\rho}{2}V_x^H(x^\star) \leq V_z^r(z^\star).$$

The first inequality above was by Pinsker, the second used $\rho \geq 6$, and the last followed by nonnegativity of divergence. This implies (E.21) by the assumed divergence bound. We next show that

(E.21) implies that x is an approximate minimizer of $f^{\times}_{\mu,\epsilon}$, which we recall means

$$\max_{y' \in \mathcal{Y}} f_{\mu,\epsilon}(x, y') - f_{\mu,\epsilon}(x^\star, y^\star) \leq \varepsilon. \tag{E.22}$$

To this end, we compute

$$\begin{aligned}
\max_{y \in \mathcal{Y}} y^\top \left(\mathbf{A}^\top x - b\right) - \max_{y \in \mathcal{Y}} y^\top \left(\mathbf{A}^\top x^\star - b\right) \leq \left\|\mathbf{A}^\top (x - x^\star)\right\|_1 \leq \|x - x^\star\|_1 \leq \frac{\varepsilon}{4}, \\
\langle c, x\rangle - \langle c, x^\star\rangle \leq C_{\max} \|x - x^\star\|_1 \leq \frac{\varepsilon}{4}, \\
\mu\left(H(x) - H(x^\star)\right) \leq 33\mu \log(m) \|x - x^\star\|_1 + \mu m^{-30} \leq \frac{\varepsilon}{4}, \\
\max_{y \in \mathcal{Y}} \left(y^2\right)^\top |\mathbf{A}|^\top x - \max_{y \in \mathcal{Y}} \left(y^2\right)^\top |\mathbf{A}|^\top x^\star \leq \left\|\mathbf{A}^\top (x - x^\star)\right\|_1 \leq \|x - x^\star\|_1 \leq \frac{\varepsilon}{4}.
\end{aligned}$$

The third line used Lemma 276. Combining these pieces proves the claim. □

We are now ready to prove Theorem 41.

Theorem 41 (Regularized box-simplex solver)**.** *Given regularized box-simplex game* (6.16) *with* $72\epsilon \leq \mu \leq 1$ *and optimizer* $(x^\star, y^\star)$*, and letting* $\varepsilon \in (m^{-10}, 1)$*,* RegularizedBS *(Algorithm 23) returns* x^K *satisfying* $\left\|x^K - x^\star\right\|_1 \leq \frac{\varepsilon}{C_{\max} \log^2 m}$ *and* $\max_{y \in [0,1]^n} f_{\mu,\epsilon}(x^K, y) - f_{\mu,\epsilon}(x^\star, y^\star) \leq \varepsilon$*. The total runtime of the algorithm is* $O(\mathrm{nnz}(\mathbf{A}) \cdot (\frac{C_{\max}}{\sqrt{\mu\epsilon}} + \log(\frac{m}{\sigma\epsilon})) \cdot \log(\frac{C_{\max} \log m}{\varepsilon}) \log(\frac{mnB_{\max}}{\varepsilon}))$.

Proof. We handle correctness and runtime separately.

Correctness. Let $(x^\star, y^\star)$ be the minimax optimal solution of $f_{\mu,\epsilon}$ in (6.16). By applying Proposition 20 and Lemma 277, shifting the definition of ε appropriately so that $\frac{3\varepsilon}{\nu} \leftarrow \frac{\varepsilon^2}{C^2_{\max} \log^4 m}$, we have with the choice of parameters our returned x_K satisfies the desired bounds.

Runtime. It is immediate to see each line of Algorithm 23 runs in time $O(\mathrm{nnz}(\mathbf{A}))$. Then, the total runtime follows from combining Proposition 20 and Corollary 20. □

We now apply Theorem 41 to prove Corollary 21.

Corollary 21 (Half-regularized approximate solver)**.** *Given regularized box-simplex game* (6.24) *with regularization parameters* $72\epsilon \leq \mu \leq 1$ *and optimizer* $(x^\star, y^\star)$*, and letting* $\epsilon \in (m^{-10}, 1)$*, Algorithm 23 with* $\varepsilon \leftarrow \frac{\epsilon}{2}$ *returns* x^K *satisfying* $\max_{y \in [0,1]^n} f_\mu(x^K, y) - f_\mu(x^\star, y^\star) \leq \epsilon$*. The total runtime of the algorithm is* $O\left(\mathrm{nnz}(\mathbf{A}) \cdot (\frac{C_{\max}}{\sqrt{\mu\epsilon}} + \log\left(\frac{m}{\epsilon}\right)) \cdot \log\left(\frac{C_{\max} \log m}{\epsilon}\right) \log\left(\frac{mnB_{\max}}{\epsilon}\right)\right)$.

Proof. We apply Theorem 41 with $\varepsilon \leftarrow \frac{\epsilon}{2}$ and the same value of μ as in the regularized objective (6.24). Note that the additive range of the quadratic regularizer is $\frac{\epsilon}{2}$, and hence

$$f_{\mu,\epsilon}(x, y) \leq f_\mu(x, y) \leq f_{\mu,\epsilon}(x, y) + \frac{\epsilon}{2}, \text{ for all } (x, y) \in \Delta^m \times [0, 1]^n.$$

It thus suffices to obtain a $\frac{\epsilon}{2}$ error guarantee from Theorem 41, which yields the runtime guarantee. □

E.3 Approximating Sinkhorn distances

In this section, we consider the problem of approximating Sinkhorn distances [151], a common computational task in the theory and practice of optimal transport. In this problem, we are given a product domain on two discrete supports $L \times R$, along with demands $d \in \mathbb{R}^{L+R}_{\geq 0}$ (with marginals $\|d_L\|_1 = \|d_R\|_1 = 1$) and a cost matrix $c \in \mathbb{R}^{L \times R}_{\geq 0}$. Letting $m = |L||R|$ and $n = |L| + |R|$, the optimal transport (OT) problem asks to find a feasible transport plan $x \in \Delta^m$ minimizing $\langle c, x \rangle$, concisely described as

$$\min_{x \in \Delta^m | \mathbf{B}^\top x = d} \langle c, x \rangle , \tag{E.23}$$

where $\mathbf{B}$ is the unsigned incidence matrix of the complete bipartite graph on $L \times R$. Prior work by [151] proposed Sinkhorn distances as an efficiently computable, differentiable alternative to (E.23), where for $\mu > 0$, the Sinkhorn distance asks for an optimal *regularized* transport plan:

$$x^\star_\mu \text{ solves } \min_{x \in \Delta^m | \mathbf{B}^\top x = d} \langle c, x \rangle + \mu H(x), \text{ where } H(x) = \sum_{i \in [m]} x_i \log x_i. \tag{E.24}$$

We use the notion of function error in terms of the regularized objective $\langle c, x \rangle + \mu H(x)$ as the approximation criterion for developing our approximate solvers in this section, and we aim to find an ϵ-approximate solution $x_\mu \in \mathcal{U}(d_L, d_R) := \{x \in \Delta^m \mid \mathbf{B}^\top x = d\}$ satisfying

$$\langle c, x_\mu \rangle + \mu H(x_\mu) \leq \left\langle c, x^\star_\mu \right\rangle + \mu H(x^\star_\mu) + \epsilon. \tag{E.25}$$

We call such an x_μ an ϵ-approximate minimizer to (E.24). One reason for choosing this definition of an approximate solution is that function error bounds the error in transportation cost and ℓ_1 distances, i.e.

$$\frac{\mu}{2} \left\| x_\mu - x^\star_\mu \right\|_1^2 \leq \langle c, x_\mu \rangle + \mu H(x_\mu) - \left(\left\langle c, x^\star_\mu \right\rangle + \mu H(x^\star_\mu) \right),$$
$$\text{and} \quad \langle c, x_\mu \rangle - \left\langle c, x^\star_\mu \right\rangle \leq \|c\|_\infty \left\| x_\mu - x^\star_\mu \right\|_1 .$$

The first inequality follows from the first-order optimality condition of $x^\star_\mu$ and Pinsker's inequality, which shows the objective minimized by $x^\star_\mu$ is μ-strongly convex in ℓ_1. The second is ℓ_1-ℓ_∞ Hölder.

We present three different approaches to approximate Sinkhorn distances (E.24).

1. In Section E.3.1, we develop a first-order method that uses our regularized box-simplex solver, which finds an ϵ-approximate minimizer to (E.24) in $\widetilde{O}(\frac{1}{\sqrt{\mu\epsilon}})$ time for $\mu = \Omega(\epsilon)$ (Theorem 85).

2. In Section E.3.2, we present a high-accuracy solver for the problem by building upon recent near-linear time box-constrained Newton's method for matrix scaling problems [145], which finds an ϵ-approximate minimizer in an improved $\widetilde{O}(\frac{1}{\mu})$ time (Theorem 86).

3. Lastly, we provide in Appendix E.3.3 an unaccelerated Sinkhorn distance solver with runtime $\widetilde{O}(\frac{1}{\mu\epsilon})$ using the classical Sinkhorn method [208] (Corollary 63).

E.3.1 Regularized box-simplex solver for computing Sinkhorn distances: Theorem 85

In this section, we summarize our application of Theorem 41 in approximating the Sinkhorn distance (E.24). We first show approximating (E.24) can be reduced to solving a *regularized box-simplex game*:

$$\min_{x\in\Delta^m}\max_{y\in[0,1]^{2n}} \frac{1}{4C}\langle c,x\rangle + \frac{\mu}{4C}H(x) + y^\top \mathbf{A}^\top x - \langle b,y\rangle, \\ \text{where } \mathbf{A} := \frac{1}{4}\left(\mathbf{B},-\mathbf{B}\right),\ b := \frac{1}{4}\left(d,-d\right), \tag{E.26}$$

for some $C := 2\left(\|c\|_\infty + 33\mu\log(m)\right)$. This characterization, which is key to our approach, follows by first replacing the equality constraints $\mathbf{B}^\top x = d$ with inequality constraints $\mathbf{B}^\top x \le d, \mathbf{B}^\top x \ge d$, and then developing an analogous reduction to the one used by [293] for solving the standard OT objective (E.23). In particular, by maximizing over y, problem (E.26) is equivalent to the following regularized ℓ_1-regression problem:

$$\min_{x\in\Delta^m} \frac{1}{4C}\langle c,x\rangle + \frac{\mu}{4C}H(x) + \frac{1}{4}\left\|\mathbf{B}^\top x - d\right\|_1. \tag{E.27}$$

The quantity $\left\|\mathbf{B}^\top x - d\right\|_1$ is naturally interpreted as the deviation of the x marginals from the demands d. We further require one simple helper fact, which shows that any transport plan x can be "fixed" to be feasible for demands d at a cost depending on its deviation $\left\|\mathbf{B}^\top x - d\right\|_1$.

Lemma 278 (Lemma 7, [25])**.** *There is an algorithm,* OTRound *(Algorithm 2, [25]), which takes as input any $x \in \mathbb{R}^m_{\ge 0}$ and produces $\tilde{x} \in \Delta^m$ such that $\|\tilde{x} - x\|_1 \le 2\left\|\mathbf{B}^\top x - d\right\|_1$ and $\mathbf{B}^\top \tilde{x} = d$. The algorithm runs in $O(m)$ time.*

The guarantees of OTRound (cf. Lemma 278) give an algorithmic proof of equivalence between (E.27) and (E.24) by relating their approximate solutions: Lemma 279 gives a formal statement.

Lemma 279. *Let x be a $\frac{1}{4C}\Delta$-approximate minimizer to* (E.27)*. Then, letting $\tilde{x}$ be the result of* OTRound(x) *(cf. Lemma 278), $\tilde{x}$ is a $\Delta + \mu m^{-30}$-approximate minimizer to* (E.24)*.*

Proof. Define $\mathrm{OPT}_{\text{Sinkhorn}}$ to be the value of the Sinkhorn objective (E.24), and $\mathrm{OPT}_{\text{Sinkhorn-Pen}}$ to be the value of (E.27). It is immediate that $\mathrm{OPT}_{\text{Sinkhorn-Pen}} \le \frac{1}{4C}\mathrm{OPT}_{\text{Sinkhorn}}$, since any feasible point for (E.24) (in particular the minimizer) is also feasible for (E.27) and obtains the same function value. Hence, we have

$$\langle c,x\rangle + \mu H(x) + C\left\|\mathbf{B}^\top x - d\right\|_1 \le 4C\cdot\mathrm{OPT}_{\text{Sinkhorn-Pen}} + \Delta \le \mathrm{OPT}_{\text{Sinkhorn}} + \Delta.$$

Moreover, we bound the objective value of $\tilde{x}$:

$$\begin{aligned}\langle c, \tilde{x}\rangle + \mu H(\tilde{x}) &= \langle c, x\rangle + \mu H(x) + \langle c, \tilde{x} - x\rangle + \mu\left(H(\tilde{x}) - H(x)\right)\\ &\leq \langle c, x\rangle + \mu H(x) + \left(\|c\|_\infty + 33\mu\log(m)\right)\|\tilde{x} - x\|_1 + \mu m^{-30}\\ &\leq \langle c, x\rangle + \mu H(x) + 2\left(\|c\|_\infty + 33\mu\log(m)\right)\left\|\mathbf{B}^\top x - d\right\|_1 + \mu m^{-30}\\ &= \langle c, x\rangle + \mu H(x) + C\left\|\mathbf{B}^\top x - d\right\|_1 + \mu m^{-30}.\end{aligned}$$

In the second line, we applied ℓ_1-ℓ_∞ Hölder and Lemma 276, in the third we used Lemma 278, and in the last we used the definition of C. Combining the above displays, we have the desired

$$\langle c, \tilde{x}\rangle + \mu H(\tilde{x}) \leq \mathrm{OPT}_{\mathrm{Sinkhorn}} + \Delta + \mu m^{-30}.$$

□

Lemma 279 shows that to obtain an $\epsilon \geq 2\mu m^{-30}$-approximate minimizer of (E.24), it suffices to solve (E.27) to $\frac{1}{8C}\cdot\epsilon$ additive accuracy. Applying Corollary 21 in Section 6.4.3, we thus develop a solver for minimizing the Sinkhorn distance objective (E.24) with the following guarantees.

Theorem 85 (Regularized box-simplex solver for Sinkhorn distances)**.** *Consider the Sinkhorn distance objective* (E.24) *on an instance with $n = |L| + |R|$ vertices and $m = |L||R|$ edges, and suppose $\epsilon \in [m^{-5}\|c\|_\infty, \|c\|_\infty]$, $\mu \in [36\epsilon, \|c\|_\infty]$. Using* OTRound *on the result of Corollary 21 applied to* (E.26) *obtains an ϵ-approximate minimizer to* (E.24) *in time*

$$O\left(m\left(\sqrt{\frac{\mu}{\epsilon}}\log(m) + \frac{\|c\|_\infty}{\sqrt{\mu\epsilon}}\right)\log(m)\log\left(\frac{\|c\|_\infty \log m}{\epsilon}\right)\right).$$

Proof. Note that the deviation between the regularized Sinkhorn objective (E.26) and the actual Sinkhorn objective (E.24) at an approximate minimizer is at most $\mu m^{-30} \leq \frac{\epsilon}{2} =: \Delta$ by Lemma 279. Thus, it suffices to find a ε-approximate minimizer to (E.26) with the choice of parameters

$$\mu' \leftarrow \frac{\mu}{4C} = \frac{\mu}{8\left(\|c\|_\infty + 33\mu\log(m)\right)}, \quad \text{and} \quad \varepsilon \leftarrow \frac{\epsilon}{8C} = \frac{\epsilon}{16\left(\|c\|_\infty + 33\mu\log(m)\right)}.$$

It is straightforward to verify that the assumptions at the start of Section 6.4 on $\|\mathbf{A}\|_\infty$, $\|b\|_1$, and $\|c\|_\infty$ are all met for this problem and the desired accuracy parameter. Thus, by applying Corollary 21, we obtain the desired runtime. □

E.3.2 Box-constrained Newton for computing Sinkhorn distances: Theorem 86

In this section, we show how to apply the recent matrix scaling solvers developed in [145] to obtain second-order high-accuracy solvers for approximating Sinkhorn distances. Our approach follows from a direct connection to the scaling problem via the well-known correspondence of matrix scaling and computing Sinkhorn distances: we provide the following results to make this reduction from matrix scaling rigorous for completeness, and to prove our claimed runtime in Theorem 86.

We first define the matrix scaling problem.

Definition 55 (Matrix scaling)**.** *Given a nonnegative matrix $\mathbf{K} \in \mathbb{R}_{\geq 0}^{n_1 \times n_2}$, the (r, c)-matrix scaling problem asks to find the vectors $u \in \mathbb{R}^{n_1}$, $v \in \mathbb{R}^{n_2}$ such that*

$$\mathbf{diag}\left(\exp(u)\right) \cdot \mathbf{K} \cdot \mathbf{diag}\left(\exp(v)\right) \mathbf{1} = r,\ \mathbf{1}^\top \mathbf{diag}\left(\exp(u)\right) \cdot \mathbf{K} \cdot \mathbf{diag}\left(\exp(v)\right) = c^\top.$$

In other words, it asks to rescale $\mathbf{K}$ by nonnegative diagonal matrices to obtain the specified row and column marginals. Prior work by [145] gives a near-linear time matrix scaling solver by using a box-constrained Newton's method, which we restate in Proposition 63 for our use.

Proposition 63 (Theorem 4.6, [145])**.** *Given $\mathbf{K} \in \mathbb{R}_{\geq 0}^{n_1 \times n_2}$ with $s_{\mathbf{K}} := \sum_{i \in [n_1], j \in [n_2]} \mathbf{K}_{ij}$, define*

$$P_{\mathbf{K}}(u, v) := \sum_{i \in [n_1], j \in [n_2]} \mathbf{K}_{ij} \exp(u_i - v_j) - \langle r, u \rangle + \langle c, v \rangle. \tag{E.28}$$

Suppose there is an approximate (r, c)-scaling (u, v) satisfying $P_{\mathbf{K}}(u, v) \leq \inf_{u', v'} P_{\mathbf{K}}(u', v') + \frac{\varepsilon^2}{n_1 + n_2}$ and $\|u\|_\infty, \|v\|_\infty \leq B$. Then we can compute an approximate (r, c) scaling (u, v) of $\mathbf{K}$ satisfying

$$\left\|\mathbf{diag}\left(\exp(u)\right) \cdot \mathbf{K} \cdot \mathbf{diag}\left(\exp(v)\right) \mathbf{1} - r\right\|_2^2 + \left\|\mathbf{diag}\left(\exp(v)\right) \cdot \mathbf{K}^\top \cdot \mathbf{diag}\left(\exp(u)\right) \mathbf{1} - c\right\|_2^2 \leq \varepsilon,$$

in time $\widetilde{O}(\mathrm{nnz}\left(\mathbf{K}\right) \cdot B \log^2(\frac{s_{\mathbf{K}}}{\varepsilon}))$.

It is well-known that computing Sinkhorn distances is equivalent to finding the optimal matrix scaling for $\mathbf{K} := \exp(-\frac{1}{\mu}\mathbf{C}) \in \mathbb{R}^{L \times R}$, where $\mathbf{C} = (c_{ij})_{i \in L, j \in R}$ comes from reshaping the cost vector (see e.g. Lemma 2, [151]). Consequently, we can apply Proposition 63 with $\varepsilon := \frac{\epsilon^2}{8\|c\|_\infty^2 m}$ and OTRound (Lemma 278) to obtain a high-precision solver for the Sinkhorn distance.

Theorem 86 (Box-constrained Newton solver for Sinkhorn distances)**.** *Consider the Sinkhorn distance objective* (E.24) *on an instance with $n = |L| + |R|$ vertices and $m = |L||R|$ edges, and suppose $\epsilon \in [m^{-5}\|c\|_\infty, \|c\|_\infty]$ and $\mu \in (0, \|c\|_\infty]$. The algorithm in Proposition 63 obtains an ϵ-approximate minimizer to* (E.24) *in time*

$$\widetilde{O}\left(\frac{m\|c\|_\infty}{\mu}\right).$$

To prove the theorem, we first require one helper lemma.

Lemma 280. *Let $d_L \in \Delta^L, d_R \in \Delta^R$ be demands, let $\mathbf{C} \in \mathbb{R}_{\geq 0}^{L \times R}$ be costs, and define $\mathbf{K} := \exp(-\frac{1}{\mu}\mathbf{C})$ for some $\mu \geq 0$. Let $x \in \mathbb{R}_{\geq 0}^m$ be $\mathbf{X} \in \mathbb{R}^{L \times R}$ appropriately vectorized, such that $\mathbf{X} = \mathbf{diag}(u) \cdot \mathbf{K} \cdot \mathbf{diag}(v)$ for some $u \in \mathbb{R}_{\geq 0}^L$, $v \in \mathbb{R}_{\geq 0}^R$. Assume that*

$$\|\mathbf{X}\mathbf{1} - d_L\|_1 + \left\|\mathbf{X}^\top \mathbf{1} - d_R\right\|_1 \leq \Delta.$$

Then, $\hat{x} := \frac{x}{\|x\|_1}$ is a $3C\Delta + \mu m^{-30}$-approximate minimizer to (E.24) with demands d, where $C := 2(\|c\|_\infty + 33\mu \log(m))$.

Proof. Clearly, $\hat{x}$ is also a diagonal scaling of $\mathbf{K}$, by multiplying u or v by appropriate scalars, and

$$\|x - \hat{x}\|_1 = |\|x\|_1 - 1| = \left|\mathbf{1}^\top \mathbf{X}\mathbf{1} - \mathbf{1}^\top d_L\right| \leq \|\mathbf{X}\mathbf{1} - d_L\|_1 \leq \Delta.$$

This along with Lemma 2 of [151] implies that $\hat{x}$, as a diagonal scaling of $\mathbf{K}$, is the optimal solution to (E.24) for some demands $\hat{d}_L = \widehat{\mathbf{X}}\mathbf{1}$, $\hat{d}_R = \widehat{\mathbf{X}}^\top \mathbf{1}$ such that

$$\left\|\hat{d} - d\right\|_1 = \|\mathbf{X}\mathbf{1} - d_L\|_1 + \left\|\mathbf{X}^\top \mathbf{1} - d_R\right\|_1 \leq \Delta + \left\|(\mathbf{X} - \widehat{\mathbf{X}})\mathbf{1}\right\|_1 + \left\|(\mathbf{X} - \widehat{\mathbf{X}})^\top \mathbf{1}\right\|_1 \leq 3\Delta.$$

The remainder of the proof follows from Lemma 281. □

Next, Lemma 280 is key to proving Theorem 86 below.

Proof of Theorem 86. Let $C := 2\left(\|c\|_\infty + 33\mu \log(m)\right)$. We first do a preprocessing step to pad the demands d_L, d_R to the modified demands $\tilde{d}_L = \mathcal{O}_\delta(d_L)$ and $\tilde{d}_R = \mathcal{O}_\delta(d_R)$ (see Definition 26), where $\delta := \frac{\epsilon}{48mC}$. Next, suppose for $\varepsilon := \frac{\epsilon^2}{288C^2m}$, we find a ε-approximate $(\tilde{d}_L, \tilde{d}_R)$-scaling (u, v) in the sense of Proposition 63, such that $\mathbf{X} := \mathbf{diag}(u) \cdot \mathbf{K} \cdot \mathbf{diag}(v)$ satisfies

$$\left\|\mathbf{X}\mathbf{1} - \tilde{d}_L\right\|_1 + \left\|\mathbf{X}^\top 1 - \tilde{d}_R\right\|_1 \leq \sqrt{2m}\sqrt{\left\|\mathbf{X}\mathbf{1} - \tilde{d}_L\right\|_2^2 + \left\|\mathbf{X}^\top 1 - \tilde{d}_R\right\|_2^2} \leq \sqrt{2m\varepsilon} = \frac{\epsilon}{12C}.$$

Recalling that the guarantees of padding imply

$$\left\|d_L - \tilde{d}_L\right\|_1 + \left\|d_R - \tilde{d}_R\right\|_1 \leq 2m\delta + 2m\delta = \frac{\epsilon}{12C},$$

we have by the triangle inequality, letting $x \in \mathbb{R}_{\geq 0}^m$ be the appropriately vectorized $\mathbf{X}$, and $\hat{x} := \frac{x}{\|x\|_1}$,

$$\begin{aligned}\left\|\mathbf{B}^\top \hat{x} - d\right\|_1 &\leq \left\|\mathbf{X}\mathbf{1} - \tilde{d}_L\right\|_1 + \left\|d_L - \tilde{d}_L\right\|_1 + \left\|\mathbf{X}^\top 1 - \tilde{d}_R\right\|_1 + \left\|d_R - \tilde{d}_R\right\|_1 + 2\left\|\hat{x} - x\right\|_1 \\ &\leq \frac{\epsilon}{3C}.\end{aligned}$$

In the last inequality, we bound (see the proof of Lemma 280) $\|x - \hat{x}\|_1 \leq \frac{\epsilon}{12C}$. Next, let x' be the plan which results after applying OTRound for $\hat{x}$ and demands d, and let $\tilde{x}$ be the resulting plan when applying OTRound for $x_\mu^\star$ and demands $\tilde{d}$, where $x_\mu^\star$ is optimal for (E.24) with demands d. By a similar argument as used in the proof of Lemma 279, we have

$$\begin{aligned} \langle c, \tilde{x} \rangle + \mu H(\tilde{x}) &\leq \langle c, x_\mu^\star \rangle + \mu H(x_\mu^\star) + C \left\| \mathbf{B}^\top x_\mu^\star - \tilde{d} \right\|_1 + \mu m^{-30}, \\ \langle c, x' \rangle + \mu H(x') &\leq \langle c, \hat{x} \rangle + \mu H(\hat{x}) + C \left\| \mathbf{B}^\top \hat{x} - d \right\|_1 + \mu m^{-30}. \end{aligned} \tag{E.29}$$

Moreover, letting $\tilde{x}_\mu^\star$ be optimal for (E.24) with demands $\tilde{d}$, applying Lemma 280 with $d \leftarrow \tilde{d}$ and $\Delta \leftarrow \frac{\epsilon}{12C}$, we obtain

$$\langle c, \hat{x} \rangle + \mu H(\hat{x}) \leq \langle c, \tilde{x}_\mu^\star \rangle + \mu H(\tilde{x}_\mu^\star) + \frac{\epsilon}{4} + \mu m^{-30} \leq \langle c, \tilde{x} \rangle + \mu H(\tilde{x}) + \frac{\epsilon}{4} + \mu m^{-30}, \tag{E.30}$$

since $\tilde{x}$ is feasible for (E.24) with demands $\tilde{d}$. Finally,

$$\left\| \mathbf{B}^\top x_\mu^\star - \tilde{d} \right\|_1 = \left\| d - \tilde{d} \right\|_1 \leq \frac{\epsilon}{12C}. \tag{E.31}$$

Combining (E.29), (E.30), and (E.31), as well as our assumed ranges on ϵ and μ, we obtain

$$\begin{aligned} \langle c, x' \rangle + \mu H(x') &\leq \langle c, x_\mu^\star \rangle + \mu H(x_\mu^\star) + C \left\| \mathbf{B}^\top x_\mu^\star - \tilde{d} \right\|_1 + C \left\| \mathbf{B}^\top \hat{x} - d \right\|_1 + \frac{\epsilon}{4} + 3\mu m^{-30} \\ &\leq \langle c, x_\mu^\star \rangle + \mu H(x_\mu^\star) + \frac{\epsilon}{12} + \frac{\epsilon}{3} + \frac{\epsilon}{4} + 3\mu m^{-30} \leq \langle c, x_\mu^\star \rangle + \mu H(x_\mu^\star) + \epsilon. \end{aligned}$$

This proves the correctness of returning the transport plan x'.

The runtime is bounded by one call to the matrix scaling solver, and one call to OTRound (which clearly does not dominate). To parameterize Proposition 63, the matrix scaling problem with $\mathbf{K} = \exp(-\frac{1}{\mu}\mathbf{C}) \in \mathbb{R}^{L \times R}$ satisfies $s_{\mathbf{K}} \leq m$ since $c \geq 0$ entrywise. Furthermore, there are optimal $(\tilde{d}_L, \tilde{d}_R)$-scaling vectors $(u^\star, v^\star)$ of $\mathbf{K}$ satisfying (see e.g. Lemma 10, [82]):

$$\|u^\star\|_\infty, \|v^\star\|_\infty = O\left(\frac{\|c\|_\infty}{\mu} + \log \frac{m}{\min_{i \in L, j \in R} \left([\tilde{d}_L]_i, [\tilde{d}_R]_j \right)} \right) = O\left(\frac{\|c\|_\infty}{\mu} + \log \frac{m \|c\|_\infty}{\epsilon} \right).$$

Thus by applying Proposition 63 with $\varepsilon = \frac{\epsilon^2}{288 C^2 m}$, the runtime is bounded as desired. □

E.3.3 Unaccelerated rate of Sinkhorn's algorithm

In this section, we demonstrate that Sinkhorn iteration obtains a complexity for approximately optimizing (E.24) scaling as $\widetilde{O}(\frac{\|c\|_\infty^2}{\mu\epsilon})$, the natural unaccelerated counterpart to Theorem 85. We note that this rate for solving (E.24) (up to logarithmic factors) appeared as Theorem 5 of [24], and

we provide a proof for completeness. We consider an algorithm and combine an analysis of Sinkhorn due to [208] and a technical lemma developed in this paper (Lemma 276 from Appendix E.2.3), which we use to reduce (E.24) to a regularized box-simplex game up to negligible (inverse-polynomial) additive error.

Throughout, we assume $\epsilon \leq \mu \leq \|c\|_\infty$, and further that $\frac{\|c\|_\infty}{\epsilon} \leq m^5$; otherwise, the runtime of cutting-plane methods [296] subsume all runtimes claimed in this paper for approximately solving (E.24). Finally, throughout this section we will fix the values of ϵ, μ, and $c \in \mathbb{R}^m_{\geq 0}$, and define for demands d the optimal value of (E.24) by

$$\mathrm{OPT}_\mu(d) := \min_{x \in \Delta^m, \mathbf{B}^\top x = d} \langle c, x \rangle + \mu H(x).$$

We first bound the difference between $\mathrm{OPT}_\mu(d)$ and $\mathrm{OPT}_\mu(d')$ for "nearby" d and d'.

Lemma 281. *For any demand vectors $d, d' \in \mathbb{R}^{L+R}_{\geq 0}$ with blockwise restrictions in Δ^L and Δ^R,*

$$\mathrm{OPT}_\mu(d) \leq \mathrm{OPT}_\mu(d') + (2\|c\|_\infty + 66\mu \log(m)) \|d - d'\|_1 + \mu m^{-30}.$$

Proof. Throughout the proof define $\Delta := \|d - d'\|_1$. Let x satisfying $\mathbf{B}^\top x = d$ achieve the value $\mathrm{OPT}_\mu(d)$, and similarly let x' with $\mathbf{B}^\top x = d'$ achieve $\mathrm{OPT}_\mu(d')$. By Lemma 278, there is a plan $\tilde{x}$ with $\mathbf{B}^\top \tilde{x} = d'$ and $\|x - \tilde{x}\|_1 \leq 2\Delta$. Since x' minimizes the Sinkhorn objective with demands d',

$$\begin{aligned}
\langle c, x' \rangle + \mu H(x') &= \langle c, \tilde{x} \rangle + \mu H(\tilde{x}) \\
&= \langle c, x \rangle + \mu H(x) + (\langle c, \tilde{x} - x \rangle + \mu H(\tilde{x}) - \mu H(x)) \\
&\leq \langle c, x \rangle + \mu H(x) + (2\|c\|_\infty + 66\mu \log(m)) \Delta + \mu m^{-30}.
\end{aligned}$$

The only inequality used ℓ_1-ℓ_∞ Hölder's, and Lemma 276. □

For the remainder of the section, we will fix a particular d which we wish to optimize the Sinkhorn objective for (and refer to all other demands by d', $\tilde{d}$, etc.). We begin by defining the "padded" demand vector $\tilde{d} \in \mathbb{R}^{L+R}_{\geq 0}$ obtained from modifying d as follows:

$$\tilde{d}_L = \frac{\max(d_L, m^{-20})}{\|\max(d_L, m^{-20})\|_1}, \; \tilde{d}_R = \frac{\max(d_R, m^{-20})}{\|\max(d_R, m^{-20})\|_1}. \tag{E.32}$$

By observation, $\|\tilde{d} - d\|_1 \leq 2m^{-19}$. Next, we recall a result from the literature on the performance of Sinkhorn's algorithm.

Proposition 64 (Theorem 1, [208])**.** *There is an algorithm,* Sinkhorn*, which given demand vector $\tilde{d}$ entrywise at least m^{-21}, returns x' such that*

$$x' = \mathrm{argmin}_{x \in \Delta^m, \mathbf{B}^\top x = d'} \langle c, x \rangle + \mu H(x)$$

for some d' with $\|d' - d\|_1 \le \Delta$, in time

$$O\left(m\left(\frac{\|c\|_\infty}{\mu\Delta} + \frac{\log m}{\Delta}\right)\right).$$

Finally, we give our main result on the performance of Sinkhorn's algorithm for optimizing (E.24).

Corollary 63. *For* (E.24) *on an instance with n vertices and m edges, suppose $\epsilon \le \mu \le \|c\|_\infty$ and $\frac{\|c\|_\infty}{\epsilon} \le m^5$. Using* Sinkhorn *(Proposition 64) on $\tilde{d}$ defined in* (E.32) *to accuracy*

$$\Delta := \frac{\epsilon}{10\|c\|_\infty + 330\mu\log(m)}$$

and then applying OTRound *(Lemma 278) with demands d results in a vector x with $\mathbf{B}^\top x = d$ and*

$$\langle c, x\rangle + \mu H(x) \le \mathrm{OPT}_\mu(d) + \epsilon$$

in time

$$O\left(\frac{\|c\|_\infty^2}{\mu\epsilon} + \frac{\mu\log^2 m}{\epsilon}\right).$$

Proof. By Proposition 64, we obtain a vector $x' \in \Delta^m$ satisfying $\mathbf{B}^\top x' = d'$, $\langle c, x\rangle + \mu H(x)$, and $\|d' - d\|_1 \le \left\|d' - \tilde{d}\right\|_1 + \left\|\tilde{d} - d\right\|_1 \le \Delta + 2m^{-19}$. Moreover, letting $x \leftarrow \mathsf{OTRound}(x')$ for demands d, we have by Lemma 278 that $\|x - x'\|_1 \le 2\Delta + 4m^{-19}$, implying

$$\begin{aligned}\langle c, x\rangle + \mu H(x) &\le \langle c, x'\rangle + \mu H(x') + (\|c\|_\infty + 33\mu\log(m))\left(2\Delta + 4m^{-19}\right) + \mu m^{-30}\\ &\le \langle c, x'\rangle + \mu H(x') + \frac{\epsilon}{2} = \mathrm{OPT}_\mu(d') + \frac{\epsilon}{2}.\end{aligned}$$

Moreover, Lemma 281 implies that

$$\mathrm{OPT}_\mu(d') \le \mathrm{OPT}_\mu(d) + (2\|c\|_\infty + 66\mu\log(m))\left(2\Delta + 4m^{-19}\right) + \mu m^{-30} \le \mathrm{OPT}_\mu(d) + \frac{\epsilon}{2}.$$

The conclusion follows from combining the above two displays. □

Appendix F

Deferred proofs from Chapter 7

F.1 Deferred proofs from Section 7.2

F.1.1 Dual averaging regret bounds

Theorem 45. *Let* $\mathbf{G}_1, \dots, \mathbf{G}_T$ *be* any *sequence of gain matrices in* S_n *and let* $\bar{\mathbf{X}}_t = \bar{\mathsf{P}}(\eta \sum_{i=1}^{t-1} \mathbf{G}_i)$ *as in Eq.* (7.14)*. Then, for every* $T \in \mathbb{N}$*,*

$$\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle \leq \frac{\log(4n)}{\eta} + \frac{3\eta}{2} \cdot \sum_{t=1}^{T} \|\mathbf{G}_t\|_\infty^2 \,. \tag{7.16}$$

If additionally $0 \preceq \mathbf{G}_t \preceq I$ *for every* t *and* $\eta \leq \frac{1}{6}$*,*

$$\lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right) - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle \leq \frac{\log(4n)}{\eta} + 3\eta \cdot \lambda_{\max}\left(\sum_{t=1}^{T} \mathbf{G}_t\right). \tag{7.17}$$

Proof. We start with the well-known Bregman 3-point identity, valid for any $\Phi_0, \Phi_1, \Phi_2 \in S_n$,

$$\left\langle \Phi_2 - \Phi_1, \bar{\mathsf{P}}(\Phi_0) - \bar{\mathsf{P}}(\Phi_1) \right\rangle = \bar{V}_{\Phi_0}(\Phi_1) - \bar{V}_{\Phi_0}(\Phi_2) + \bar{V}_{\Phi_1}(\Phi_2); \tag{F.1}$$

the identity follows from the definition (7.15) of $\bar{V}$ by direct substitution. Fix some $\mathbf{S} \in \operatorname{relint} \varDelta_n$ and $\mathbf{S} \in S_n$ such that $\mathbf{S} = \bar{\mathsf{P}}(\Psi)$ (which exists by Proposition 21.4). Let $\mathbf{Y}_t = \eta \sum_{i=1}^{t-1} \mathbf{G}_i$ so that $\bar{\mathbf{X}}_t = \bar{\mathsf{P}}(\mathbf{Y}_t)$. For a given t, we use the 3-point identity with $\Phi_0 = \Psi, \Phi_1 = \mathbf{Y}_t$ and $\Phi_2 = \mathbf{Y}_{t+1}$, yielding

$$\eta \left\langle \mathbf{G}_t, \mathbf{S} - \bar{\mathbf{X}}_t \right\rangle = \left\langle \mathbf{Y}_{t+1} - \mathbf{Y}_t, \bar{\mathsf{P}}(\Psi) - \bar{\mathsf{P}}(\mathbf{Y}_t) \right\rangle = \bar{V}_\Psi(\mathbf{Y}_t) - \bar{V}_\Psi(\mathbf{Y}_{t+1}) + \bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_{t+1}).$$

Summing these equalities over $t = 1, \dots, T$ and dividing by η gives

$$\left\langle \sum_{t=1}^{T} \mathbf{G}_t, \mathbf{S} \right\rangle - \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle = \frac{\bar{V}_\Psi(\mathbf{Y}_1) - \bar{V}_\Psi(\mathbf{Y}_{T+1})}{\eta} + \frac{1}{\eta} \sum_{t=1}^{T} \bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_{t+1}) \tag{F.2}$$

$$\leq \frac{\log 4n}{\eta} + \frac{3\eta}{2} \sum_{t=1}^{T} \|\mathbf{G}_t\|_\infty^2 . \tag{F.3}$$

Above, we used $\bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_{t+1}) = \bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_t + \eta \mathbf{G}_t) \leq \frac{3}{2}\eta^2 \|\mathbf{G}_t\|_\infty^2$ (Proposition 21.1) along with $\mathbf{Y}_1 = 0$ and $\bar{V}_\Psi(0) - \bar{V}_\Psi(\mathbf{Y}_{T+1}) \leq \log 4n$ (Proposition 21.3).

Since the bound (F.3) is valid for any $\mathbf{S} \in \operatorname{relint} \varDelta_n$, we may supremize it over $\mathbf{S}$. The result (7.16) follows from noting that $\sup_{S \in \operatorname{relint} \varDelta_n} \left\langle \sum_{t=1}^{T} \mathbf{G}_t, \mathbf{S} \right\rangle = \lambda_{\max}\big(\sum_{i=1}^{T} \mathbf{G}_t \big)$.

To see the second bound (7.17), we return to the identity (F.2) and note that the assumptions $0 \preceq \mathbf{G}_t \preceq I$ and $\eta \leq \frac{1}{6}$ imply $\|\eta \mathbf{G}_t\|_\infty \leq \frac{1}{6}$. Therefore we may use Proposition 21.2 to obtain

$$\bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_{t+1}) = \bar{V}_{\mathbf{Y}_t}(\mathbf{Y}_t + \eta \mathbf{G}_t) \leq 3 \|\eta \mathbf{G}_t\|_\infty \left\langle \eta \mathbf{G}_t, \bar{\mathsf{P}}(\mathbf{Y}_t) \right\rangle = 3\eta^2 \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle .$$

Substituting back into (F.2), rearranging and taking the supremum over $\mathbf{S}$ as before, we obtain

$$\lambda_{\max}\left(\sum_{i=1}^{T} \mathbf{G}_t \right) \leq (1 + 3\eta) \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle + \frac{\log(4n)}{\eta}. \tag{F.4}$$

Dividing through by $(1 + 3\eta)$ and noting that $1 - x \leq \frac{1}{1+x} \leq 1$ for every $x \geq 0$, we obtain the result (7.17), concluding the proof. □

F.1.2 High probability regret bounds

Corollary 23. *Let $\mathbf{G}_1, \dots, \mathbf{G}_T$ be symmetric gain matrices satisfying Assumption 5 and let $\mathbf{X}_t$ be generated according to Eq. (7.13). If $\|\mathbf{G}_t\|_\infty \leq 1$ for every t, then for every $T \in \mathbb{N}$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\lambda_{\max}\left(\sum_{i=1}^{T} \mathbf{G}_t \right) - \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \leq \frac{\log(4n)}{\eta} + \frac{3\eta}{2} T + \sqrt{2T \log \tfrac{1}{\delta}}. \tag{7.18}$$

If additionally $0 \preceq \mathbf{G}_t \preceq I$ for every t and $\eta \leq \frac{1}{6}$, then with probability at least $1 - \delta$,

$$\lambda_{\max}\left(\sum_{i=1}^{T} \mathbf{G}_t \right) - \sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \leq \frac{\log(4n/\delta)}{\eta} + 4\eta\, \lambda_{\max}\left(\sum_{i=1}^{T} \mathbf{G}_t \right). \tag{7.19}$$

Proof. We start with the first claim (7.18). Recall that a random process D_t adapted to a filtration $\mathcal{F}_t$ is σ^2-sub-Gaussian if $\mathbb{E}[\exp(\lambda D_t) \mid \mathcal{F}_{t-1}] \leq \exp(\lambda^2 \sigma^2 / 2)$ for all $\lambda \in \mathbb{R}$. Then using the boundedness

assumption that $\langle \mathbf{G}_t, \mathbf{X}_t \rangle \leq \|\mathbf{G}_t\|_\infty \leq 1$, Hoeffding's lemma on bounded random variables implies that the martingale difference sequence $\langle \mathbf{G}_t, \mathbf{X}_t - \bar{\mathbf{X}}_t \rangle$ is 1-sub-Gaussian. Consequently, the Azuma-Hoeffding inequality immediately implies that

$$\sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \geq \sum_{t=1}^{T} \langle \mathbf{G}_t, \bar{\mathbf{X}}_t \rangle - \sqrt{2T \log \tfrac{1}{\delta}} \quad \text{w.p. } \geq 1 - \delta.$$

The bound (7.16) in Theorem 45 thus gives the result (7.18).

For the multiplicative bound (7.19), we require a slightly different relative martingale convergence guarantee.

Lemma 282 ([20], Lemma G.1)**.** *Let $\{D_t\}$ be adapted to the filtration $\{\mathcal{F}_t\}$ and satisfy $0 \leq D_t \leq 1$. Then, for any $\delta, \mu \in (0,1)$, and any $T \in \mathbb{N}$,*

$$\mathbb{P}\left(\sum_{t=1}^{T} D_t \geq (1-\mu) \sum_{t=1}^{T} \mathbb{E}\left[D_t \mid \mathcal{F}_{t-1} \right] - \frac{\log \frac{1}{\delta}}{\mu} \right) \geq 1 - \delta.$$

Similarly, the assumption $0 \preceq \mathbf{G}_t \preceq \mathbf{I}$, along with $\mathbf{X}_t \in \Delta_n$, imply $0 \leq \langle \mathbf{G}_t, \mathbf{X}_t \rangle \leq 1$. Therefore, the conditions of Lemma 282 hold for $D_t = \langle \mathbf{G}_t, \mathbf{X}_t \rangle$, and we use it with $\mu = \eta \leq 1$, obtaining

$$\sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \geq (1-\eta) \sum_{t=1}^{T} \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t \right\rangle - \frac{\log \frac{1}{\delta}}{\eta} \text{ w.p. } \geq 1 - \delta.$$

The bound (7.17) in Theorem 45 thus yields that with probability at least $1-\delta$ over the randomness in $\mathbf{X}_t$ and $\mathbf{G}_t$,

$$\sum_{t=1}^{T} \langle \mathbf{G}_t, \mathbf{X}_t \rangle \geq (1-\eta)(1-3\eta) \lambda_{\max} \left(\sum_{t=1}^{T} \mathbf{G}_t \right) - \frac{\log(4n/\delta)}{\eta}.$$

Noting that $(1-\eta)(1-3\eta) \geq 1 - 4\eta$ completes the proof. □

F.1.3 Proof of Lemma 7.2.2

For any $\mathbf{Y}, \mathbf{D} \in S_n$, $\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y})[\mathbf{D}, \mathbf{D}] \leq \left\langle \mathbf{D}^2, \mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) \right\rangle$.

Proof. While the result is evident from the development in [421], it is not stated there formally. We therefore derive it here using our notation and one key lemma from [421]. First, note that

$$\langle \mathbf{D}, \nabla \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}) \rangle = \langle \mathbf{D}, \mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) \rangle = \frac{\left\langle \mathbf{D}, e^{\mathbf{Y}} \right\rangle}{\operatorname{tr} e^{\mathbf{Y}}},$$

where throughout ∇ denotes differentiation with respect to $\mathbf{Y}$ and $\mathbf{D}$ is viewed as fixed. Applying

∇ again gives,

$$\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y})[\mathbf{D},\mathbf{D}] = \left\langle \mathbf{D}, \nabla\left(\frac{\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle}{\operatorname{tr} e^{\mathbf{Y}}}\right)\right\rangle = \frac{\left\langle \mathbf{D}, \nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle\right\rangle}{\operatorname{tr} e^{\mathbf{Y}}} - \left(\frac{\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle}{\operatorname{tr} e^{\mathbf{Y}}}\right)^2 \leq \frac{\left\langle \mathbf{D}, \nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle\right\rangle}{\operatorname{tr} e^{\mathbf{Y}}}.$$

Note that $\nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle \neq \mathbf{D}e^{\mathbf{Y}}$ when $\mathbf{D}$ and $\mathbf{Y}$ do not commute. However, using the Taylor series for the exponential and the formula $\nabla\left\langle \mathbf{D}, \mathbf{Y}^k\right\rangle = \sum_{i=0}^{k-1} \mathbf{Y}^i\mathbf{D}\mathbf{Y}^{k-1-i}$ gives,

$$\nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle = \sum_{k=0}^{\infty}\frac{1}{k!}\nabla\left\langle \mathbf{D}, \mathbf{Y}^k\right\rangle = \sum_{k=1}^{\infty}\sum_{i=0}^{k-1}\frac{1}{k!}\mathbf{Y}^i\mathbf{D}\mathbf{Y}^{k-1-i}.$$

Consequently, we may write

$$\left\langle \mathbf{D}, \nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle\right\rangle = \sum_{k=1}^{\infty}\sum_{i=0}^{k-1}\frac{1}{k!}\left\langle \mathbf{D}, \mathbf{Y}^i\mathbf{D}\mathbf{Y}^{k-1-i}\right\rangle = \sum_{k=1}^{\infty}\sum_{i=0}^{k-1}\frac{1}{2(k!)}\left\langle \mathbf{D}, \mathbf{Y}^i\mathbf{D}\mathbf{Y}^{k-1-i} + \mathbf{Y}^{k-1-i}\mathbf{D}\mathbf{Y}^i\right\rangle.$$

Lemma 1 in [420] shows that, when $\mathbf{Y} \succeq 0$,

$$\left\langle \mathbf{D}, \mathbf{Y}^i\mathbf{D}\mathbf{Y}^{k-1-i} + \mathbf{Y}^{k-1-i}\mathbf{D}\mathbf{Y}^i\right\rangle \leq 2\left\langle \mathbf{D}^2, \mathbf{Y}^{k-1}\right\rangle.$$

Substituting back, this gives

$$\left\langle \mathbf{D}, \nabla\left\langle \mathbf{D}, e^{\mathbf{Y}}\right\rangle\right\rangle \leq \sum_{k=1}^{\infty}\frac{1}{(k-1)!}\left\langle \mathbf{D}^2, \mathbf{Y}^{k-1}\right\rangle = \left\langle \mathbf{D}^2, e^{\mathbf{Y}}\right\rangle,$$

and consequently

$$\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y})[\mathbf{D},\mathbf{D}] \leq \frac{\left\langle \mathbf{D}^2, e^{\mathbf{Y}}\right\rangle}{\operatorname{tr} e^{\mathbf{Y}}} = \left\langle \mathbf{D}^2, \mathsf{P}^{\mathrm{mw}}(\mathbf{Y})\right\rangle = \left\langle \mathbf{D}^2, \nabla\mathsf{p}^{\mathrm{mw}}(\mathbf{Y})\right\rangle$$

as required. Finally, note that the assumption $\mathbf{Y} \succeq 0$ is without loss of generality, as $\mathsf{P}^{\mathrm{mw}}(\mathbf{Y}) = \mathsf{P}^{\mathrm{mw}}(\mathbf{Y} + cI)$ for every $c \in \mathbb{R}$, and therefore $\nabla^2\mathsf{p}^{\mathrm{mw}}$ is also invariant to scalar shifts. □

F.1.4 Proof of Lemma 7.2.2

For any $\mathbf{Y}, \mathbf{D} \in S_n$, orthogonal eigenbasis $\mathbf{Q}$ for Y, and $w \sim \mathrm{Dirichlet}(\frac{1}{2}, \ldots, \frac{1}{2})$,

$$\nabla^2\bar{\mathsf{p}}(\mathbf{Y})[\mathbf{D},\mathbf{D}] \leq 3 \cdot \mathbb{E}_w \nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y} + \mathbf{Q}\operatorname{diag}(\log w)\mathbf{Q}^T)[\mathbf{D},\mathbf{D}] \tag{7.24}$$

$$\leq 3\left\langle \mathbf{D}^2, \bar{\mathsf{P}}(\mathbf{Y})\right\rangle. \tag{7.25}$$

Proof. Let $\tilde{\mathbf{D}} = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$, where as before $\mathbf{Y} = \mathbf{Q}\Lambda\mathbf{Q}^T$ is an eigen-decomposition and $\Lambda = \operatorname{diag}(\lambda)$. Recall that $\operatorname{lse} : \mathbb{R}^n \to \mathbb{R}$ denotes the vector softmax function, $\operatorname{lse}(y) := \log(\sum_{i=1}^n e^{y_i}) = \mathsf{p}^{\mathrm{mw}}(\operatorname{diag} y)$. Similarly, define $\overline{\operatorname{lse}}(y) := \mathbb{E}_w \operatorname{lse}(y + \log w)$ for $w \sim \text{Dirichlet}(\frac{1}{2}, \ldots, \frac{1}{2})$. By Lemma 87, $\bar{\mathsf{p}}(\mathbf{Y}) = \overline{\operatorname{lse}}(\lambda)$ is a spectral function. Theorem 3.3 of [357] prove that

$$\nabla^2 \bar{\mathsf{p}}(\mathbf{Y})[\mathbf{D}, \mathbf{D}] = \nabla^2 \overline{\operatorname{lse}}(\lambda)[\operatorname{diag} \tilde{\mathbf{D}}, \operatorname{diag} \tilde{\mathbf{D}}] + \left\langle \bar{\mathbf{A}}(\lambda), \tilde{\mathbf{D}} \circ \tilde{\mathbf{D}} \right\rangle, \tag{F.5}$$

where $\circ$ denotes elementwise multiplication, $\operatorname{diag}(\tilde{\mathbf{D}})$ is a vector comprised of the diagonal of $\tilde{\mathbf{D}}$, and the matrix $\bar{\mathbf{A}}$ is given by

$$\bar{\mathbf{A}}_{ij}(\lambda) = \frac{\nabla_i \overline{\operatorname{lse}}(\lambda) - \nabla_j \overline{\operatorname{lse}}(\lambda)}{\lambda_i - \lambda_j} = \mathbb{E}_w \underbrace{\frac{\nabla_i \operatorname{lse}(\lambda + \log w) - \nabla_j \operatorname{lse}(\lambda + \log w)}{\lambda_i - \lambda_j}}_{:= \mathbf{A}^w_{ij}(\lambda)}$$

for $i \neq j$ and 0 otherwise, whenever λ has distinct elements. This distinctiveness assumption is without loss of generality, as $\bar{\mathsf{p}}$ is $\mathcal{C}^2$ (Theorem 4.2, [357]) so we may otherwise consider an arbitrarily small perturbation of λ and appeal to continuity of $\nabla^2 \bar{\mathsf{p}}$.

We now use the spectral function Hessian formula to write down $\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})[\mathbf{D}, \mathbf{D}]$ where $\mathbf{Y}_{\{w\}} := \mathbf{Y} + Q \operatorname{diag}(\log w) Q^T$ (noting that $\mathbf{Y}$ and $\mathbf{Y}_{\{w\}}$ have the same eigenvectors),

$$\nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})[\mathbf{D}, \mathbf{D}] = \nabla^2 \operatorname{lse}(\lambda + \log w)[\operatorname{diag} \tilde{\mathbf{D}}, \operatorname{diag} \tilde{\mathbf{D}}] + \left\langle A^{\mathrm{mw}}(\lambda + \log w), \tilde{\mathbf{D}} \circ \tilde{\mathbf{D}} \right\rangle, \tag{F.6}$$

where

$$A^{\mathrm{mw}}_{ij}(\lambda) := \frac{\nabla_i \operatorname{lse}(\lambda) - \nabla_j \operatorname{lse}(\lambda)}{\lambda_i - \lambda_j} = A^{\mathbf{1}}_{ij}(\lambda)$$

for $i \neq j$ and 0 otherwise. Taking the expectation over w in (F.6) and recalling the definition $\overline{\operatorname{lse}}(\lambda) = \mathbb{E}_w \operatorname{lse}(\lambda + \log w)$ gives

$$\mathbb{E}_w \nabla^2 \mathsf{p}^{\mathrm{mw}}(\mathbf{Y}_{\{w\}})[\mathbf{D}, \mathbf{D}] = \nabla^2 \overline{\operatorname{lse}}(\lambda)[\operatorname{diag} \tilde{\mathbf{D}}, \operatorname{diag} \tilde{\mathbf{D}}] + \left\langle \mathbb{E}_w A^{\mathrm{mw}}(\lambda + \log w), \tilde{\mathbf{D}} \circ \tilde{\mathbf{D}} \right\rangle. \tag{F.7}$$

Comparing Eq. (F.7) to (F.5) and the desired bound (7.24), we see that it remains to upper bound $\bar{\mathbf{A}}(\lambda) = \mathbb{E}_w A^w(\lambda)$ in terms of $\mathbb{E}_w A^{\mathrm{mw}}(\lambda + \log w)$. Fix indices $i, j \in [n]$ such that $i \neq j$, and let

$$\delta := \frac{\lambda_i - \lambda_j}{2} \quad \text{and} \quad \rho := \frac{1}{2} \log \frac{w_i}{w_j}.$$

Since $\bar{\mathbf{A}}$ and A^{mw} are both symmetric matrices, we may assume that $\lambda_i > \lambda_j$ and so $\delta > 0$ (recall we assumed $\lambda_i \neq \lambda_j$ without loss of generality). Let $w^{i \leftrightarrow j}$ denote a vector identical to w except coordinates i and j are swapped. With this notation, Lemma 283, which we prove in Section F.1.4,

yields the bound

$$A_{ij}^{w}(\lambda)+A_{ij}^{w^{i\leftrightarrow j}}(\lambda)\leq\left(1+\frac{|\rho|\tanh(\delta)}{\delta}\right)\left[A_{ij}^{\mathrm{mw}}(\lambda+\log w)+A_{ij}^{\mathrm{mw}}(\lambda+\log w^{i\leftrightarrow j})\right].$$

Taking the expectation over w and using the fact that Dirichlet$(\frac{1}{2},\ldots,\frac{1}{2})$ is invariant to permutations, we have

$$\bar{\mathbf{A}}_{ij}(\lambda)\leq\mathbb{E}_{w}\left[\left(1+\frac{|\rho|\tanh(\delta)}{\delta}\right)A_{ij}^{\mathrm{mw}}(\lambda+\log w)\right]. \tag{F.8}$$

We now focus on the term $\mathbb{E}_{w}\frac{|\rho|\tanh(\delta)}{\delta}A_{ij}^{\mathrm{mw}}(\lambda+\log w)$. We have

$$\begin{aligned}\mathbb{E}_{w}\frac{|\rho|\tanh(\delta)}{\delta}A_{ij}^{\mathrm{mw}}(\lambda+\log w)&=\mathbb{E}_{w}\frac{|\rho|\tanh(\delta)}{\delta}A_{ij}^{\mathrm{mw}}(\lambda+\log w)\left[\mathbb{I}_{\{|\rho|\leq\delta\}}+\mathbb{I}_{\{|\rho|>\delta\}}\right]\\&\leq(\tanh\delta)\mathbb{E}_{w}A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{|\rho|\leq\delta\}}+\frac{\tanh\delta}{\delta}\mathbb{E}_{w}|\rho|A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{|\rho|>\delta\}},\end{aligned} \tag{F.9}$$

where the final transition uses $|\rho|\mathbb{I}_{\{|\rho|\leq\delta\}}\leq\delta\mathbb{I}_{\{|\rho|\leq\delta\}}$ and $A_{ij}^{\mathrm{mw}}(\zeta)\geq 0$ for every $\zeta\in\mathbb{R}^{n}$. The latter is a consequence of the convexity of lse and is also evident from Eq. (F.14) in Section F.1.4.

Since $w\sim$ Dirichlet$(\frac{1}{2},\ldots,\frac{1}{2})$, $\rho=\frac{1}{2}\log\frac{w_i}{w_j}$ is independent of $w_{\setminus ij}:=\{w_k\}_{k\neq i,j}$. Moreover, w_i,w_j are completely determined by ρ and $w_{\setminus ij}$ (see explicit expression in Section F.1.4). Therefore, conditional on $w_{\setminus ij}$, $A_{ij}^{\mathrm{mw}}(\lambda+\log w)$ is a function of ρ. In Lemma 284 we prove that for every λ and $w_{\setminus ij}$, this function is decreasing in ρ for $\rho>\delta$. Hence, conditionally on $w_{\setminus ij}$ and the event $\rho>\delta$, the random variables $|\rho|$ and $A_{ij}^{\mathrm{mw}}(\lambda+\log w)$ are *negatively correlated*: the expectation of their product at most the product of their expectations. Let $\mathbb{E}_{\rho}$ denote expectation conditional on $w_{\setminus ij}$. Lemma 285, with $f(\rho)=|\rho|$, $g(\rho)=A_{ij}^{\mathrm{mw}}(\lambda+\log w)$, and $\mathcal{S}=\{\rho\mid\rho>\delta\}$ gives that

$$\mathbb{E}_{\rho}|\rho|A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{\rho>\delta\}}\leq\left(\mathbb{E}_{\rho}\left[\,|\rho|\mid\rho>\delta\right]\right)\left(\mathbb{E}_{\rho}A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{\rho>\delta\}}\right). \tag{F.10}$$

Similarly, Lemma 284 also gives that (conditional on $w_{\setminus ij}$) $A_{ij}^{\mathrm{mw}}(\lambda+\log w)$ is increasing in ρ for $\rho<-\delta$, and therefore, by Lemma 285,

$$\mathbb{E}_{\rho}|\rho|A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{\rho<-\delta\}}\leq\left(\mathbb{E}_{\rho}\left[\,|\rho|\mid\rho<-\delta\right]\right)\left(\mathbb{E}_{\rho}A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{\rho<-\delta\}}\right). \tag{F.11}$$

Let $z\sim\mathrm{Beta}(\frac{1}{2},\frac{1}{2})$. The random variable $\rho=\frac{1}{2}\log\frac{w_i}{w_j}$ is symmetric and distributed as $\frac{1}{2}\log(\frac{1-z}{z})$. Therefore

$$\mathbb{E}_{\rho}\left[\,|\rho|\mid\rho<-\delta\right]=\mathbb{E}_{\rho}\left[\,|\rho|\mid\rho>\delta\right]=\frac{1}{2}\mathbb{E}\left[\log\tfrac{1-z}{z}\mid\log\tfrac{1-z}{z}>2\delta\right]\overset{(\star)}{\leq}\delta+\sqrt{1+e^{-2\delta}},$$

where we prove the inequality $(\star)$ in Lemma 288. Substituting this bound into inequalities (F.10)

and (F.11) and summing them, we obtain

$$\mathbb{E}_\rho|\rho|A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{|\rho|>\delta\}} \le \left(\delta+\sqrt{1+e^{-2\delta}}\right)\mathbb{E}_\rho A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{|\rho|>\delta\}}.$$

Taking expectation over $w_{\setminus ij}$ and substituting back into (F.9) therefore gives,

$$\mathbb{E}_w\frac{|\rho|\tanh(\delta)}{\delta}A_{ij}^{\mathrm{mw}}(\lambda+\log w) \le \left(\tanh(\delta)+\sqrt{1+e^{-2\delta}}\cdot\frac{\tanh(\delta)}{\delta}\right)\mathbb{E}_w A_{ij}^{\mathrm{mw}}(\lambda+\log w),$$

where we used again $A_{ij}^{\mathrm{mw}}(\cdot)\ge 0$ in order to increase the multiplier of $\mathbb{E}_w A_{ij}^{\mathrm{mw}}(\lambda+\log w)\mathbb{I}_{\{|\rho|\le\delta\}}$. Computation shows that $\tanh(\delta)+\sqrt{1+e^{-2\delta}}\cdot\frac{\tanh(\delta)}{\delta}\le 1.58\le 2$ for every $\delta\ge 0$. Therefore, by the bound (F.8) we have

$$\bar{\mathbf{A}}_{ij}(\lambda) \le 3\cdot\mathbb{E}_w\mathbf{A}_{ij}(\lambda+\log w). \tag{F.12}$$

Returning to (F.5), we write

$$\begin{aligned}\nabla^2\bar{\mathsf{p}}(\mathbf{Y})[\mathbf{D},\mathbf{D}] &\le \nabla^2\overline{\mathrm{lse}}(\lambda)[\operatorname{diag}\tilde{\mathbf{D}},\operatorname{diag}\tilde{\mathbf{D}}] + 3\left\langle\mathbb{E}_w A^{\mathrm{mw}}(\lambda+\log w),\tilde{\mathbf{D}}\circ\tilde{\mathbf{D}}\right\rangle\\ &\le 3\left[\nabla^2\overline{\mathrm{lse}}(\lambda)[\operatorname{diag}\tilde{\mathbf{D}},\operatorname{diag}\tilde{\mathbf{D}}] + \left\langle\mathbb{E}_w A^{\mathrm{mw}}(\lambda+\log w),\tilde{\mathbf{D}}\circ\tilde{\mathbf{D}}\right\rangle\right].\end{aligned}$$

In the first inequality above, we substituted the bound (F.12), using the fact that all the entries of $\tilde{\mathbf{D}}\circ\tilde{\mathbf{D}}$ are nonnegative. In the second inequality, we used that fact that $\nabla^2\overline{\mathrm{lse}}(\lambda)[\operatorname{diag}\tilde{\mathbf{D}},\operatorname{diag}\tilde{\mathbf{D}}]\ge 0$ since $\overline{\mathrm{lse}}$ is convex. Recalling the expression (F.7) gives (7.24). The final bound (7.25) follows from applying Lemma 7.2.2 to the right side of (7.24) and using the identity (7.21). □

A pointwise bound for Lemma 7.2.2

In this section we prove an elementary inequality that plays a central role in the proof of Lemma 7.2.2. Let $i,j\in[n]$ be such that $i\ne j$. For $\lambda\in\mathbb{R}^n$, we define

$$N_{ij}(\lambda) := \nabla_i\mathrm{lse}(\lambda)-\nabla_j\mathrm{lse}(\lambda) = \frac{e^{\lambda_i}-e^{\lambda_j}}{\sum_{k=1}^n e^{\lambda_k}} = \frac{\sinh\left(\frac{\lambda_i-\lambda_j}{2}\right)}{\cosh\left(\frac{\lambda_i-\lambda_j}{2}\right)+\frac{1}{2}\sum_{k\ne i,j}e^{\lambda_k-\frac{\lambda_i+\lambda_j}{2}}} \tag{F.13}$$

and

$$A_{ij}^{\mathrm{mw}}(\lambda)=\frac{N_{ij}(\lambda)}{\lambda_i-\lambda_j}\quad\text{and}\quad A_{ij}^{w}(\lambda)=\frac{N_{ij}(\lambda+\log w)}{\lambda_i-\lambda_j}. \tag{F.14}$$

Additionally, for any vector $w\in\mathbb{R}^n$, let $w^{i\leftrightarrow j}$ denote a vector identical to w except coordinates i and j are swapped. With this notation in hand, we state and prove our bound.

Lemma 283. *Let $\lambda \in \mathbb{R}^n$, $w \in \mathbb{R}^n_+$ and $i, j \in [n]$, $i \neq j$. Set $\delta = \frac{\lambda_i - \lambda_j}{2}$ and $\rho = \frac{1}{2}\log\frac{w_i}{w_j}$. Then,*

$$A^w_{ij}(\lambda) + A^{w^{i\leftrightarrow j}}_{ij}(\lambda) \leq \left(1 + \frac{|\rho|\tanh(\delta)}{\delta}\right)\left[A^{\mathrm{mw}}_{ij}(\lambda + \log w) + A^{\mathrm{mw}}_{ij}(\lambda + \log w^{i\leftrightarrow j})\right].$$

Proof. Define

$$r = \frac{1}{2}\sum_{k\notin\{i,j\}} e^{\lambda_k + \log w_k - \frac{\lambda_i + \log w_i + \lambda_j + \log w_j}{2}} \geq 0.$$

Observe that if we swap w_i and w_j, δ and r remain unchanged and the sign of ρ reverses. For $x \in \mathbb{R}$, let $f(x) := \frac{\sinh(x)}{\cosh(x)+r}$. Using (F.13), we may write

$$q_1 := 2A^w_{ij}(\lambda) + 2A^{w^{i\leftrightarrow j}}_{ij}(\lambda) = \frac{f(\delta+\rho)}{\delta} + \frac{f(\delta-\rho)}{\delta}$$

and

$$q_2 := 2A^{\mathrm{mw}}_{ij}(\lambda + \log w) + 2A^{\mathrm{mw}}_{ij}(\lambda + \log w^{i\leftrightarrow j}) = \frac{f(\delta+\rho)}{\delta+\rho} + \frac{f(\delta-\rho)}{\delta-\rho}.$$

With these definitions, our goal is to prove that $\frac{q_1-q_2}{q_2} \leq \frac{|\rho|\tanh(\delta)}{\delta}$. Since $f(x)$ is an odd function of x, the terms q_1 and q_2 are invariant to sign flips in either δ or ρ. Therefore, we may assume both

$$\delta \geq 0 \text{ and } \rho \geq 0$$

without loss of generality.

Substituting back the expressions for q_1, q_2 and using that $|\rho| = \rho$ by assumption yields

$$\frac{q_1 - q_2}{q_2} = \frac{\rho}{\delta}\cdot\frac{\frac{f(\delta+\rho)}{\delta+\rho} - \frac{f(\delta-\rho)}{\delta-\rho}}{\frac{f(\delta+\rho)}{\delta+\rho} + \frac{f(\delta-\rho)}{\delta-\rho}} = \frac{\rho}{\delta}\cdot\frac{g(\delta+\rho) - g(\delta-\rho)}{g(\delta+\rho) + g(\delta-\rho)}, \tag{F.15}$$

where

$$g(x) := \frac{f(x)}{x} = \frac{\tanh(x)}{x}\cdot\frac{\cosh(x)}{\cosh(x)+r}.$$

Note that $\frac{\tanh(x)}{x}$ is decreasing in $|x|$. Since $|\delta - \rho| \leq |\delta + \rho|$ by the assumption $\rho, \delta \geq 0$, we have

$$g(\delta-\rho) \geq \frac{\tanh(\delta+\rho)}{\delta+\rho}\cdot\frac{\cosh(\delta-\rho)}{\cosh(\delta-\rho)+r}.$$

and therefore

$$g(\delta+\rho) - g(\delta-\rho) \leq \frac{\tanh(\delta+\rho)}{\delta+\rho}\left(\frac{\cosh(\delta+\rho)}{\cosh(\delta+\rho)+r} - \frac{\cosh(\delta-\rho)}{\cosh(\delta-\rho)+r}\right)$$

and similarly,

$$g(\delta+\rho)+g(\delta-\rho) \geq \frac{\tanh(\delta+\rho)}{\delta+\rho}\left(\frac{\cosh(\delta+\rho)}{\cosh(\delta+\rho)+r}+\frac{\cosh(\delta-\rho)}{\cosh(\delta-\rho)+r}\right).$$

As $g(x)>0$ for every x, we may divide these bounds and obtain via elementary manipulation,

$$\begin{aligned}\frac{g(\delta+\rho)-g(\delta-\rho)}{g(\delta+\rho)+g(\delta-\rho)} &\leq \frac{\frac{\cosh(\delta+\rho)}{\cosh(\delta+\rho)+r}-\frac{\cosh(\delta-\rho)}{\cosh(\delta-\rho)+r}}{\frac{\cosh(\delta+\rho)}{\cosh(\delta+\rho)+r}+\frac{\cosh(\delta-\rho)}{\cosh(\delta-\rho)+r}} \\ &= \frac{r\left[\cosh\left(\delta+\rho\right)-\cosh\left(\delta-\rho\right)\right]}{2\cosh\left(\delta+\rho\right)\cosh\left(\delta-\rho\right)+r\left[\cosh\left(\delta+\rho\right)+\cosh\left(\delta-\rho\right)\right]} \\ &\leq \frac{\cosh(\delta+\rho)-\cosh(\delta-\rho)}{\cosh(\delta+\rho)+\cosh(\delta-\rho)} = \tanh(\rho)\tanh(\delta) \leq \tanh(\delta).\end{aligned}$$

Substituting back into (F.15) establishes the desired bound. Examining the proof, we see that the bound is tight for large values of r and $|\rho|$. □

Piecewise monotonicity of A^{mw}

Lemma 284. *Let $\lambda \in \mathbb{R}^n$, $w \in \sigma_n$ (the simplex in $\mathbb{R}^n$), and $i,j \in [n]$ such that $\delta := \frac{1}{2}(\lambda_i - \lambda_j) > 0$, and set $\rho := \frac{1}{2}\log\frac{w_i}{w_j}$. When λ and $\{w_k\}_{k\neq i,j}$ are held fixed, $A^{\mathrm{mw}}_{ij}(\lambda+\log w)$ is increasing in ρ for $\rho < -\delta$, and decreasing in ρ for $\rho > \delta$.*

Proof. First, we write $A^{\mathrm{mw}}_{ij}(\lambda+\log w)$ explicitly as a function of ρ, with λ and $\{w_k\}_{k\neq i,j}$ as fixed parameters. By (F.14) we have

$$A^{\mathrm{mw}}_{ij}(\lambda+\log w) = \frac{\sinh(\rho+\delta)}{2(\rho+\delta)\left[\cosh(\rho+\delta)+\frac{1}{2}\sum_{k\notin\{i,j\}}\frac{w_k}{\sqrt{w_i w_j}}e^{\lambda_k-\frac{\lambda_i+\lambda_j}{2}}\right]}.$$

Let $m = w_i + w_j = 1 - \sum_{k\neq i,j} w_k$. Since $\frac{w_i}{w_j} = e^{2\rho}$ and $w \in \sigma_n$, we have that $w_i = \frac{m}{1+e^{-2\rho}}$ and $w_j = \frac{m}{1+e^{2\rho}}$. Therefore,

$$\frac{1}{\sqrt{w_i w_j}} = \frac{1}{m}\sqrt{(1+e^{-2\rho})(1+e^{2\rho})} = \frac{2}{m}\cosh(\rho).$$

Thus,

$$A^{\mathrm{mw}}_{ij}(\lambda+\log w) = \frac{\sinh(\rho+\delta)}{2(\rho+\delta)\left[\cosh(\rho+\delta)+r_0\cosh(\rho)\right]},$$

where $r_0 = \sum_{k\notin\{i,j\}}\frac{w_k}{m}e^{\lambda_k-\frac{\lambda_i+\lambda_j}{2}}$ is a function of only λ and $\{w_k\}_{k\neq i,j}$, and therefore $A^{\mathrm{mw}}_{ij}(\lambda+\log w)$ can be viewed as a function of ρ as claimed.

Writing $x = \rho + \delta$, showing the desired monotonicity properties is equivalent to showing that

$$b(x) := \frac{\sinh(x)}{x\left(\cosh(x) + r_0\cosh(x-\delta)\right)}$$

is decreasing for $x > 2\delta$ and increasing for $x < 0$. The derivative of $b(x)$ is

$$b'(x) = \frac{\cosh(x) - \frac{1}{x}\sinh(x)}{x\left(\cosh(x) + r_0\cosh(x-\delta)\right)} - \frac{\sinh(x)\left[\sinh(x) + r_0\sinh(x-\delta)\right]}{x\left[\cosh(x) + r_0\cosh(x-\delta)\right]^2},$$

and has, for all $x \in \mathbb{R}$, the same sign as

$$s := \frac{x\left[\cosh(x) + r_0\cosh(x-\delta)\right]}{\sinh(x)} b'(x) = \coth(x) - \frac{1}{x} - \frac{\sinh(x) + r_0\sinh(x-\delta)}{\cosh(x) + r_0\cosh(x-\delta)}. \tag{F.16}$$

For $x > 2\delta$, we have by Dan's favorite inequality ($\frac{a_1+a_2}{b_1+b_2} \geq \min\{\frac{a_1}{b_1}, \frac{a_2}{b_2}\}$ for all $a_1, a_2, b_1, b_2 \geq 0$),

$$\frac{\sinh(x) + r_0\sinh(x-\delta)}{\cosh(x) + r_0\cosh(x-\delta)} \geq \min\left\{\tanh(x), \tanh(x-\delta)\right\} = \tanh(x-\delta) > \tanh(x/2),$$

where in the last transition we used the fact that $x > 2\delta$ implies $x - \delta > x/2$. Therefore, for $x > 2\delta$ we have the following bound for s,

$$s \leq \coth(x) - \frac{1}{x} - \tanh(x/2) = \frac{1}{\sinh(x)} - \frac{1}{x} < 0,$$

so we have that $b(x)$ is decreasing for $x > 2\delta$ as required, since s has the same sign as $b'(x)$.

Similarly, for $x < 0$, we have by Dan's favorite inequality,

$$\frac{-\sinh(x) - r_0\sinh(x-\delta)}{\cosh(x) + r_0\cosh(x-\delta)} \geq \min\left\{-\tanh(x), -\tanh(x-\delta)\right\} = -\tanh(x).$$

Therefore, for $x < 0$ we have

$$s \geq \coth(x) - \frac{1}{x} - \tanh(x) = \frac{1}{-x} - \frac{2}{\sinh(-2x)} > 0,$$

which shows that $b(x)$ is increasing for $x < 0$, concluding the proof. □

The following Lemma proves the intuitive fact that decreasing and increasing functions of the same random variable are negatively correlated.

Lemma 285. *Let ρ be a real-valued random variable, let f, g be functions from $\mathbb{R}$ to $\mathbb{R}$ and let $\mathcal{S} \subset \mathbb{R}$ be an interval. If $f(x)$ is non-decreasing in x for $x \in \mathcal{S}$ and $g(x)$ is non-increasing in x for $x \in \mathcal{S}$, then*

$$\mathbb{E} f(\rho)g(\rho)\mathbb{I}_{\{\rho\in\mathcal{S}\}} \leq \left(\mathbb{E}\left[f(\rho) \mid \rho \in \mathcal{S}\right]\right) \cdot \left(\mathbb{E} g(\rho)\mathbb{I}_{\{\rho\in\mathcal{S}\}}\right).$$

Proof. For every $x, x' \in \mathcal{S}$ we have $(f(x) - f(x')) \cdot (g(x) - g(x')) \leq 0$. Hence, for every $x, x' \in \mathbb{R}$, the bound $(f(x) - f(x')) \cdot (g(x) - g(x')) \cdot \mathbb{I}_{\{x \in \mathcal{S}\}} \mathbb{I}_{\{x' \in \mathcal{S}\}} \leq 0$ holds as well. Let ρ' be an independent copy of ρ, then

$$\mathbb{E}\left[(f(\rho) - f(\rho')) \cdot (g(\rho) - g(\rho')) \cdot \mathbb{I}_{\{\rho \in \mathcal{S}\}} \mathbb{I}_{\{\rho' \in \mathcal{S}\}}\right] \leq 0.$$

Rearranging and using the fact that ρ, ρ' are i.i.d., we have

$$\left(\mathbb{E} f(\rho) g(\rho) \mathbb{I}_{\{\rho \in \mathcal{S}\}}\right) \cdot \left(\mathbb{E}\mathbb{I}_{\{\rho' \in \mathcal{S}\}}\right) \leq \left(\mathbb{E}\left[f(\rho)\mathbb{I}_{\{\rho \in \mathcal{S}\}}\right]\right) \cdot \left(\mathbb{E}\left[g(\rho')\mathbb{I}_{\{\rho' \in \mathcal{S}\}}\right]\right).$$

Dividing by $\mathbb{E}\mathbb{I}_{\{\rho' \in \mathcal{S}\}} = \mathbb{P}(\rho \in \mathcal{S})$ yields the desired bound. □

F.1.5 Facts about the Beta distribution

Here we collect properties of Beta-distributed random variables, which we use in our development.

Lemma 286. *Let $n \in \mathbb{N}$ and let $z \sim \mathrm{Beta}(\frac{1}{2}, \frac{n-1}{2})$. Then*

$$\mathbb{E} \log \tfrac{1}{z} = \psi\left(\tfrac{n}{2}\right) - \psi\left(\tfrac{1}{2}\right) \leq \log(n) + \log(2) + \gamma \leq \log(4n),$$

where $\psi(x) = \frac{d}{dx} \log \Gamma(x)$ is the digamma function, and γ is the Euler-Mascheroni constant.

Proof. $\mathbb{E} \log \frac{1}{z} = \psi\left(\frac{n}{2}\right) - \psi\left(\frac{1}{2}\right)$ by the well-known formula for expectation of the logarithm of a Beta random variable. We have $\psi(x) \leq \log(x)$ and $\psi(\frac{1}{2}) = -\log(4) - \gamma$. Moreover, $\gamma \leq \log 2$, giving the final bound. □

Lemma 287. *Let $z \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{1}{2}\right)$ and $\ell \geq 0$. Then*

$$\frac{2}{\pi} \frac{e^{-\ell/2}}{\sqrt{1 + e^{-\ell}}} \leq \mathbb{P}\left(\log \frac{1-z}{z} \geq \ell\right) \leq \frac{2}{\pi} e^{-\ell/2}.$$

Proof. The distribution $\mathrm{Beta}\left(\frac{1}{2}, \frac{1}{2}\right)$ has density $\frac{1}{\pi} x^{-1/2} (1-x)^{-1/2}$. Therefore

$$\mathbb{P}\left(\log \frac{1-z}{z} \geq \ell\right) = \mathbb{P}\left(z \leq \frac{1}{1+e^{\ell}}\right) = \frac{1}{\pi} \int_0^{\left(1+e^{\ell}\right)^{-1}} x^{-1/2} (1-x)^{-1/2} \, dx.$$

To obtain a lower bound, we use $(1-x)^{-1/2} \geq 1$ for every $x \in [0,1]$, and therefore,

$$\mathbb{P}\left(\log \frac{1-z}{z} \geq \ell\right) \geq \frac{1}{\pi} \int_0^{\left(1+e^{\ell}\right)^{-1}} x^{-1/2} dx = \frac{2}{\pi\sqrt{1+e^{\ell}}} = \frac{2}{\pi} \frac{e^{-\ell/2}}{\sqrt{1+e^{-\ell}}}.$$

For the upper bound, we use $(1-x)^{-1/2} \le \left(1-\frac{1}{1+e^\ell}\right)^{-1/2}$ for every $0 \le x \le (1+e^\ell)^{-1}$, giving

$$\mathbb{P}\left(\log\frac{1-z}{z} \ge \ell\right) \le \frac{1}{\pi}\sqrt{\frac{1+e^\ell}{e^\ell}}\int_0^{\left(1+e^\ell\right)^{-1}} x^{-1/2}dx = \frac{2}{\pi}e^{-\ell/2}.$$

□

Lemma 288. *Let $z \sim \mathrm{Beta}\left(\frac{1}{2},\frac{1}{2}\right)$ and $\ell \ge 0$. Then*

$$\mathbb{E}\left[\log\frac{1-z}{z} \middle| \log\frac{1-z}{z} \ge \ell\right] \le \ell + 2\sqrt{1+e^{-\ell}}.$$

Proof. Conditional on $\log\frac{1-z}{z} \ge \ell$, $\log\frac{1-z}{z}$ is a nonnegative random variable, and we may therefore write

$$\begin{aligned}\mathbb{E}\left[\log\frac{1-z}{z} \middle| \log\frac{1-z}{z} \ge \ell\right] &= \int_{x=0}^{\infty}\mathbb{P}\left(\log\frac{1-z}{z} \ge x \,\middle|\, \log\frac{1-z}{z} \ge \ell\right)dx \\ &= \ell + \int_{x=\ell}^{\infty}\frac{\mathbb{P}\left(\log\frac{1-z}{z} \ge x\right)}{\mathbb{P}\left(\log\frac{1-z}{z} \ge \ell\right)}dx.\end{aligned}$$

By Lemma 287,

$$\frac{\mathbb{P}\left(\log\frac{1-z}{z} \ge x\right)}{\mathbb{P}\left(\log\frac{1-z}{z} \ge \ell\right)} \le \sqrt{1+e^{-\ell}}\cdot e^{-(x-\ell)/2}.$$

Integrating, we obtain the desired bound. □

Lemma 289. *Let $3 \le n \in \mathbb{N}$ and let $z \sim \mathrm{Beta}(\frac{1}{2},\frac{n-1}{2})$. For every $\delta \in (0,1)$,*

$$\mathbb{P}\left(z \ge \frac{\delta^2}{n}\right) > 1-\delta.$$

Proof. The random variable z has density

$$\frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{1}{2})\Gamma(\frac{n-1}{2})}x^{-1/2}(1-x)^{(n-3)/2} \le \sqrt{\frac{n}{2\pi x}},$$

where we used $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ and Gautschi's inequality $\Gamma(m+1)/\Gamma(m+s) \le (m+1)^{1-s}$ with $m = \frac{n}{2}-1$ and $s = \frac{1}{2}$. Integrating the upper bound on the density, we find $\mathbb{P}(z \le \delta^2/n) \le \sqrt{\frac{2}{\pi}}\delta < \delta$. □

F.1.6 Efficient computation of matrix exponential-vector products

In this section we give a more detailed discussion of matrix exponential-vector product approximation using the Lanczos method, and prove the results stated in Section 7.2.3. We first formally state the Lanczos method. In Next, we survey known approximation guarantees and derive simple

corollaries. We then show that we can apply the matrix exponential to a random vector with a multiplicative error guarantee, and then we prove it implies Proposition 22. We next discuss some possible improvement to our guarantees via modifications and alternatives to the Lanczos method. Finally, we prove Corollary 24.

Throughout this section we use $\mathrm{mv}(\mathbf{A})$ to denote the time required to multiply the matrix $\mathbf{A}$ with any vector.

F.1.7 Description of the Lanczos method

Algorithm 75: Lanczos method for computing matrix exponential vector product $\widetilde{\exp}_k(\mathbf{A}, b)$

input : $\mathbf{A} \in S_n$, number of iterations k, vector $b \in \mathbb{R}^n$

$q_0 \leftarrow 0 \in \mathbb{R}^n$, $q_1 \leftarrow b/\left\|b\right\|_2$, $\beta_1 \leftarrow 1$

for $i = 1$ *to* k **do**

  $q_{i+1} \leftarrow \mathbf{A}q_i - \beta q_{i-1}$ and $\alpha_i \leftarrow q_{i+1}^T q_i$

  $q_{i+1} \leftarrow q_{i+1} - \alpha q_i$ and $\beta_{i+1} = \left\|q_{i+1}\right\|_2$

  if $\beta_{i+1} = 0$ **then break else** $q_{i+1} \leftarrow q_{i+1}/\beta_{i+1}$

end

Let

$$Q = [q_1 \ \cdots \ q_k] \quad \text{and} \quad T = \begin{bmatrix} \alpha_1 & \beta_2 & & 0 \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ 0 & & \beta_k & \alpha_k \end{bmatrix}$$

Compute tridiagonal eigen-decomposition $T = V\Lambda V^T$

return : $\widetilde{\exp}_k(\mathbf{A}, b) = \left\|b\right\|_2 \cdot QV \exp(\Lambda) V^T e_1$

Ignoring numerical precision issues, each iteration in the for loop requires $O(\mathrm{mv}(\mathbf{A}))$ time, and that for a k-by-k tridiagonal matrix, eigen-decomposition requires $O(k^2)$ time [256], and so the total complexity is $O(\mathrm{mv}(\mathbf{A})k + k^2)$. In practical settings $k \ll n \leq \mathrm{mv}(\mathbf{A})$ and the cost of the eigen-decomposition is negligible. Nevertheless, there are ways to avoid performing it, which we discuss briefly.

Known approximation results, and some corollaries

We begin with a result on uniform polynomial approximation of the exponential due to [466].

Theorem 87 ([466], Theorem 4.1 Restated)**.** *For every $b > 0$ and every $\epsilon \in (0, 1]$ there exists*

polynomial $p : \mathbb{R} \to \mathbb{R}$ *of degree* $O(\sqrt{\max\{b, \log(1/\epsilon)\}\log(1/\epsilon)})$ *such that*

$$\sup_{x \in [0,b]} |\exp(-x) - p(x)| \leq \epsilon\,.$$

As an immediate corollary of this we obtain the following bounds for approximating $\exp(x)$ over arbitrary values

Corollary 64. *For every* $a < b \in \mathbb{R}$ *and every* $\epsilon \in (0, 1]$ *there exists polynomial* $p : \mathbb{R} \to \mathbb{R}$ *of degree* $O(\sqrt{\max\{b-a, \log(1/\epsilon)\}\log(1/\epsilon)})$ *polynomial such that*

$$\sup_{x \in [a,b]} |\exp(x) - p(x)| \leq \epsilon \exp(b)\,.$$

Proof. For all $x \in [a, b]$ we have $b - x \in [0, b - a]$ and therefore by Theorem 87 there is a degree $O(\sqrt{\max\{b-a, \log(1/\epsilon)\}\log(1/\epsilon)})$ polynomial $q : \mathbb{R} \to \mathbb{R}$ such that

$$\sup_{x \in [a,b]} |\exp(-(b-x)) - q(b-x)| \leq \epsilon\,.$$

Since $\exp(-(b-x)) = \exp(-b)\exp(x)$, the polynomial $p(x) = \exp(b)q(b-x)$ is as desired. □

The classical theory on the Lanczos method tells us that its error is bounded by twice that of any uniform polynomial approximation. However, this theory does not account for finite precision. A recent result [405] ties polynomial approximation to the error of the Lanczos method using finite bitwidth floating point operations.

Theorem 88 ([405], Theorem 1)**.** *Let* $\mathbf{A} \in S_n$*,* $u \in \mathbb{R}^n$*, and* $f : \mathbb{R} \to \mathbb{R}$*. Suppose* $k \in \mathbb{N}$*,* $\eta \in (0, \|\mathbf{A}\|_{\mathrm{op}}]$ *and a polynomial* p *for degree* $< k$ *satisfy,*

$$\sup_{x \in [\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |f(x) - p(x)| \leq \epsilon_k \quad \textit{and} \quad \sup_{x \in [\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |f(x)| \leq C.$$

For any $\mu \in (0, 1)$*, let* $y_{k,\mu}$ *be the output of* k *iterations of the Lanczos method for approximating* $f(\mathbf{A})v$*, using floating point operations with* $B \geq c\log(\frac{nk\|\mathbf{A}\|_{\mathrm{op}}}{\mu\eta})$ *bits precision (for numerical constant* $c < \infty$*). Then* $y_{k,\mu}$ *satisfies*

$$\|f(\mathbf{A})u - y_{k,\mu}\|_2 \leq (7k \cdot \epsilon_k + \mu \cdot C)\,\|u\|_2\,.$$

If arithmetic operations with B *bits of precision can be performed in* $O(1)$ *time then the method can be implemented in time* $O(\mathrm{mv}(\mathbf{A})k + kB\max\{k, B\})$*.*

Specializing to the matrix exponential and using the uniform approximation guarantee of Corollary 64, we immediately obtain the following.

Corollary 65. *Let $\mathbf{A} \in S_n$, $u \in \mathbb{R}^n$, and $\epsilon > 0$, and set $M = \max\{\|\mathbf{A}\|_{\mathrm{op}}, \log(1/\epsilon), 1\}$. There exists numerical constants $c, c' < \infty$ such that, for $k \geq c\sqrt{M \log(M/\epsilon)}$ and $B \geq c' \log(\frac{nM}{\epsilon})$, computing $y = \widetilde{\exp}_k(A, u)$ with B bits of floating point precision guarantees*

$$\|\exp(\mathbf{A})u - y\|_2 \leq \epsilon \exp(\lambda_{\max}(\mathbf{A})) \|u\|_2 .$$

The computation takes time

$$O\left(\mathrm{mv}(\mathbf{A})\sqrt{M \log(M/\epsilon)} + M \log^2(nM/\epsilon)\right)$$

provided $\Theta(\log(\frac{nM}{\epsilon}))$ bit arithmetic operations can be performed in time $O(1)$.

Proof. Let $\eta = 1$. Using $\lambda_{\max}(\mathbf{A}) - \lambda_{\min}(\mathbf{A}) \leq 2\|\mathbf{A}\|$, Corollary 64 yields that for all $\alpha \in (0, 1]$ there exists a degree $O\left(\sqrt{\max\{1 + \|\mathbf{A}\|_{\mathrm{op}}, \log(\frac{1}{\alpha})\} \log(\frac{1}{\alpha})}\right)$ polynomial $p : \mathbb{R} \to \mathbb{R}$ such that

$$\sup_{x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]} |\exp(x) - p(x)| \leq \alpha \exp(\eta) \exp(\lambda_{\max}(\mathbf{A})) .$$

Further, since $|\exp(x)| \leq \exp(\eta)\exp(\lambda_{\max}(\mathbf{A}))$ for all $x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$, Theorem 88 with $f(x) = e^x$ and $\eta = 1$ implies that for all $\mu \in (0, 1)$, after applying Lanczos for $k = O(\sqrt{\max\{\|\mathbf{A}\|_{\mathrm{op}}, \log(1/\alpha)\} \log(1/\alpha)})$ iterations on a floating point machine with $\Theta(B)$ bits of precision for $B = \log(\frac{nk\|\mathbf{A}\|}{\mu})$ returns y with

$$\|f(\mathbf{A})u - y\|_2 \leq \left(\mu + \alpha \cdot O(\sqrt{\max\{\|\mathbf{A}\|_{\mathrm{op}}, \log(1/\alpha)\} \log(1/\alpha)})\right) \exp(\eta) \exp(\lambda_{\max}(\mathbf{A})))$$

in time $O((\mathrm{mv}(\mathbf{A}) + n)k + kB \max\{k, B\})$. Choosing, $\alpha = O(\epsilon/(M \log(M/\epsilon)))$ and $\mu = O(\epsilon)$ yields the result. □

Multiplicative approximation for random vectors

We now combine the known results cited in the previous section with the randomness of the vector fed to the matrix exponential, to obtain a multiplicative guarantee that holds with high-probability over the choice of u, but not for all $u \in \mathbb{S}^{n-1}$.

Proposition 65. *Let $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, and $\mathbf{A} \in S_n$. If u is sampled uniformly at random from the unit sphere and for $k = \Omega(\sqrt{M \log(nM/(\epsilon\delta))}) \in \mathbb{N}$ for $M = \max\{\|\mathbf{A}\|_{\mathrm{op}}, \log(n/(\epsilon\delta)), 1\}$ we let $y = \widetilde{\exp}_k(\mathbf{A}, u)$ (See Algorithm 75) then*

$$\|\exp(\mathbf{A})u - y\|_2 \leq \epsilon \|\exp(\mathbf{A})u\|_2 \text{ with probability } \geq 1 - \delta.$$

This can be implemented in time $O\left(\mathrm{mv}(\mathbf{A})\sqrt{M \log(nM/(\epsilon\delta)} + M \log^2(nM/(\epsilon\delta))\right)$ on a floating point machine with $O(\log(nM/(\epsilon\delta)))$ bits of precision where arithmetic operations take $O(1)$ time.

Proof. Consider an application of Corollary 65 to compute y such that

$$\|\exp(\mathbf{A})u - y\|_2 \leq \epsilon' \exp(\lambda_{\max}(\mathbf{A})) \|u\|_2 \ .$$

Now let v be a unit eigenvector of $\mathbf{A}$ with eigenvalue $\lambda_{\max}(\mathbf{A})$. Since v is an eigenvector or the PSD matrix $\exp(\mathbf{A})$ with eigenvalue $\exp(\lambda_{\max}(\mathbf{A}))$ we have that $\|\exp(\mathbf{A})u\| \geq \exp(\lambda_{\max}) \left|v^T u\right|$. However, since u is a random unit vector we have that $|v^T u|^2 / \|u\|_2^2 \sim \text{Beta}(\frac{1}{2}, \frac{n-1}{2})$. Lemma 289 therefore gives that $|v^T u|^2 / \|u\|_2^2 \geq \frac{\delta^2}{n}$ with probability at least $1 - \delta$. Consequently, $\exp(\lambda_{\max}(\mathbf{A})) \|u\|_2 \leq \frac{\sqrt{n}}{\delta} \|\exp(\mathbf{A})u\|_2$ with the same probability. Choosing $\epsilon' = \epsilon\delta/\sqrt{n}$ and invoking Corollary 65 yields the result. □

Proof of Proposition 22

The following lemma relates the multiplicative approximation error for matrix exponential vector products with the additive approximation error for $\mathsf{P}_u(\mathbf{Y})$ under trace norm. Combining it with Proposition 65 immediately yields Proposition 22.

Lemma 290. *Let $\mathbf{Y} \in S_n$, $u, y \in \mathbb{R}^n$ and $\epsilon \in [0, 1)$. If $y \in \mathbb{R}^n$ satisfies*

$$\|\exp(\mathbf{Y}/2)u - y\|_2 \leq \frac{\epsilon}{\sqrt{8}} \|\exp(\mathbf{Y}/2)u\|_2$$

then

$$\left\| \mathsf{P}_u(\mathbf{Y}) - \frac{yy^T}{\|y\|_2^2} \right\|_1 \leq \epsilon \ .$$

Proof. Let $z := \exp(\mathbf{Y}/2)u$ so that by assumption $\|z - y\|_2 \leq \epsilon \|z\|_2$. Further, let $\bar{z} := z/\|z\|_2$ and $\bar{y} := y/\|y\|_2$. Direct calculation (see e.g. Lemma 27 of [143]) yields that the eigenvalues of $\bar{z}\bar{z}^T - \bar{y}\bar{y}^T$ are $\pm\sqrt{1 - (\bar{z}^T\bar{y})^2} = \pm\frac{1}{2} \|\bar{z} + \bar{y}\|_2 \|\bar{z} - \bar{y}\|_2$ and therefore the definition of $\mathsf{P}_u(\mathbf{Y})$ yields

$$\left\| \mathsf{P}_u(\mathbf{Y}) - \frac{yy^T}{\|y\|_2^2} \right\|_1 = \left\| \bar{z}\bar{z}^T - \bar{y}\bar{y}^T \right\|_1 = \|\bar{z} + \bar{y}\|_2 \cdot \|\bar{z} - \bar{y}\|_2 \leq \sqrt{2} \|\bar{z} - \bar{y}\|_2 , \tag{F.17}$$

where in the last inequality we used that $\bar{z}$ and $\bar{y}$ are unit vectors. Further, by the triangle inequality and the definitions of $\bar{y}$ and $\bar{z}$ we have

$$\begin{aligned} \|\bar{z} - \bar{y}\|_2 &\leq \left\| \frac{z}{\|z\|_2} - \frac{y}{\|z\|_2} \right\|_2 + \left\| \frac{y}{\|z\|_2} - \frac{y}{\|y\|_2} \right\|_2 \\ &= \frac{\|z - y\|_2}{\|z\|_2} + \frac{|\, \|y\|_2 - \|z\|_2 \,|}{\|z\|_2} \leq 2\frac{\|z - y\|_2}{\|z\|_2} \end{aligned} \tag{F.18}$$

Combining (F.17) and (F.18) with the fact that $\|z - y\|_2 \leq (\epsilon/\sqrt{8})\|z\|_2$ then yields

$$\left\| \mathsf{P}_u(\mathbf{Y}) - \frac{yy^T}{\|y\|_2^2} \right\|_1 \leq \sqrt{2} \cdot 2 \cdot (\epsilon/\sqrt{8}) = \epsilon.$$

□

Therefore, Proposition 22 follows immediately by invoking 65 with slightly smaller ϵ.

Improvements to the Lanczos method

In this chapter we focused on the Lanczos method for approximating matrix exponential vector products because of its excellent practicality and clean analysis. However, there are several modifications to the method with appealing features, which we now describe briefly. A common theme among these modifications is the use of rational approximations to the exponential, which converge far faster than polynomial approximations [429, 466]. Consequently, it suffices to perform $\widetilde{O}(1)$ Lanczos iterations on a carefully shifted and inverted version of the matrix. Each of these iterations then involves solving a linear system, and the efficacy of the shift-invert scheme will depend on how quickly they are solved.

One basic approach to solving these systems is via standard iterative methods, e.g. conjugate gradient. We expect such approach to offer little to no advantage over applying the Lanczos approximation directly, as both methods produce vectors in the same Krylov subspace. However, the approach renders the number of Lanczos iterations k logarithmic in $\|\mathbf{A}\|_{\text{op}}$, and therefore the cost k^2 will never dominate the cost of the matrix-vector products [429, 405]

There is, however, a simpler way of avoiding the eigen-decomposition—simply use the rational approximation on the tridiagonal matrix formed by running the ordinary Lanczos method, as [465] proposes. With an appropriate rational function, computing a highly accurate approximation to $\exp(T)e_1$ requires $\widetilde{O}(1)$ tridiagonal system solves, each costing $O(k)$ time. We leave the derivation of explicit error bounds for this technique (similar to Corollary 64) to future work. In practice, the cost $O(k^2)$ of tridiagonal eigen-decomposition will often be very small compared to the cost $O(\text{mv}(\mathbf{A})k)$ of the matrix-vector products.

More significant improvements are possible if the linear system solving routine is able to exploit information beyond matrix-vector products. For example, consider the case where the matrix to be exponentiated is a sum of very sparse matrices—this will happen for our sketch whenever the $\mathbf{G}_t$ matrices are much sparser than their cumulative sum. Then, it is possible to use stochastic variance reduced optimization methods to solve the linear system, as [20] describe. Another scenario of interest is when the input matrix has a Laplacian/SDD structure and in this case the performance of specialized linear system solvers implies approximation guarantees where the polynomial dependence on $\|\mathbf{A}\|_{\text{op}}$ is removed altogether [429]. A final useful structure is a chordal sparsity pattern, which

enables efficient linear system solving through fast Cholesky decomposition.

Proof of Corollary 24

Corollary 24. *Let $\mathbf{G}_1, \dots, \mathbf{G}_T$ be symmetric gain matrices satisfying $\|\mathbf{G}_t\|_\infty \le 1$ for every t. There exists a numerical constant $k_0 < \infty$, such that for every $T \in \mathbb{N}$ and $\delta \in (0,1)$, $\tilde{\mathbf{X}}_{t;k_t}$ defined in (7.29) with $k_t = \left\lceil k_0(\sqrt{1+\eta t})\log(\frac{nT}{\delta})\right\rceil$, and $\mathbf{X}_t$ defined in (7.13) satisfy*

$$\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \tilde{\mathbf{X}}_{t;k_t}\right\rangle \ge -1 + \sum_{t=1}^{T}\langle \mathbf{G}_t, \mathbf{X}_t\rangle \quad w.p. \ge 1-\delta/2. \tag{7.30}$$

Let $\epsilon \in (0,1]$, $T = \frac{16\log(4en/\delta)}{\epsilon^2}$ and $\eta = \sqrt{\frac{2\log(4en)}{3T}}$. If Assumption 5 holds with respect to the actions $\tilde{\mathbf{X}}_{t;k_t}$, then with probability at least $1-\delta$, $\frac{1}{T}\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right) - \frac{1}{T}\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \tilde{\mathbf{X}}_{t;k_t}\right\rangle \le \epsilon$. Computing the actions $\tilde{\mathbf{X}}_{1;k_1}, \dots, \tilde{\mathbf{X}}_{T;k_T}$ requires $O(\epsilon^{-2.5}\log^{2.5}(\frac{n}{\epsilon\delta}))$ matrix-vector products.

Proof. To obtain the bound (7.30) we use Proposition 22 with $\epsilon \leftarrow \frac{1}{T}$ and $\delta \leftarrow \delta/(2T)$ (since we will use a union bound). At iteration t, $\|\mathbf{G}_i\|_\infty \le 1$ for all $i < t$, the quantity M appearing in Proposition 22 can be bounded as

$$M \le \left(1 + \left\|\frac{\eta}{2}\sum_{i=1}^{t-1}\mathbf{G}_i\right\|_\infty\right)\log\frac{nT^2}{\delta} \le O(1)(1+\eta t)\log\frac{nT}{\delta}.$$

Therefore, our choice of k_t suffices to guarantee, for $\mathbf{Y}_t = \eta\sum_{i=1}^{t-1}\mathbf{G}_i$,

$$\|\mathsf{P}_{u_t}(\mathbf{Y}_t) - \widetilde{\mathsf{P}}_{u_t;k_t}(\mathbf{Y}_t)\|_1 \le \frac{1}{T} \text{ with probability } \ge 1 - \frac{\delta}{2T},$$

and so by the union bound the inequality above holds for all $t = 1, \dots, T$ with probability at least $1-(\delta/2)$. Note that when using Proposition 22 we use the fact that u_t is independent of $\mathbf{Y}_t$. Thus, we have

$$\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \mathbf{X}_t - \tilde{\mathbf{X}}_{t;k_t}\right\rangle \le \sum_{t=1}^{T}\|\mathbf{G}_t\|_\infty\|\mathbf{X}_t - \tilde{\mathbf{X}}_{t;k_t}\|_1 \le \sum_{t=1}^{T}\|\mathsf{P}_{u_t}(\mathbf{Y}_t) - \widetilde{\mathsf{P}}_{u_t;k_t}(\mathbf{Y}_t)\|_1 = 1,$$

giving (7.30), where we have used $\|\mathbf{G}_t\|_\infty \le 1$ for every t.

Note that if Assumption 5 holds with respect to the actions $\tilde{\mathbf{X}}_{t;k_t}$ then we have $\mathbf{G}_t \perp u_t \mid \mathcal{F}_{t-1}$ and therefore $\mathbb{E}\left[\langle \mathbf{G}_t, \mathbf{X}_t\rangle \,|\mathcal{F}_{t-1}\right] = \left\langle \mathbf{G}_t, \bar{\mathbf{X}}_t\right\rangle$ so that Corollary 23 holds. Thus, to obtain the second part of the corollary, we use the bound (7.18) with $\delta \leftarrow \delta/2$ and η and T as specified; using a union bound again we have that (7.30) and (7.18) hold together with probability at least $1-\delta$. Note that

$\eta \leq \epsilon \leq 1$ and therefore $1/T \leq 1/(\eta T)$. This gives,

$$\frac{1}{T}\lambda_{\max}\left(\sum_{i=1}^{T}\mathbf{G}_t\right) - \frac{1}{T}\sum_{t=1}^{T}\left\langle \mathbf{G}_t, \tilde{\mathbf{X}}_{t;k_t}\right\rangle \leq \frac{1}{T} + \frac{3\eta}{2} + \frac{\log(4n)}{\eta T} + \sqrt{\frac{2\log\frac{2}{\delta}}{T}}$$
$$\leq \frac{3\eta}{2} + \frac{\log(4en)}{\eta T} + \sqrt{\frac{2\log\frac{2}{\delta}}{T}} = \sqrt{\frac{6\log(4en)}{T}} + \sqrt{\frac{2\log\frac{2}{\delta}}{T}} \leq \epsilon,$$

as required. Finally note that $1 + \eta T = O(\epsilon^{-1}\log(\frac{n}{\delta}))$ and consequently

$$k_T = O\left(\epsilon^{-1/2}\log^{1/2}(\tfrac{n}{\delta})\log^{1/2}(\tfrac{nT}{\delta})\right) = O\left(\epsilon^{-1/2}\log^{1.5}(\tfrac{n}{\epsilon\delta})\right).$$

Since $k_1 \leq k_2 \leq \cdots k_T$, the total number of matrix-vector products is bounded by $T \cdot k_T = O(\epsilon^{-2.5}\log^{2.5}(\frac{n}{\epsilon\delta}))$, which concludes the proof. □

F.2 Deferred proofs from Sections 7.4

F.2.1 Proof of Proposition 26

We give a proof of Proposition 26 in this section. First, we recall an algorithm for the testing variant of a pure packing SDP problem given in Section 7.5.

Proposition 66 (Restatement of Theorem 48)**.** *There is an algorithm, $\mathcal{A}_{\text{test}}$, which given matrices $\{\mathbf{M}_i\}_{i\in[n]}$ and a parameter C, is an ϵ-tolerant tester for the decision problem*

$$\text{does there exist } w \in \Delta^n \text{ such that } \sum_{i\in[n]} w_i\mathbf{M}_i \preceq C\mathbf{I}? \tag{F.19}$$

The algorithm $\mathcal{A}_{\text{test}}$ succeeds with probability $\geq 1-\delta$ and runs in time

$$O\left(\mathcal{T}_{\text{mv}}\left(\{\mathbf{M}_i\}_{i\in[n]}\right)\cdot\frac{\log^2(nd(\delta\epsilon)^{-1})\log^2 d}{\epsilon^5}\right).$$

Proof of Proposition 26. As an immediate result of Proposition 66, we can solve (7.59) to multiplicative accuracy ϵ using a binary search. This reduction is derived as Lemma A.1 of [285], but we give a brief summary here. We subdivide the range $[\stackrel{\star}{-},\stackrel{\star}{+}]$ into K buckets of multiplicative range $1+\frac{\epsilon}{3}$, i.e. with endpoints $\stackrel{\star}{-}\cdot(1+\frac{\epsilon}{3})^k$ for $0 \leq k \leq K$ and

$$K = O\left(\frac{1}{\epsilon}\cdot\log\left(\frac{\stackrel{\star}{+}}{\stackrel{\star}{-}}\right)\right).$$

We then binary search over $0 \leq k \leq K$ to determine the value of $^\star(v)$ to ϵ-multiplicative accuracy, returning the largest endpoint for which the decision variant in Proposition 66 returns feasible (with

accuracy $\frac{\epsilon}{3}$). By the guarantees of Proposition 66, the feasible point returned by Proposition 66 for this endpoint will attain an ϵ-multiplicative approximation to the optimization variant (7.59), and the runtime is that of Proposition 66 with an overhead of $O(\log K)$. □

F.3 Deferred proofs from Section 7.5

F.3.1 Proofs from Section 7.5.2

Since our notion of approximation is multiplicative, we can assume without more than constant loss that $\mathbf{A}$ has bounded entries. This observation is standard, and formalized in the following lemma.

Lemma 291 (Entrywise bounds on $\mathbf{A}$)**.** *Feasibility of Problem 3 is unaffected (up to constants in ϵ) by removing columns of $\mathbf{A}$ with entries larger than $n\epsilon^{-1}$.*

Proof. If $\mathbf{A}_{ji} > n\epsilon^{-1}$ for any entry, then $x_i \leq \frac{\epsilon(1+\epsilon)}{n}$, else $\|\mathbf{A}x\|_p$ is already larger than $1+\epsilon$. Ignoring all such entries of x and rescaling can only change the objective by a $1 + O(\epsilon)$ factor. □

Lemma 110. *In all iterations t of Algorithm 31, defining $\Phi_t := \|\mathbf{A}w_t\|_p - \|w_t\|_1$, $\Phi_{t+1} \leq \Phi_t$.*

Proof. Fix an iteration t. Define $\delta = \eta g_t$, and note $w_{t+1} = w_t + \delta \circ w_t$; henceforth in this proof, we will drop subscripts t when clear. Observe that

$$\|\mathbf{A}w_{t+1}\|_p = \|\mathbf{A}((1+\delta)\circ w)\|_p = \left(\sum_{j\in[d]} [\mathbf{A}w]_j^p \left(1 + \frac{[\mathbf{A}(\delta \circ w_t)]_j}{[\mathbf{A}w_t]_j}\right)^p\right)^{1/p}.$$

As $g \leq \mathbf{1} \implies \delta \leq p^{-1}\mathbf{1}$, $\frac{\mathbf{A}(\delta\circ w_t)}{\mathbf{A}w_t} \leq p^{-1}$ entrywise. Via $(1+x)^p \leq \exp(px) \leq 1 + px + p^2x^2$ for $x \leq p^{-1}$, it follows that

$$\|\mathbf{A}((1+\delta)\circ w)\|_p \leq \left(\sum_{j\in[d]} [\mathbf{A}w]_j^p \left(1 + \frac{p[\mathbf{A}(\delta\circ w)]_j}{[\mathbf{A}w]_j} + \left(\frac{p[\mathbf{A}(\delta\circ w)]_j}{[\mathbf{A}w]_j}\right)^2\right)\right)^{1/p}.$$

By direct manipulation of the above quantity, and recalling we defined $v = \frac{\mathbf{A}w}{\|\mathbf{A}w\|_p}$,

$$\begin{aligned}
&\left(\sum_{j\in[d]}\left([\mathbf{A}w]_j^p + p[\mathbf{A}w]_j^{p-1}[\mathbf{A}(\delta\circ w)]_j + p^2[\mathbf{A}w]_j^{p-2}[\mathbf{A}(\delta\circ w)]_j^2\right)\right)^{1/p} \\
&= \left(\|\mathbf{A}w\|_p^p \sum_{j\in[d]}\left(v_j^p + pv_j^{p-1}\frac{[\mathbf{A}(\delta\circ w)]_j}{\|\mathbf{A}w\|_p} + p^2v_j^{p-2}\left(\frac{[\mathbf{A}(\delta\circ w)]_j}{\|\mathbf{A}w\|_p}\right)^2\right)\right)^{1/p} \\
&= \|\mathbf{A}w\|_p\left(1 + \sum_{j\in[d]}\left(pv_j^{p-1}\frac{[\mathbf{A}(\delta\circ w)]_j}{\|\mathbf{A}w\|_p} + p^2v_j^{p-2}\left(\frac{[\mathbf{A}(\delta\circ w)]_j}{\|\mathbf{A}w\|_p}\right)^2\right)\right)^{1/p}.
\end{aligned}$$

Using $(1+x)^p > 1+px$, i.e. $(1+px)^{1/p} < 1+x$, we thus obtain

$$\|\mathbf{A}((1+\delta)\circ w)\|_p \le \|\mathbf{A}w\|_p + \left\langle v^{p-1}, \mathbf{A}(\delta\circ w)\right\rangle + p\left\langle v^{p-1}, \frac{(\mathbf{A}(\delta\circ w))^2}{\mathbf{A}w}\right\rangle.$$

Cauchy-Schwarz yields that $[\mathbf{A}(\delta\circ w)]_j^2 \le [\mathbf{A}(\delta^2\circ w)]_j[\mathbf{A}w]_j$, $\forall j\in[d]$. Substituting into the above,

$$\begin{aligned}
\|\mathbf{A}((1+\delta)\circ w)\|_p &\le \|\mathbf{A}w\|_p + \left\langle v^{p-1}, \mathbf{A}(\delta\circ w)\right\rangle + p\left\langle v^{p-1}, \mathbf{A}(\delta^2\circ w)\right\rangle \\
&= \|\mathbf{A}w\|_p + \sum_{j\in[d]}\left[\mathbf{A}^\top v^{p-1}\right]_j\delta_j w_j(1+p\delta_j).
\end{aligned} \tag{F.20}$$

Finally, to bound this latter quantity, since $\delta = \eta g$, we observe that for all j either $\delta_j = 0$ or $1+p\delta_j = 1+g_j = 2-[\mathbf{A}^\top v^{p-1}]_j$, in which case

$$\left[\mathbf{A}^\top v^{p-1}\right]_j(1+p\delta_j) = \left[\mathbf{A}^\top v^{p-1}\right]_j\left(2-\left[\mathbf{A}^\top v^{p-1}\right]_j\right) \le 1.$$

Thus, plugging this bound into (F.20) entrywise,

$$\|\mathbf{A}((1+\delta)\circ w)\|_p - \|\mathbf{A}w\|_p \le \sum_{j\in[d]}\delta_j w_j\left[\mathbf{A}^\top v^{p-1}\right]_j(1+p\delta_j) \le \sum_{j\in[d]}\delta_j w_j = \|w_{t+1}\|_1 - \|w_t\|_1.$$

Rearranging yields the desired claim. □

F.3.2 Proofs from Section 7.5.3

Our analysis of Algorithm 32 will use the following helper fact.

Lemma 292 (Spectral bounds on $\{\mathbf{A}_i\}_{i\in[n]}$)**.** *Feasibility of Problem 4 is unaffected (up to constants in ϵ) by removing matrices $\mathbf{A}_i$ with an eigenvalue larger than $n\epsilon^{-1}$.*

Proof. The proof is identical to Lemma 291; we also require the additional fact that the Schatten

norm $\|\cdot\|_p$ is monotone in the Loewner order, forcing the constraint $x_i \leq \frac{\epsilon(1+\epsilon)}{n}$. □

We remark that we can perform this preprocessing procedure via power iteration on each $\mathbf{A}_i$.

Lemma 111. *In all iterations t of Algorithm 32, defining $\Phi_t := \left\|\sum_{i\in[n]}[w_t]_i\mathbf{A}_i\right\|_p - \|w_t\|_1$, $\Phi_{t+1} \leq \Phi_t$.*

Proof. Drop t and define $\delta = \eta g$. For simplicity, define the matrices

$$\mathbf{M}_0 := \sum_{i\in[n]} w_i\mathbf{A}_i,\ \mathbf{M}_1 := \sum_{i\in[n]} \delta_i w_i\mathbf{A}_i,\ \mathbf{M}_2 := \sum_{i\in[n]} \delta_i^2 w_i\mathbf{A}_i.$$

We recall the Lieb-Thirring inequality $\mathrm{Tr}((\mathbf{ABA})^p) \leq \mathrm{Tr}(\mathbf{A}^{2p}\mathbf{B}^p)$. Applying this, we have

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p^p = \mathrm{Tr}\left((\mathbf{M}_0 + \mathbf{M}_1)^p\right) \leq \mathrm{Tr}\left(\mathbf{M}_0^p\left(\mathbf{I} + \mathbf{M}_0^{-\frac{1}{2}}\mathbf{M}_1\mathbf{M}_0^{-\frac{1}{2}}\right)^p\right).$$

As $g \leq \mathbf{1}$, we have $\mathbf{M}_0^{-\frac{1}{2}}\mathbf{M}_1\mathbf{M}_0^{-\frac{1}{2}} \preceq p^{-1}\mathbf{I}$. Applying the bounds $(\mathbf{I}+\mathbf{M})^p \preceq \exp(p\mathbf{M}) \preceq \mathbf{I} + p\mathbf{M} + p^2\mathbf{M}^2$ for $\mathbf{M} = \mathbf{M}_0^{-\frac{1}{2}}\mathbf{M}_1\mathbf{M}_0^{-\frac{1}{2}}$, where we use that $\mathbf{I}$ commutes with all $\mathbf{M}$, it follows that

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p^p \leq \mathrm{Tr}\left(\mathbf{M}_0^p + p\mathbf{M}_0^{p-1}\mathbf{M}_1 + p^2\mathbf{M}_0^{p-1}\mathbf{M}_1\mathbf{M}_0^{-1}\mathbf{M}_1\right).$$

Definitions of $\mathbf{M}_0$, $\mathbf{M}_1$, $\mathbf{M}_2$, and preservation of positiveness under Schur complements imply

$$\begin{pmatrix}\mathbf{M}_0 & \mathbf{M}_1\\ \mathbf{M}_1 & \mathbf{M}_2\end{pmatrix} \succeq 0 \implies \mathbf{M}_2 - \mathbf{M}_1\mathbf{M}_0^{-1}\mathbf{M}_1 \succeq 0.$$

Thus, $\mathbf{M}_1\mathbf{M}_0^{-1}\mathbf{M}_1 \preceq \mathbf{M}_2$. Applying this and recalling $\mathbf{V} = \frac{\mathbf{M}_0}{\|\mathbf{M}_0\|_p}$,

$$\begin{aligned}\|\mathbf{M}_0 + \mathbf{M}_1\|_p^p &\leq \mathrm{Tr}\left(\mathbf{M}_0^p + p\mathbf{M}_0^{p-1}\mathbf{M}_1 + p^2\mathbf{M}_0^{p-1}\mathbf{M}_2\right)\\ &= \|\mathbf{M}_0\|_p^p\left(1 + p\left\langle\mathbf{V}^{p-1}, \frac{\mathbf{M}_1}{\|\mathbf{M}_0\|_p} + \frac{p\mathbf{M}_2}{\|\mathbf{M}_0\|_p}\right\rangle\right).\end{aligned}$$

By $(1+px)^{1/p} < 1 + x$, taking p^{th} roots we thus have

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p \leq \|\mathbf{M}_0\|_p + \left\langle\mathbf{V}^{p-1}, \mathbf{M}_1 + p\mathbf{M}_2\right\rangle.$$

Finally, the conclusion follows as in Lemma 110; by linearity of trace and $g = p\delta$,

$$\left\langle\mathbf{V}^{p-1}, \mathbf{M}_1 + p\mathbf{M}_2\right\rangle = \sum_{i\in[n]}\left\langle\mathbf{V}^{p-1}, \mathbf{A}_i\right\rangle\delta_i w_i(1 + p\delta_i) \leq \sum_{i\in[n]}\delta_i w_i.$$

Here, we used the inequality for all nonzero g_i,

$$\left\langle \mathbf{V}^{p-1}, \mathbf{A}_i \right\rangle (1 + p\delta_i) = \left\langle \mathbf{V}^{p-1}, \mathbf{A}_i \right\rangle \left(2 - \left\langle \mathbf{V}^{p-1}, \mathbf{A}_i \right\rangle\right) \leq 1.$$

□

Theorem 48. *Let p be odd. Algorithm 32 runs in $O(\frac{p \log(nd/\epsilon)}{\epsilon})$ iterations, and its output solves Problem 4. Each iteration is implementable in $O(\text{nnz} \cdot \frac{p \log(nd/\epsilon)}{\epsilon^2})$, where* nnz *is the number of nonzero entries amongst all $\{\mathbf{A}_i\}_{i \in [n]}$, losing $O(\epsilon)$ in the quality of Problem 4 with probability $1 - \text{poly}((nd/\epsilon)^{-1})$.*

Proof. The proof is analogous to that of Theorem 47; we sketch the main differences here. By applying Lemma 292 and monotonicity of Schatten norms in the Loewner order, we again have $\Phi_0 \leq 1$, implying correctness whenever the algorithm terminates on Line 4. Correctness of dual certification again follows from lack of termination and the choice of T, as well as setting u to indicate each coordinate. Finally, the returned matrix in Line 8 is correct by convexity of the Schatten-q norm, and the fact that all $\mathbf{V}_t^{p-1}$ have unit Schatten-q norm.

We now discuss issues regearding computing g_t in Line 5 of the algorithm, the bottleneck step; these techniques are standard in the approximate SDP literature, and we defer a more formal discussion to e.g. [285]. First, note that each coordinate of g_t requires us to compute

$$\frac{1}{\left\|\sum_{i \in [n]} [w_t]_i \mathbf{A}_i\right\|_p^{p-1}} \cdot \left\langle \mathbf{A}_i, \left(\sum_{i \in [n]} [w_t]_i \mathbf{A}_i\right)^{p-1} \right\rangle. \tag{F.21}$$

We estimate the two quantities in the above expression each to $1 + \epsilon$ multiplicative error with high probability. Union bounding over iterations, and modifying Lemma 111 to use the potential $\left\|\sum_{i \in [n]} [w_t]_i \mathbf{A}_i\right\|_p - (1 + O(\epsilon)) \|w_t\|_1$, the analysis remains valid up to constants in ϵ with this multiplicative approximation quality. We now discuss our approximation strategies.

For shorthand, denote $\mathbf{M} = \sum_{i \in [n]} [w_t]_i \mathbf{A}_i$. To estimate the denominator of (F.21), it suffices to multiplicatively approximate $\|\mathbf{M}\|_p^p = \text{Tr}[\mathbf{M}^p]$ within a $1 + \epsilon$ factor, as raising to the $\frac{p-1}{p}$ power can only improve this. To do so, we use the well-known fact (e.g. [160]) that letting $\mathbf{Q}$ be a $k \times d$ matrix with independent entries $\sim \mathcal{N}(0, \frac{1}{k})$, for $k = O(\frac{\log(\frac{nd}{\epsilon})}{\epsilon^2})$, with probability $1 - \text{poly}((\frac{nd}{\epsilon})^{-1})$,

$$\text{Tr}[\mathbf{M}^p] \approx \sum_{\ell \in [k]} \mathbf{Q}_{\ell:}^\top \mathbf{M}^p \mathbf{Q}_{\ell:}$$

to a $1 + \epsilon$ factor. To read this from the standard Johnson-Lindestrauss guarantee, it suffices to factorize $\mathbf{M}^p$ and use that each row of the square root's ℓ_2 norm is preserved with high probability under multiplication by $\mathbf{Q}$, and then apply the cyclic definition of trace. Similarly, for each $i \in [n]$,

we can approximate the numerators via

$$\operatorname{Tr}\left(\mathbf{Q}\mathbf{M}^{\frac{p-1}{2}}\mathbf{A}_i\mathbf{M}^{\frac{p-1}{2}}\mathbf{Q}^\top\right).$$

We can simultaneously compute all such quantities by first applying $O(p)$ matrix-vector multiplications through $\mathbf{M}$ to each row of $\mathbf{Q}$, and then computing all quadratic forms. In total, the computational cost per iteration of all approximations is $O(\operatorname{nnz}\cdot\frac{p\log(\frac{nd}{\epsilon})}{\epsilon^2})$ as desired. □

F.3.3 Proof of Proposition 30

In this section, following our prior developments, we prove the following claim.

Proposition 30. *Following Theorem 48's notation, let p be odd, $\{\mathbf{A}_i\}_{i\in[n]}\in\mathbb{S}^d_{\succeq 0}$, $0<\epsilon=O(\alpha)$, and*

$$\min_{\substack{x\in\Delta^n\\ \|x\|_\infty\le\frac{1+\alpha}{n}}}\|\mathcal{A}(x)\|_p=\mathrm{OPT}. \tag{7.74}$$

for $\mathcal{A}(x):=\sum_{i\in[n]}x_i\mathbf{A}_i$. Given estimate of OPT *exponentially bounded in $\frac{nd}{\epsilon}$, there is a procedure calling Algorithm 76 $O(\log\frac{nd}{\epsilon})$ times giving $x\in\Delta^n$ with $\|x\|_\infty\le\frac{(1+\alpha)(1+\epsilon)}{n}$, $\|\mathcal{A}(x)\|_p\le(1+\epsilon)^\star$. Algorithm 76 runs in $O(\frac{\log(nd/\epsilon)\log n}{\epsilon^2})$ iterations, each implementable in time $O(\operatorname{nnz}\cdot\frac{p\log(nd/\epsilon)}{\epsilon^2})$.*

Reduction to a decision problem

Given access to an oracle for the following approximate decision problem, we can implement an efficient binary search for estimating *. Specifically, letting the range of * be $(\mu_{\text{lower}},\mu_{\text{upper}})$, we can subdivide the range into $O(\frac{1}{\epsilon}\log\frac{\mu_{\text{upper}}}{\mu_{\text{lower}}})$ multiplicative intervals of range $1+\epsilon$, and then compute a binary search using our decision oracle. This incurs a multiplicative $\log(\frac{nd}{\epsilon})$ overhead in the setting of Proposition 30 (see Appendix A, [285], for a more formal treatment).

Problem 5. *Given $\{\mathbf{A}_i\}_{i\in[n]}\in\mathbb{S}^d_{\succeq 0}$, either find primal solution $x\in\Delta^n$ with $\|\mathcal{A}(x)\|_p\le 1+\epsilon$, $\|x\|_\infty\le\frac{(1+\epsilon)(1+\alpha)}{n}$, or conclude no $x\in\Delta^n$ satisfies $\|\mathcal{A}(x)\|_p\le 1-\epsilon$, $\|x\|_\infty\le\frac{(1-\epsilon)(1+\alpha)}{n}$.*

The hard constraint $\|x\|_\infty\le\frac{1+\alpha}{n}$ in the definition (7.74) can be adjusted by constant factors to admit the ℓ_∞ bound in Problem 5, since we assumed $\epsilon=O(\alpha)$ is sufficiently small.

Preliminaries

We use the shorthand $\mathbf{S}:=\frac{n}{1+\alpha}\mathbf{I}$, and $p':=\frac{\log n}{\epsilon}$, so $\ell_{p'}$ and ℓ_∞ are interchangeable up to $1+O(\epsilon)$ factors. In other words, Problem 5 asks to certify whether there exists $x\in\Delta^n$ with

$$\max\left(\|\mathcal{A}(x)\|_p,\ \|\mathbf{S}x\|_{p'}\right)\le 1, \tag{F.22}$$

up to multiplicative $1+\epsilon$ tolerance on either side. Consider the potential function

$$\Phi(w) := \log\left(\exp\left(\|\mathcal{A}(w)\|_p\right) + \exp\left(\|\mathbf{S}w\|_{p'}\right)\right) - \|w\|_1 . \tag{F.23}$$

It is clear that the first term of $\Phi(w)$ approximates the left hand side of (F.22) up to a $\log 2$ additive factor, so if any of $\|\mathcal{A}(w)\|_p$, $\|\mathcal{A}(w)\|_{p'}$, or $\|w\|_1$ reaches the scale $3\epsilon^{-1}$ and $\Phi(w)$ is bounded by 1, we can safely terminate. and conclude primal feasibility for Problem 5. Next, we compute

$$\begin{aligned}\nabla_i \Phi(w) = 1 - \frac{\exp\left(\|\mathcal{A}(w)\|_p\right)\langle \mathbf{A}_i, \mathbf{Y}(w)\rangle + \exp\left(\|\mathbf{S}w\|_{p'}\right)[\mathbf{S}z(w)]_i}{\exp\left(\|\mathcal{A}(w)\|_p\right) + \exp\left(\|\mathbf{S}w\|_{p'}\right)} \text{ for all } i \in [n],\\ \text{where } \mathbf{Y}(w) := \left(\frac{\mathcal{A}(w)}{\|\mathcal{A}(w)\|_p}\right)^{p-1},\ z(w) := \left(\frac{\mathbf{S}w}{\|\mathbf{S}w\|_{p'}}\right)^{p'-1}\end{aligned} \tag{F.24}$$

The following helper lemma will be useful in concluding dual infeasibility of Problem 5.

Lemma 293. *In the setting of Problem 5, suppose there exists $x^* \in \Delta^n$ with*

$$\|\mathcal{A}(x^*)\|_p \le 1-\epsilon,\ \|\mathbf{S}x^*\|_{p'} \le 1-\epsilon.$$

Then, for any w,

$$\langle \nabla\Phi(w), x^*\rangle \ge \epsilon.$$

Proof. From the definitions in (F.24), it is clear that $\|\mathbf{Y}(w)\|_q = \|z(w)\|_{q'} = 1$, where q, q' are the dual norms of p, p' respectively. Moreover, by the definition of x^*, we have for all $\|\mathbf{Y}\|_q = \|z\|_{q'} = 1$,

$$\langle \mathbf{Y}, \mathcal{A}(x)\rangle \le 1-\epsilon,\ \langle z, \mathbf{S}x\rangle \le 1-\epsilon.$$

This follows from the dual definition of the ℓ_p norm. Now, note that for some nonnegative $\alpha(w)$, $\beta(w)$ summing to 1, using the above claim and (F.24),

$$\langle \nabla\Phi(w), x^*\rangle = 1 - (\alpha(w)\langle \mathbf{Y}(w), \mathcal{A}(x^*)\rangle + \beta(w)\langle z(w), \mathbf{S}x^*\rangle) \ge \epsilon,$$

as desired (here, we used positivity of all relevant quantities). □

Potential monotonicity

We prove a monotonicity property regarding the potential Φ in (F.23).

Lemma 294. *Let $w \in \mathbb{R}^n_{\ge 0}$ satisfy $\|\mathcal{A}(w)\|_p \le 3\epsilon^{-1}$, $\|\mathbf{S}w\|_{p'} \le 3\epsilon^{-1}$, let $g = \max(0, \nabla\Phi(w))$ entrywise, and let $w' = (1-\eta g)\circ w$, where $\eta = (4p')^{-1}$. Then, $\Phi(w') \le \Phi(w)$.*

Proof. Denote for simplicity the threshold $K = 3\epsilon^{-1}$ and the step vector $\delta = \eta g$. First, by prior calculations in Lemma 110 and Lemma 111, it follows that

$$\|\mathcal{A}(w')\|_p \le \|\mathcal{A}(w)\|_p + \Delta_{\mathcal{A}},\ \|\mathbf{S}w'\|_{p'} \le \|\mathbf{S}w\|_{p'} + \Delta_{\mathbf{S}},$$
$$\text{where } \Delta_{\mathcal{A}} := \sum_{i\in[n]} \langle \mathbf{A}_i, \mathbf{Y}(w)\rangle\, \delta_i w_i(1+p\delta_i),\ \Delta_{\mathbf{S}} := \sum_{i\in[n]} [\mathbf{S}z(w)]_i \delta_i w_i(1+p'\delta_i).$$

Next, note that by $\delta \le \eta$ entrywise and lack of termination (i.e. the threshold K),

$$\Delta_{\mathcal{A}} \le (1+p\eta)\eta \langle \mathbf{Y}(w), \mathcal{A}(w)\rangle \le 2\eta \|\mathcal{A}(w)\|_p \le 1.$$

Therefore, by $\exp(x) \le 1 + x + x^2$ for $x \le 1$,

$$\exp\left(\|\mathcal{A}(w')\|_p\right) \le \exp\left(\|\mathcal{A}(w)\|_p\right)\left(1 + \Delta_{\mathcal{A}} + \Delta_{\mathcal{A}}^2\right). \tag{F.25}$$

Moreover, by applying Cauchy-Schwarz and the threshold $\|\mathcal{A}(w)\|_p \le K$ once more,

$$\begin{aligned}\Delta_{\mathcal{A}}^2 &\le (1+p\eta)^2 \left(\sum_{i\in[n]} \langle \mathbf{A}_i, \mathbf{Y}(w)\rangle\, \delta_i w_i\right)^2 \\ &\le 2\left(\sum_{i\in[n]} \langle \mathbf{A}_i, \mathbf{Y}(w)\rangle\, \delta_i^2 w_i\right)\langle \mathbf{Y}(w), \mathcal{A}(w)\rangle \le 2K\left(\sum_{i\in[n]} \langle \mathbf{A}_i, \mathbf{Y}(w)\rangle\, \delta_i^2 w_i\right).\end{aligned} \tag{F.26}$$

Combining (F.25) and (F.26) (and applying similar reasoning to the term $\Delta_{\mathbf{S}}$), we conclude

$$\exp\left(\|\mathcal{A}(w')\|_p\right) \le \exp\left(\|\mathcal{A}(w)\|_p\right)\left(1 + \sum_{i\in[n]} \langle \mathbf{A}_i, \mathbf{Y}(w)\rangle\, \delta_i w_i(1 + (p+2K)\delta_i)\right),$$
$$\exp\left(\|\mathbf{S}w'\|_{p'}\right) \le \exp\left(\|\mathbf{S}w\|_{p'}\right)\left(1 + \sum_{i\in[n]} [\mathbf{S}z(w)]_i \delta_i w_i(1 + (p'+2K)\delta_i)\right).$$

Recall the inequality $\log(1+x) \le x$ for nonnegative x. Expanding the definition of Φ and $\nabla\Phi$ (cf.

(F.23)), and plugging in the above bounds, we conclude that

$$\begin{aligned}\Phi(w') - \Phi(w) &= \log\left(\frac{\exp\left(\left\|\mathcal{A}(w')\right\|_p\right) + \exp\left(\left\|\mathbf{S}w'\right\|_{p'}\right)}{\exp\left(\left\|\mathcal{A}(w)\right\|_p\right) + \exp\left(\left\|\mathbf{S}w\right\|_{p'}\right)}\right) - \langle \delta, w\rangle \\ &\le \sum_{i\in[n]} (1 - \nabla_i\Phi(w))\delta_i w_i(1 + (p'+2K)\delta_i) - \langle \delta, w\rangle \\ &= \sum_{i\in[n]} \left((1 - \nabla_i\Phi(w))(1 + (p'+2K)\delta_i) - 1\right)\delta_i w_i.\end{aligned}$$

As before, we show that this sum is entrywise nonpositive. For any $i \in [n]$ with $\delta_i \neq 0$, we have

$$\begin{aligned}(1 - \nabla_i\Phi(w))(1 + (p'+2K)\delta_i) - 1 &= (1 - \nabla_i\Phi(w))(1 + (p'+2K)\eta\nabla_i\Phi(w)) - 1 \\ &\le (1 - \nabla_i\Phi(w))(1 + \nabla_i\Phi(w)) - 1 \le 0,\end{aligned}$$

as desired, where we used that $\eta^{-1} \ge p' + 2K$. This yields the conclusion $\Phi(w') \le \Phi(w)$. □

Algorithm and analysis

Finally, we state Algorithm 76 and prove Proposition 30.

Algorithm 76: BoxedSchattenPacking($\{\mathbf{A}_i\}_{i\in[n]}, \epsilon, p, \alpha$)

Input: $\{\mathbf{A}_i\}_{i\in[n]} \in \mathbb{S}^d_{\succeq 0}, \epsilon \in [0, \frac{1}{2}], p \ge 2, \alpha \in [0, n-1]$;
$p' \leftarrow \frac{\log n}{\epsilon}$, $\mathbf{S} \leftarrow \frac{n}{1+\alpha}\mathbf{I}$;
$\eta \leftarrow (4p')^{-1}$, $K \leftarrow 3\epsilon^{-1}$, $T \leftarrow \frac{6\log(\frac{nd}{\epsilon})}{\eta\epsilon}$;
$[w_0]_i \leftarrow \frac{\epsilon}{n^2 d}$ for all $i \in [n]$, $t \leftarrow 0$;
while $\left\|\mathcal{A}(w_t)\right\|_p, \left\|\mathbf{S}w_t\right\|_{p'}, \left\|w_t\right\|_1 \le K$ **do**
  $g_t \leftarrow \max(0, \nabla\Phi(w_t))$ entrywise, where we use the definition (F.23);
  $w_{t+1} \leftarrow w_t \circ (1 + \eta g_t)$, $t \leftarrow t + 1$;
  if $t \ge T$ **then**
    Return: Infeasible;
Return: $x = \frac{w_t}{\|w_t\|_1}$;

Proof of Proposition 30. Correctness of the reduction to deciding Problem 5 follows from the discussion in Section F.3.3. Moreover, by the given Algorithm 76, it is clear (following e.g. the preprocessing of Lemma 292) that $\Phi(w_t) \le 1$ throughout the algorithm, so whenever the algorithm terminates we have primal feasibility. It suffices to prove that whenever the problem admits x^* with

$$\left\|\mathcal{A}(x^*)\right\|_p \le 1 - \epsilon, \ \left\|\mathbf{S}x^*\right\|_{p'} \le 1 - \epsilon,$$

then the algorithm terminates on Line 5 in T iterations. Analogously to Theorem 47, we have

$$\eta(1-\eta)\sum_{0\leq t<T}\langle g_t, x^*\rangle \leq \log n - \log\|w_0\|_1 + \log\|w_T\|_1 \leq 2\log\left(\frac{nd}{\epsilon}\right) + \log\|w_T\|_1 .$$

Next, since g_t is an upwards truncation of $\nabla\Phi(w_t)$, applying Lemma 293 implies that

$$\|w_T\|_1 \geq \exp\left(\frac{\eta\epsilon T}{2} - 2\log\left(\frac{nd}{\epsilon}\right)\right).$$

The conclusion follows by the definition of T, as desired. Finally, the iteration complexity follows analogously to the discussion in Theorem 48's proof, where the only expensive cost is estimating coordinates of the $\mathcal{A}$ component of $\nabla\Phi(w_t)$ every iteration. □

Finally, we remark that by opening up the dual certificates $\mathbf{Y}(w)$, $\mathbf{Z}(w)$ of our mirror descent analysis, we can in fact implement a stronger version of the decision Problem 5 which returns a feasible dual certificate whenever the primal problem is infeasible. We omit this extension for brevity, as it is unnecessary for our applications, but it is analogous to the analysis of Theorem 48.

Appendix G

Deferred proofs from Chapter 8

G.1 List-decodable mean estimation for $\alpha^{-1} = \Omega(d)$

We give a simple algorithm for list-decodable mean estimation in the regime $\alpha^{-1} = \Omega(d)$.

Algorithm 77: SamplePostProcess(T, δ)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumption 13, $\alpha \le \frac{1}{Cd}$ for a universal constant C, $\delta \in (0, 1)$;
Output: $L \subset \mathbb{R}^d$ with $|L| \le \frac{3}{\alpha}$ satisfying (G.2) with probability $\ge 1 - \delta$;
$N \leftarrow \left\lceil \frac{36 \log(2/\delta)}{\alpha} \right\rceil$;
$\widetilde{L} \leftarrow \{X_i\}_{i \in [N]}$, where each X_i is an independent uniform sample from T;
$\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log(\frac{1}{\alpha\delta}))$;
Let L be a maximal subset of $\widetilde{L}$ such that for each $X_i \in L$, $\left\|\mathbf{G}^\top (X_i - X_j)\right\|_2^2 \le 8.8d$ for at least $\frac{\alpha N}{3}$ of the $X_j \in \widetilde{L}$, and $\left\|\mathbf{G}^\top (X_i - X_j)\right\|_2^2 \ge 35.2d$, $\forall X_j \in L$;
L;

Proposition 67. *Algorithm 77,* SamplePostProcess*, meets its output specifications in runtime*

$$O\left(\frac{1}{\alpha^2} \log^4\left(\frac{1}{\alpha\delta}\right)\right).$$

Proof. It is straightforward by Assumption 13 (cf. correctness proof of Theorem 89) that at least $\frac{\alpha n}{2}$ of the points $X_i \in T$ satisfy

$$\|X_i - \mu^*\|_2^2 \le 2d. \tag{G.1}$$

For each $i \in [N]$ indexing the set $\widetilde{L}$, let E_i be the event that X_i satisfies the bound (G.1); each of these events is an independent Bernoulli variable with mean at least $\frac{\alpha}{2}$. Thus, by applying a Chernoff bound, with probability at least $1 - \frac{\delta}{2}$, at least $\frac{\alpha N}{3}$ of the points in $\widetilde{L}$ satisfy (G.1). Next,

by the Johnson-Lindenstrauss lemma of [6], for a sufficiently large dimensionality c, with probability at least $1 - \frac{\delta}{2}$, all of the $\left\|\mathbf{G}^\top (X_i - X_j)\right\|_2^2$ are within a 1.1 factor of the corresponding $\|X_i - X_j\|_2^2$. Condition on both of these events for the remainder of the proof.

By definition of the greedy process in Line 6, we have the output size guarantee, since each element of $\widetilde{L}$ is associated with a (disjoint) cluster of $\frac{\alpha N}{3}$ points, by the separation property. So, for correctness, it suffices to prove that (G.2) is met for a universal constant (depending on C). Call $\widetilde{S}$ the set of points in T satisfying (G.1). If any point in $\widetilde{S}$ is chosen in L, then indeed

$$\|X_i - \mu^*\|_2^2 \le 2d \le \frac{2}{C\alpha},$$

so (G.2) is met with constant $\sqrt{\frac{2}{C}}$. Further, observe that the only thing preventing any point in $\widetilde{S}$ from being chosen is the separation condition for L. This is because by triangle inequality and the definition (G.1), any pair of points $X_i, X_j \in \widetilde{S}$ satisfies $\|X_i - X_j\|_2^2 \le 8d$, so after multiplication by $\mathbf{G}^\top$ they pass the clustering requirement. Thus, suppose no point in $\widetilde{S}$ is in L. For any $X_i \in \widetilde{S} \cup \widetilde{L}$, this implies there exists a $X_j \in \widetilde{L}$ with

$$\left\|\mathbf{G}^\top (X_i - X_j)\right\|_2^2 \le 35.2d \implies \|X_i - X_j\|_2^2 \le 40d.$$

By triangle inequality, this implies that (G.2) is met with constant $\sqrt{\frac{84}{C}}$, via

$$\|X_j - \mu^*\|_2^2 \le 84d \le \frac{84}{C\alpha}.$$

Finally, the runtime is dominated by the cost of multiplying all points in $\widetilde{L}$ by $\mathbf{G}^\top$, and performing all pairwise distance comparisons of the $\{\mathbf{G}^\top X_i\}_{i \in [N]}$. Both of these fit in the allotted time budget. □

We make a final remark that up to logarithmic factors, the runtime in Proposition 67 is not larger than $\frac{nd}{\alpha}$ asymptotically, since we take sample size $n \ge \alpha^{-1}$. Thus, in the regime $\alpha^{-1} = \Omega(d)$, we obtain the correct list size and error bound up to constants, in time $\widetilde{O}(\frac{nd}{\alpha})$ as desired.

G.2 Filtering in k dimensions: SIFT

In this section, we develop a simple, polynomial-time algorithm for solving the list-decodable mean estimation problem based on a "soft downweighting" approach. We outline some preliminary notions and bounds used in our algorithms and analysis in Sections G.2.1 and G.2.2, which will also be used in Sections G.3 and G.4. We then use these tools to analyze our "slow" algorithm, SIFT, in Section G.2.3.

G.2.1 General preliminaries

We will frequently use the following well-known facts throughout the chapter. In both, $w \in \Delta^n$ is a weight vector corresponding to a set of points $T \subseteq \mathbb{R}^d$.

Fact 38. *We have that*

$$\mathbf{0} \preceq \sum_{i\in T} w_i (X_i - \mu_w(T))(X_i - \mu_w(T))^\top \implies \mu_w(T)\mu_w(T)^\top \preceq \sum_{i\in T} \frac{w_i}{\|w\|_1} X_i X_i^\top.$$

Thus, for any vector $v \in \mathbb{R}^d$,

$$(\mu_w(T) - v)(\mu_w(T) - v)^\top \preceq \sum_{i\in T} \frac{w_i}{\|w\|_1} (X_i - v)(X_i - v)^\top.$$

Fact 39. *For any vector $v \in \mathbb{R}^d$,*

$$\begin{aligned}\sum_{i\in[n]} w_i (X_i - v)(X_i - v)^\top &= \sum_{i\in[n]} w_i (X_i - \mu_w(T))(X_i - \mu_w(T))^\top + \|w\|_1 (\mu_w(T) - v)(\mu_w(T) - v)^\top \\ &\succeq \sum_{i\in[n]} w_i (X_i - \mu_w(T))(X_i - \mu_w(T))^\top.\end{aligned}$$

In the *list-decodable mean estimation* problem, we are given a set T of n points $\{X_i\}_{i\in T}$ in $\mathbb{R}^d$.[1] For some known $\alpha \in (0, \frac{1}{2}]$, there is a subset $S \subseteq T$ of size αn such that all $\{X_i\}_{i\in S}$ are independent draws from distribution $\mathcal{D}$ with mean μ^*, where the covariance of $\mathcal{D}$ is identity-bounded:

$$\mathbb{E}_{x\sim\mathcal{D}} \left[(x - \mu^*)(x - \mu^*)^\top \right] \preceq \mathbf{I}.$$

It is clear that by scaling the space, this assumption appropriately generalizes to the case when the covariance bound is $\sigma^2\mathbf{I}$. The goal of list-decodable mean estimation is to output a list L, such that one of the elements of the list is close to the "true mean" μ^*. Our aim will be to output a list of size $|L| = O(\frac{1}{\alpha})$, which is necessary simply by identifiability of the subset S; it was shown as Proposition 5.4(ii) of [186] that for such a list size, the minimax optimal error for the problem scales as

$$\min_{\mu\in L} \|\mu - \mu^*\|_2 = \Theta\left(\frac{1}{\sqrt{\alpha}}\right). \tag{G.2}$$

Regarding the sample size n, we additionally recall the following (note in Assumption 13 that the matrix of interest is *not* the covariance of S, as it is centered at the true mean μ^*).

Proposition 68 (Proposition B.1, [118])**.** *For any constant $\epsilon \in (0, 1)$, there are constants $c, C > 0$*

[1] In an abuse of notation, we will both let T denote the set of points itself, as well as an index set for the points. Correspondingly, we will interchangeably use $X_i \in T$ and $i \in T$.

such that with probability at least $1 - \exp(-\Omega(n))$*, for* $n = \frac{Cd}{\alpha}$*, if an* $(1+\epsilon)\alpha$ *fraction of points in* $\{X_i\}_{i \in T} \subseteq \mathbb{R}^d$ *is drawn from* $\mathcal{D}$ *with covariance bounded by* $c\mathbf{I}$*, then Assumption 13 holds.*

Assumption 13. *There is a subset* $S \subseteq \{X_i\}_{i \in T} \subseteq \mathbb{R}^d$ *of size* $\alpha n = \Theta(d)$ *satisfying*

$$\frac{1}{|S|} \sum_{i \in S} (X_i - \mu^*) (X_i - \mu^*)^\top \preceq \mathbf{I}.$$

In the remainder of Sections G.2, G.3, and G.4, we will operate under Assumption 13. We will also explicitly assume that $\frac{1}{\alpha} = o(d)$, and $d \leq n = \Theta(\frac{d}{\alpha})$, for simplicity. The latter assumption is without loss of generality for any failure probability larger than $\exp(-\Omega(d))$; for any smaller failure probability, Proposition 68 implies that the assumption still holds by adjusting the sample size by a logarithmic factor. It is also fairly straightforward to see that the former assumption is also without loss of generality, since in the case $\frac{1}{\alpha} \gg d$, it suffices to sample $O(\frac{1}{\alpha} \log \frac{1}{\delta})$ random points and apply a variant of the post-processing procedure of Section G.4.1 to obtain the correct list size and error guarantee; we give a formal treatment of this case in Section G.1.

Finally, throughout the variable k will be reserved for values which are $\Theta(\frac{1}{\alpha})$ for explicitly stated constants. In particular, many of our algorithms will rely on performing operations such as principal components analysis in $\Theta(\frac{1}{\alpha})$ dimensions. As discussed earlier, this is because a substantial portion of the challenge in the estimation problem is reducing to the problem of learning the mean in $\Theta(\frac{1}{\alpha})$ dimensions, at which point naïve random sampling solves the problem up to logarithmic factors.

G.2.2 Filtering preliminaries

We define two concepts which will be useful in stating guarantees of our downweighting methods.

Definition 56 (Saturated weights)**.** *We call weights* $w \in \Delta^n$ *"saturated" if* $w \leq \frac{1}{n}\mathbf{1}$ *entrywise, and*

$$\|w_S\|_1 \geq \alpha \sqrt{\|w\|_1}.$$

Definition 57 (Safe scores)**.** *We call scores* $\{\tau_i\}_{i \in T} \in \mathbb{R}^n_{\geq 0}$ *"safe with respect to* $w \in \Delta^n$*" if*

$$\sum_{i \in S} \frac{w_i}{\|w_S\|_1} \tau_i \leq \frac{1}{2} \sum_{i \in T} \frac{w_i}{\|w\|_1} \tau_i.$$

When the weights w *are clear from context, we will simply call the scores* τ *"safe".*

In algorithms based on soft filtering in the presence of a small amount of adversarial noise (see e.g. [177, 360, 492]), a typical goal is to remove more "good weight" than "bad weight" from an iteratively updated weight vector. However, when the overwhelming majority of the initial weight is bad, clearly this is too strong of a goal. The intuition for Definition 56 is that a weaker goal suffices for the guarantees of our methods; while the amount of good weight is decreasing throughout,

Definition 56 requires that the good weight becomes more saturated in the weight vector when more weight is removed. We now make the connection between these definitions formal.

Lemma 295. *Consider a set of saturated (cf. Definition 56) weights $w^{(0)}$, and updates of the form:*

1. *For $0 \le t < N$:*
   (a) *Let $\left\{\tau_i^{(t)}\right\}_{i\in T}$ be safe (cf. Definition 57) with respect to $w^{(t)}$.*
   (b) *Update for all $i \in T$:*

$$w_i^{(t+1)} \leftarrow \left(1 - \frac{\tau_i^{(t)}}{\tau_{\max}^{(t)}}\right) w_i^{(t)}, \text{ where } \tau_{\max}^{(t)} := \max_{i\in T | w_i^{(t)} \neq 0} \tau_i^{(t)}. \tag{G.3}$$

Then, the result of the updates $w^{(N)}$ is also saturated.

Proof. First, fix some iteration t, and let $w := w^{(t)}$, $\tau := \tau^{(t)}$, and $w' := w^{(t+1)}$. Define

$$\delta_S := \sum_{i\in S} \frac{w_i - w_i'}{\|w_S\|_1}, \ \delta_T := \sum_{i\in T} \frac{w_i - w_i'}{\|w\|_1}.$$

Note that by the assumption that τ is safe and the iteration (G.3),

$$\delta_S = \frac{1}{\tau_{\max}} \sum_{i\in S} \frac{w_i}{\|w_S\|_1} \tau_i \le \frac{1}{2\tau_{\max}} \sum_{i\in T} \frac{w_i}{\|w\|_1} \tau_i = \frac{1}{2}\delta_T.$$

Hence, using $1 - \frac{1}{2}\delta_T \ge \sqrt{1-\delta_T}$ for all $\delta_T \in [0,1]$, we have

$$\frac{\left\|w_S^{(t+1)}\right\|_1}{\left\|w_S^{(t)}\right\|_1} = 1 - \delta_S \ge \sqrt{1-\delta_T} = \sqrt{\frac{\left\|w_T^{(t+1)}\right\|_1}{\left\|w_T^{(t)}\right\|_1}}. \tag{G.4}$$

Inductively telescoping (G.4), using that $w^{(0)}$ was assumed to be saturated, and finally comparing with Definition 56, yields the desired conclusion that $w^{(N)}$ is saturated. □

We next give three helper lemmas which help reason about how the quality of empirical estimates based on S deteriorate, as the amount of weight allocated to S is reduced. The first shows how the quality of the empirical mean is related to the empirical covariance and proportion of weight in S (and is essentially a rephrasing of Fact A.3 in [136]).

Lemma 296. *Let $w \in \Delta^n$ have $w \le \frac{1}{n}\mathbf{1}$ entrywise. Then,*

$$\|\mu_w(T) - \mu^*\|_2 \le \sqrt{2\|\mathrm{Cov}_w(T)\|_{\mathrm{op}} \frac{\|w\|_1}{\|w_S\|_1} + \frac{2\alpha}{\|w\|_1}}.$$

Proof. Let $w^* \in \Delta^n$ be the weight vector which is $\frac{1}{|S|}$ on coordinates in S, and zero elsewhere. Note that by definition $\langle w, w^* \rangle = \frac{\|w_S\|_1}{\alpha n}$. Next,

$$\begin{aligned}\|\mu_w(T) - \mu^*\|_2^2 &= \max_{\|u\|_2=1} \left\langle \left(\sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu_w(T)) \right) - \left(\sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu^*) \right), u \right\rangle^2 \\ &\leq 2 \max_{\|u\|_2=1} \left\langle \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu_w(T)), u \right\rangle^2 + 2 \max_{\|u\|_2=1} \left\langle \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu^*), u \right\rangle^2.\end{aligned}$$

We bound these two terms separately. First, by applying a quadratic form in u to Fact 38,

$$\begin{aligned}\max_{\|u\|_2=1} \left\langle \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu_w(T)), u \right\rangle^2 &\leq \max_{\|u\|_2=1} \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} \langle X_i - \mu_w(T), u \rangle^2 \\ &= \frac{\|w\|_1}{\alpha n \langle w, w^* \rangle} \max_{\|u\|_2=1} \sum_{i \in T} \frac{w_i}{\|w\|_1} \langle X_i - \mu_w(T), u \rangle^2 \\ &= \|\mathrm{Cov}_w(T)\|_{\mathrm{op}} \frac{\|w\|_1}{\|w_S\|_1}.\end{aligned}$$

Next, by again applying Fact 38, and recalling Assumption 13,

$$\begin{aligned}\max_{\|u\|_2=1} \left\langle \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} (X_i - \mu^*), u \right\rangle^2 &\leq \max_{\|u\|_2=1} \sum_{i \in T} \frac{w_i w_i^*}{\langle w, w^* \rangle} \langle X_i - \mu^*, u \rangle^2 \\ &\leq \frac{\|w\|_\infty}{\langle w, w^* \rangle} \max_{\|u\|_2=1} \sum_{i \in T} w_i^* \langle X_i - \mu^*, u \rangle^2 \leq \frac{\alpha n \|w\|_\infty}{\|w\|_1}.\end{aligned}$$

□

The second shows how the empirical covariance of S grows relative to how much of S is kept.

Lemma 297. *Let $w \in \Delta^n$ have $w \leq \frac{1}{n}\mathbf{1}$ entrywise. Then $\mathrm{Cov}_w(S) \preceq \frac{\alpha}{\|w_S\|_1}\mathbf{I}$.*

Proof. For any vector u with $\|u\|_2 = 1$,

$$\begin{aligned}u^\top \mathrm{Cov}_w(S) u &= \sum_{i \in S} \frac{w_i}{\|w_S\|_1} \langle u, X_i - \mu_w(S) \rangle^2 \\ &\leq \sum_{i \in S} \frac{\alpha w_i^*}{\|w_S\|_1} \langle u, X_i - \mu^* \rangle^2 \leq \frac{\alpha}{\|w_S\|_1} \left\| \sum_{i \in S} w_i^* (X_i - \mu^*)(X_i - \mu^*)^\top \right\|_{\mathrm{op}}.\end{aligned}$$

In the second line we used Fact 39. Using Assumption 13 yields the conclusion. □

The third shows how a bound on the saturation of S in a weight vector can be used to bound the distance between empirical means in S and T via the empirical covariance matrix.

Lemma 298. *We have that*

$$(\mu_w(S) - \mu_w(T))(\mu_w(S) - \mu_w(T))^\top \preceq \frac{\|w\|_1}{\|w_S\|_1}\mathrm{Cov}_w(T).$$

Proof. This follows from the following observations (via Fact 38)

$$\begin{aligned}(\mu_w(S) - \mu_w(T))(\mu_w(S) - \mu_w(T))^\top &\preceq \sum_{i\in S}\frac{w_i}{\|w_S\|_1}(X_i - \mu_w(T))(X_i - \mu_w(T))^\top \\ &\preceq \frac{\|w\|_1}{\|w_S\|_1}\sum_{i\in T}\frac{w_i}{\|w\|_1}(X_i - \mu_w(T))(X_i - \mu_w(T))^\top \\ &= \frac{\|w\|_1}{\|w_S\|_1}\mathrm{Cov}_w(T).\end{aligned}$$

□

G.2.3 Analysis of SIFT

We now present SIFT as Algorithm 78. It requires calls to an approximate k-PCA subroutine Power, the classical simultaneous power iteration method, which is stated as Algorithm 26 in Section 7.3.3, where we present an improved analysis of its guarantees. However, for analysis in this section it suffices to use the following guarantee. For simplicity in this section we drop the arguments $\lambda_{\max}$ and $\lambda_{\min}$ as inputs to Power, which do not play a role in Proposition 69.

Proposition 69 (Theorem 1, [404])**.** *For any $\delta \in (0,1)$ and $k \in [d]$, there is an algorithm,* Power*, which takes as input k, δ, $\mathbf{A} \in \mathbb{S}^d_{\geq 0}$ and $\epsilon \in (0,1)$, and returns with probability $1-\delta$ a set of orthonormal vectors $\mathbf{V} \in \mathbb{R}^{d\times k}$ such that if $\mathbf{V}_{:i}$ is column i of $\mathbf{V}$,*

$$\begin{aligned}\langle \mathbf{V}_{:i}, \mathbf{A}\mathbf{V}_{:i}\rangle \in [1-\epsilon, 1+\epsilon]\,\lambda_i(\mathbf{A}) \text{ for all } i \in [k], \\ \text{and } \left\|\left(\mathbf{I} - \mathbf{V}\mathbf{V}^\top\right)\mathbf{A}\left(\mathbf{I} - \mathbf{V}\mathbf{V}^\top\right)\right\|_{\mathrm{op}} \leq (1+\epsilon)\lambda_{k+1}(\mathbf{A}).\end{aligned}$$

When $\mathbf{A}$ is given in the form $\mathbf{M}^\top\mathbf{M}$ for some $\mathbf{M} \in \mathbb{R}^{n\times d}$, the runtime of Power *is*

$$O\left(\frac{ndk}{\epsilon}\log\left(\frac{d}{\delta\epsilon}\right)\right).$$

Note that Lines 6 through 10 of Algorithm 78 exactly constitute a weight removal method of the form given in Lemma 295. Consequently, to use Lemma 295 it suffices to prove that the weights τ_i used in each iteration are safe with respect to the current set of weights, which we now demonstrate.

Lemma 299. *In each iteration t of Algorithm 78 until termination, $\tau^{(t)}$ is safe with respect to $w^{(t)}$.*

Proof. Throughout this proof, let $w := w^{(t)}$ and $\tau := \tau^{(t)}$. Furthermore, let $\mathbf{V}$, $\mathbf{\Sigma}$, and β correspond to the weights w at the iteration's start. We will inductively prove that τ is safe with respect to w,

Algorithm 78: $\mathsf{SIFT}(T, \delta)$

1 **Input:** $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumption 13, $\delta \in (0, 1)$;
2 $w^{(0)} \leftarrow \frac{1}{n}\mathbf{1}_T$, $t \leftarrow 0$, $\beta \leftarrow 1$, $k \leftarrow \lceil \frac{4}{\alpha} \rceil$;
3 $\mathbf{V} \leftarrow \mathsf{Power}(\mathrm{Cov}_{w^{(t)}}(T), k, 0.2, \frac{\delta}{2n})$;
4 $\mathbf{\Sigma} \leftarrow \mathbf{V}^\top \mathrm{Cov}_{w^{(t)}}(T)\mathbf{V}$;
5 **while** $\lambda_k(\mathbf{\Sigma}) \geq \frac{4}{\sqrt{\beta}}$ **do**
6 $\quad \tau_i^{(t)} \leftarrow \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (X_i - \mu_w(T)) \right\|_2^2$ for all $i \in T$;
7 $\quad w_i^{(t+1)} \leftarrow \left(1 - \frac{\tau_i^{(t)}}{\tau_{\max}^{(t)}}\right) w_i^{(t)}$ for all $i \in T$, where $\tau_{\max}^{(t)} := \max_{i \in T | w_i^{(t)} \neq 0} \tau_i^{(t)}$;
8 $\quad t \leftarrow t + 1$, $\beta \leftarrow \left\| w^{(t)} \right\|_1$;
9 $\quad \mathbf{V} \leftarrow \mathsf{Power}(\mathrm{Cov}_{w^{(t)}}(T), k, 0.2, \frac{\delta}{2n})$;
10 $\quad \mathbf{\Sigma} \leftarrow \mathbf{V}^\top \mathrm{Cov}_{w^{(t)}}(T)\mathbf{V}$;
11 **Return:**
$L := \{\mathbf{V}\mathbf{V}^\top X_i + \left(\mathbf{I} - \mathbf{V}\mathbf{V}^\top\right) \mu_{w^{(t)}}(T)$ where $i \in T$ is sampled uniformly at random$\}$, with list size $|L| = \lceil \frac{2}{\alpha} \log \frac{2}{\delta} \rceil$

which by applying Lemma 295 implies that at the start of the iteration, w is saturated (since clearly $w^{(0)}$ is saturated). We first compute the average score in S:

$$\begin{aligned}
\sum_{i \in S} \frac{w_i}{\|w_S\|_1} \tau_i &= \sum_{i \in S} \frac{w_i}{\|w_S\|_1} \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (X_i - \mu_w(T)) \right\|_2^2 \\
&= \sum_{i \in S} \frac{w_i}{\|w_S\|_1} \left(\left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (X_i - \mu_w(S)) \right\|_2^2 + \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (\mu_w(S) - \mu_w(T)) \right\|_2^2 \right) \\
&= \left\langle \mathbf{\Sigma}^{-1}, \mathbf{V}^\top \mathrm{Cov}_w(S) \mathbf{V} \right\rangle + \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (\mu_w(S) - \mu_w(T)) \right\|_2^2 \\
&\leq \left\langle \mathbf{\Sigma}^{-1}, \frac{\alpha}{\|w_S\|_1} \mathbf{I} \right\rangle + \frac{\|w\|_1}{\|w_S\|_1} \leq \frac{1}{4} \left\langle \sqrt{\beta}\mathbf{I}, \frac{1}{\sqrt{\beta}} \mathbf{I} \right\rangle + \frac{\sqrt{\beta}}{\alpha} \leq \frac{k}{2}.
\end{aligned}$$

The first three equalities follow by expanding definitions; the first inequality is by Lemmas 297 and 298, as well as the definition of $\mathbf{\Sigma}$. The second inequality is by using the definition of saturated weights (Definition 56) twice, which implies that $\|w_S\|_1 \geq \alpha\sqrt{\beta}$, as well as the exit condition in Line 5. The third inequality follows from the definition of k. Finally, we conclude that τ is indeed safe, since the average score in T is exactly k by design:

$$\begin{aligned}
\sum_{i \in T} \frac{w_i}{\|w\|_1} \tau_i &= \sum_{i \in T} \frac{w_i}{\|w\|_1} \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{V}^\top (X_i - \mu_w(T)) \right\|_2^2 \\
&= \left\langle \mathbf{\Sigma}^{-1}, \mathbf{V}^\top \left(\sum_{i \in T} \frac{w_i}{\|w\|_1} (X_i - \mu_w(T)) (X_i - \mu_w(T))^\top \right) \mathbf{V} \right\rangle = \left\langle \mathbf{\Sigma}^{-1}, \mathbf{\Sigma} \right\rangle = k.
\end{aligned}$$

□

Finally, we prove a runtime and correctness guarantee on Algorithm 78.

Theorem 89. *Under Assumption 13, with probability* $1-\delta$*, the output of Algorithm 78 satisfies*

$$\min_{\mu \in L} \|\mu - \mu^*\|_2^2 \leq \frac{22}{\alpha}.$$

The overall runtime of Algorithm 78 is

$$O\left(n^2 dk \log\left(\frac{d}{\delta}\right)\right).$$

Proof. We will show correctness and complexity of Algorithm 78 separately.

Complexity guarantee. It is clear that there are at most n iterations in Algorithm 78, since at least one weight is zeroed out in Line 7 each iteration. Further, the bottleneck operation in each iteration is clearly the complexity of Power, since an eigendecomposition of $\mathbf{\Sigma}$ takes time $O(k^3) = O(ndk)$. Since ϵ is a constant in Proposition 69 and $n = O(d^2)$, this yields the complexity bound. Using a union bound, with probability $1 - \frac{\delta}{2}$, the conclusion of Proposition 69 applies in every iteration; we will condition on this event for the remainder of the proof.

We finally note that the algorithm must terminate the while loop before removing all the weight. This is because throughout the algorithm since w is saturated (by Lemmas 295 and 299), $\|w\|_1 \geq \alpha^2$ holds directly by using Definition 56 and $\|w\|_1 \geq \|w_S\|_1$.

Correctness guarantee. As in Lemma 299, we let w denote the weights on the last iteration of the algorithm (after exiting on Line 12). Denote $\mathbf{P} := \mathbf{V}\mathbf{V}^\top$ and $Y_i := \mathbf{P}X_i$ for all $i \in T$. Since

$$\sum_{i \in S} \frac{1}{\alpha n} (Y_i - \mathbf{P}\mu^*)(Y_i - \mathbf{P}\mu^*)^\top = \mathbf{P}\left(\sum_{i \in S} \frac{1}{\alpha n}(X_i - \mu^*)(X_i - \mu^*)^\top\right)\mathbf{P} \preceq \mathbf{P},$$

by Assumption 13, the expectation of $\|Y_i - \mathbf{P}\mu^*\|_2^2$ for a uniformly random sample $i \in S$ is $\frac{4}{\alpha}$ by linearity of trace. Hence, by Markov with probability at least $\frac{1}{2}$ a sample from S has $\|Y_i - \mathbf{P}\mu^*\|_2^2 \leq \frac{8}{\alpha}$, so with probability at least $1 - \frac{\delta}{2}$, one of the random samples in L will have an X_i with $\|Y_i - \mathbf{P}\mu^*\|_2^2 \leq \frac{8}{\alpha}$. For this value of i, we expand via the Pythagorean theorem

$$\begin{aligned}\|(\mathbf{P}X_i + (\mathbf{I} - \mathbf{P})\mu_w(T)) - \mu^*\|_2^2 &= \|Y_i - \mathbf{P}\mu^*\|_2^2 + \|(\mathbf{I} - \mathbf{P})(\mu_w(T) - \mu^*)\|_2^2 \\ &\leq \frac{8}{\alpha} + \|(\mathbf{I} - \mathbf{P})(\mu_w(T) - \mu^*)\|_2^2.\end{aligned}$$

To bound this second term, we apply Lemma 296 on the set of points $\{(\mathbf{I}-\mathbf{P})X_i\}_{i\in T}$. This implies

$$\begin{aligned}\|(\mathbf{I}-\mathbf{P})\left(\mu_w(T)-\mu^*\right)\|_2^2 &\le \frac{2\beta}{\|w_S\|_1}\|(\mathbf{I}-\mathbf{P})\mathrm{Cov}_w(T)(\mathbf{I}-\mathbf{P})\|_{\mathrm{op}}+\frac{2\alpha}{\beta}\\ &\le \frac{12\sqrt{\beta}}{\|w_S\|_1}+\frac{2\alpha}{\beta}\le\frac{14}{\alpha}.\end{aligned}$$

Here, the last inequality used the definition of saturation, which also implies that $\beta\ge\alpha^2$. The second inequality used that the guarantees of Power and the termination condition imply that

$$\|(\mathbf{I}-\mathbf{P})\,\mathrm{Cov}_w(T)\,(\mathbf{I}-\mathbf{P})\|_{\mathrm{op}}\le 1.2\lambda_k(\mathrm{Cov}_w(T))\le 1.5\lambda_k(\mathbf{\Sigma})\le\frac{6}{\sqrt{\beta}}.$$

Here, we use that the eigenvalues of $\mathbf{\Sigma}$ are the same as those of $\mathbf{V}\mathbf{V}^\top\mathrm{Cov}_w(T)\mathbf{V}\mathbf{V}^\top$; this calculation is given in the correctness proof of Proposition 25. Combining the above bounds yields the conclusion. □

While Theorem 89 achieves the desired error guarantee (G.2), it unfortunately has a quadratic dependence on the sample complexity n, as well as a suboptimal list size by a factor of $O(\log\frac{1}{\delta})$. We address the latter issue with a post-processing step in Section G.4.1; regarding the former issue, Algorithm 78 will play a role in our final "fast" algorithm in the following Section G.3, which obtains a runtime with a linear dependence on n via more sophisticated weight removal.

G.3 Fast filtering in k dimensions under a diameter bound

We now give an algorithm, FastSIFT, with an improved dependence on the sample size n compared to the method SIFT developed in Section G.2. We use the following assumption in this section.

Assumption 14. *All data points in T lie in a Euclidean ball of radius R.*

We eventually show how to reduce the more general mean estimation problem to mean estimation on datasets satisfying Assumption 14 in Section G.4.2 to obtain our final algorithm. The primary goal of this section is to develop a method for quickly finding a "good" tuple $(\mathbf{B},w)$, defined as follows.

Definition 58 (Good tuple)**.** *We call $(\mathbf{B},w)$ "good" if it obeys the following conditions.*

1. *$\mathbf{B}\in\mathbb{R}^{d\times k'}$ has orthogonal columns, for some $k'=O(\frac{\log R}{\alpha})$, and $w\in\Delta^n$ is saturated.*
2. *Let $\mathbf{P}_{\mathbf{B}}:=\mathbf{B}\mathbf{B}^\top$. The restriction of $\mathrm{Cov}_w(T)$ to the complement of $\mathbf{P}_{\mathbf{B}}$, denoted by*

$$\mathrm{Cov}_w^{\mathbf{P}_{\mathbf{B}}^\perp}(T):=(\mathbf{I}-\mathbf{P}_{\mathbf{B}})\,\mathrm{Cov}_w(T)\,(\mathbf{I}-\mathbf{P}_{\mathbf{B}})$$

satisfies for a universal constant c,

$$\left\|\mathrm{Cov}_w^{\mathbf{P}_{\mathbf{B}}^{\perp}}(T)\right\|_{\mathrm{op}} \le \frac{c}{\sqrt{\|w\|_1}}.$$

Intuitively, a good tuple signifies that in all but $O(\frac{\log R}{\alpha})$ dimensions, we have learned the mean via the guarantee of Lemma 296. However, in the remaining dimensions we can simply run the algorithm of Section G.2, which obtains an additive $\mathrm{poly}(k)$ runtime dependence. We now make this rigorous.

Algorithm 79: $\mathsf{FastSIFT}(T, \delta, \mathsf{ProduceGoodTuple})$

1 **Input:** $T = T_{\mathrm{fast}} \cup T_{\mathrm{slow}} \subset \mathbb{R}^d$ with $|T_{\mathrm{fast}}| = n$ satisfying Assumptions 13 and 14, $|T_{\mathrm{slow}}| = O(\frac{\log R}{\alpha^2})$ satisfying Assumption 13 for a fixed $O(\frac{\log R}{\alpha})$-dimensional subspace, $\delta \in (0,1)$, subroutine $\mathsf{ProduceGoodTuple}$ which returns a good tuple with specified failure probability;
2 $(\mathbf{B}, w) \leftarrow \mathsf{ProduceGoodTuple}(T_{\mathrm{fast}}, \frac{\delta}{2})$;
3 $\mu_{\mathrm{fast}} \leftarrow (\mathbf{I} - \mathbf{B}\mathbf{B}^\top)\mu_w(T)$;
4 $L_{\mathrm{slow}} \leftarrow \mathsf{SIFT}(\{\mathbf{B}\mathbf{B}^\top X_i \mid X_i \in T_{\mathrm{slow}}\}, \frac{\delta}{2})$;
5 **Return:** $L \leftarrow \{\mu_{\mathrm{slow}} + \mu_{\mathrm{fast}} \mid \mu_{\mathrm{slow}} \in L_{\mathrm{slow}}\}$

Lemma 300. *With probability $1-\delta$, some $\hat{\mu} \in L$ outputted by Algorithm 79 satisfies*

$$\|\hat{\mu} - \mu^*\|_2^2 \le \frac{48 + 4c}{\alpha}.$$

The overall runtime of Algorithm 79 is the cost of running $\mathsf{ProduceGoodTuple}(T_{\mathrm{fast}}, \frac{\delta}{2})$ *plus*

$$O\left(\frac{d}{\alpha^3}\log(R)\log\left(\frac{dR}{\delta}\right) + \frac{1}{\alpha^6}\log^3(R)\log\left(\frac{d}{\delta}\right)\right) \text{ additional runtime overhead.}$$

Proof. By the proof of Theorem 89 and the second part of Definition 58, it is immediate that

$$\|(\mathbf{I} - \mathbf{P}_{\mathbf{B}})(\mu_w(T) - \mu^*)\|_2^2 \le \frac{2c+2}{\alpha}.$$

Moreover, since the size of T_{slow} is large enough for Proposition 68 to apply, it satisfies Assumption 13 on the k'-dimensional subspace whose projection matrix is $\mathbf{P}_{\mathbf{B}} = \mathbf{B}\mathbf{B}^\top$. Thus, Theorem 89 shows

$$\|\mu_{\mathrm{slow}} - \mathbf{P}_{\mathbf{B}}\mu^*\|_2^2 \le \frac{22}{\alpha} \text{ for some } \mu_{\mathrm{slow}} \in L_{\mathrm{slow}}.$$

Combining these two bounds and the Pythagorean theorem yields the correctness guarantee. For the runtime overhead guarantee, it is clear the bottleneck operation is Line 4 since Line 3 can be implemented in time $O(\frac{d}{\alpha}\log R)$. For Line 4, we run Algorithm 78 entirely in the coordinate system

of the columns of $\mathbf{B}$, which is isomorphic to $\mathbb{R}^{k'}$, and then left-multiply the resulting list by $\mathbf{B}$. Forming the input set $\{\mathbf{B}^\top X_i \mid X_i \in T_{\text{slow}}\}$ takes time $O(|T_{\text{slow}}|k'd) = O(\frac{d}{\alpha^3}\log^2 R)$; multiplying the resulting output list by $\mathbf{B}$ cannot be the dominant cost by more than a $\log\frac{1}{\delta}$ factor. □

Here, we note that because we take $n = \Omega(d\alpha^{-1}) = \Omega(\alpha^{-2})$ in accordance with Proposition 68, the cost of $O(d\alpha^{-3}\log R\log\frac{dR}{\delta})$ incurred by Lemma 300 is no more than the cost of logarithmically many k-PCAs on the original dataset. Regarding the separation of the original dataset into T_{fast} and T_{slow}, which appropriately satisfy Assumption 13, we make the following comment.

Remark 14. *We can form a partitioned dataset $T = T_{\text{fast}} \cup T_{\text{slow}}$ of the form required by Algorithm 79 by independently drawing n samples to form T_{fast}, $O(\frac{\log R}{\alpha^2})$ samples to form T_{slow}, and applying Assumption 13 to T_{fast} and the projection of T_{slow} into a k'-dimensional subspace. Up to a $\log\frac{1}{\delta}$ factor in the sample complexity (for error probabilities which are smaller than $\exp(-\Omega(\alpha^{-1}))$), these are valid applications of Assumption 13 because of independence; in particular, the draws T_{slow} are independent of the k'-dimensional subspace learned by running* ProduceGoodTuple *on T_{fast}, which only depends on randomness used in Step 2 of* FastSIFT*.*

We now state our strategy for the implementation of ProduceGoodTuple. Roughly speaking, ProduceGoodTuple is a composition of three subroutines at different levels, named BicriteriaFilter, DecreaseKFNorm, and KFMMW. Each subroutine is associated with one or more potential functions which show that the subroutine "one level down" is called $O(\log d)$ times.

1. ProduceGoodTuple iteratively calls BicriteriaFilter, an algorithm which takes as input saturated weights w and either produces saturated weights $\|w'\|_1 \le \frac{1}{2}\|w\|_1$, or a good tuple.

2. BicriteriaFilter iteratively calls DecreaseKFNorm, an algorithm which takes as input saturated weights w and maintains an updated set of orthogonal vectors $\mathbf{B}$. Each call to DecreaseKFNorm either (1) halves the ℓ_1 norm of w, (2) halves the Ky Fan k norm of the covariance matrix, or (3) decreases the operator norm of the covariance matrix by a constant factor and adds k vectors to $\mathbf{B}$, for some $k = \Theta(\frac{1}{\alpha})$.

3. DecreaseKFNorm is based on a "win-win-win" analysis of the fine-grained guarantees of a Ky Fan norm matrix multiplicative weights procedure, developed in Section 7.3. We will show that in $O(\log d)$ iterations of KFMMW, either the Ky Fan k norm has halved, or one of the other two "exit conditions" required by DecreaseKFNorm has been certifiably met.

Given the guarantees of DecreaseKFNorm, correctness of ProduceGoodTuple and BicriteriaFilter follow straightforwardly. Thus, in Section G.3.1, we state and prove a performance guarantee on DecreaseKFNorm, which we use to give a simple analysis of ProduceGoodTuple in Section G.3.2. Combining our analysis of ProduceGoodTuple with Lemma 300 gives the main export from this section. Finally, we note that in the following development of ProduceGoodTuple and its subroutines, we will overload the input set T to be T_{fast} in Algorithm 79, because it is the input to ProduceGoodTuple.

G.3.1 Analysis of DecreaseKFNorm

We first state a guarantee for ApproxKFMMW as Proposition 70, which is a computationally efficient variant of KFMMW (these methods are both given and analyzed in Section 7.3). Proposition 70 is a restatement of Corollary 26 and Lemma 95 from Section 7.3 with $\Delta = \frac{1}{200}$.

Proposition 70. *There is an algorithm,* ApproxKFMMW *(Algorithm 28), which takes as input a sequence of matrices $\{\mathbf{G}_t\}_{t\geq 0} \subset \mathbb{S}^d_{\succeq 0}$ each in the form $\mathbf{M}_t^\top \mathbf{M}_t$ for $\mathbf{M}_t \in \mathbb{R}^{n\times d}$ for explicitly given $\mathbf{M}_t$, and $k \in [d]$. Suppose that the matrices $\{\mathbf{G}_t\}_{t\geq 0}$ are weakly decreasing in Loewner order, and let $\eta \leq \frac{1}{2\|\mathbf{G}_0\|_{\mathrm{op}}}$. For any $N \geq 1$, with probability $1-\delta'$,* ApproxKFMMW *defines a sequence of matrices $\{\widehat{\mathbf{Y}}_t\}_{0\leq t<N}$, where $\widehat{\mathbf{Y}}_t$ only depends on $\{\mathbf{G}_s\}_{0\leq s<t}$, such that*

$$\|\mathbf{G}_N\|_k \leq \frac{2}{T}\sum_{t=0}^{N-1}\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle + \frac{k\log d}{\eta N} + \frac{k}{200\eta}.$$

Each $\widehat{\mathbf{Y}}_t$ satisfies $\left\|\widehat{\mathbf{Y}}_t\right\|_{\mathrm{op}} \leq 1.01$ and $\left\|\widehat{\mathbf{Y}}_t\right\|_{\mathrm{tr}} \leq 1.01k$. The cost of the algorithm is

$$O\left(ndkN^2\log^2\left(\frac{dN}{\delta'}\right)\right).$$

Furthermore, for any set of n fixed vectors $\{v_i\}_{i\in[n]} \subset \mathbb{R}^d$ and any iteration t, 1.05-approximations to all $v_i^\top \widehat{\mathbf{Y}}_t v_i$ can be computed in time

$$O\left(ndN\log\left(\frac{nd}{\delta'}\right)\right) \text{ with probability at least } 1-\delta'.$$

We are now ready to state the algorithm DecreaseKFNorm as Algorithm 80. At a high level, the goal of DecreaseKFNorm is to implement Proposition 70 in a way so that each of the inner products $\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t\right\rangle$ is sufficiently small, via decreasing weights defined in terms of the matrix $\widehat{\mathbf{Y}}_t$. We will be able to successfully do this as long as the ℓ_1 norm of the weight remains stable, and the top eigenvalue of the covariance matrix is not too much larger than the k^{th} largest. When either of these conditions fail, we will exit the algorithm via a different termination condition.

The first step in the analysis of Algorithm 80 is to guarantee that any time a weight removal procedure is performed, it is with respect to safe scores, and hence the weights remain saturated throughout the course of the algorithm. We give this proof of safe weight removal as Lemma 301, and then an overall correctness and runtime guarantee in Proposition 71.

Lemma 301. *Throughout the course of Algorithm 80, any time weight removal is performed in Line 12, it is with respect to safe scores, and thus $w^{(t)}$ is saturated for all $0 \leq t < N$.*

Proof. With probability $1-\delta$, all executions of Lines 5 and 10 throughout the algorithm succeed, so we will condition on this event for the remainder of this proof. We also note that in any iteration

Algorithm 80: DecreaseKFNorm(T, w, γ, δ)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumptions 13 and 14, saturated w, $\gamma \leftarrow$ 1.05-approximation to $\|\mathrm{Cov}_w(T)\|_k$ with probability at least $1 - \frac{\delta}{3(N+1)}$ for $k := \lceil \frac{612}{\alpha} \rceil$ satisfying $\gamma \geq \frac{110k}{\sqrt{\|w\|_1}}$, $\delta \in (0,1)$;

Output: Saturated w', satisfying one of the following possibilities with probability $\geq 1 - \delta$: w' has $\|w'\|_1 \leq \frac{1}{2}\|w\|_1$ (marked "Case 1"), $\mathbf{V} \in \mathbb{R}^{d \times k}$ is also outputted, and $\left\|\mathrm{Cov}_{w'}^{\mathbf{P}_{\mathbf{V}}^{\perp}}(T)\right\|_{\mathrm{op}} \leq \frac{2}{3}\|\mathrm{Cov}_w(T)\|_{\mathrm{op}}$ (marked "Case 2"), w' has $\|\mathrm{Cov}_{w'}(T)\|_k \leq \frac{1}{2}\|\mathrm{Cov}_w(T)\|_k$ (marked "Case 3");

$N \leftarrow \lceil 425 \log d \rceil$, $w^{(0)} \leftarrow w$, $\bar{\beta} \leftarrow \left\|w^{(0)}\right\|_1$, $\eta \leftarrow \frac{1}{2.1\rho}$, where ρ is a 1.05-approximation of $\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{\mathrm{op}}$ with probability at least $1 - \frac{\delta}{3(N+1)}$;

for $0 \leq t < N$ **do**

  $\mathbf{V} \leftarrow$ Power$(\mathrm{Cov}_{w^{(t)}}(T), k, 0.05, \frac{\delta}{3(N+1)})$;

  $\tilde{\lambda}_1 \leftarrow \langle \mathbf{V}_{:1}, \mathrm{Cov}_{w^{(t)}}(T)\mathbf{V}_{:1} \rangle$, $\tilde{\lambda}_k \leftarrow \langle \mathbf{V}_{:k}, \mathrm{Cov}_{w^{(t)}}(T)\mathbf{V}_{:k} \rangle$;

  if $\tilde{\lambda}_1 \geq 3.5\tilde{\lambda}_k$ **then**

    Return: $(w^{(t)}, \mathbf{V}$, "Case 2");

  $\tau_i^{(t)} \leftarrow$ 1.05-approximation to $\left\langle (X_i - \mu_{w^{(t)}}(T)), \widehat{\mathbf{Y}}_t (X_i - \mu_{w^{(t)}}(T)) \right\rangle$ for all $i \in T$, with probability at least $1 - \frac{\delta}{3(N+1)}$;

  if $\sum_{i \in T} w_i^{(t)} \tau_i^{(t)} > \frac{\gamma \bar{\beta}}{12}$ **then**

    $w^{(t+1)} \leftarrow w^{(t,K)}$, where $K \leftarrow$ smallest natural number such that

$$\text{either } \left\|w^{(t,K)}\right\|_1 \leq \frac{\bar{\beta}}{2}, \text{ or } \sum_{i \in T} w_i^{(t,K)} \tau_i^{(t)} \leq \frac{\gamma \bar{\beta}}{12}, \quad \text{where } w_i^{(t,K)} := \left(1 - \frac{\tau_i^{(t)}}{\tau_{\max}^{(t)}}\right)^K w_i^{(t)}, \text{ and } \tau_{\max}^{(t)} := \max_{i \in T | w_i^{(t)} \neq 0} \tau_i^{(t)} \tag{G.5}$$

    ;

    if $\left\|w^{(t+1)}\right\|_1 \leq \frac{\bar{\beta}}{2}$ **then**

      Return: $(w^{(t+1)}$, "Case 1");

    else

      $w^{(t+1)} \leftarrow w^{(t)}$;

  Feed $\mathbf{G}_t \leftarrow \widetilde{\mathrm{Cov}}_{w^{(t+1)}}(T)$ into the routine ApproxKFMMW with step size η and $\delta' \leftarrow \frac{\delta}{3}$;

Return: $(w^{(N)}$, "Case 3");

t where Line 12 is reached, Line 7 did not pass, and thus

$$\lambda_1\left(\mathrm{Cov}_{w^{(t)}}(T)\right) \leq 1.05\tilde{\lambda}_1 < 3.675\tilde{\lambda}_k \leq 4\lambda_k\left(\mathrm{Cov}_{w^{(t)}}(T)\right) \implies \|\mathrm{Cov}_{w^{(t)}}(T)\|_k \geq \frac{k}{4}\|\mathrm{Cov}_{w^{(t)}}(T)\|_{\mathrm{op}} \tag{G.6}$$

by the guarantees of Power in Proposition 69. Consider now a single iteration $0 \leq t < N$, and

suppose inductively that $w^{(t)}$ is saturated before Line 12 is executed. In every round of weight removal $0 \le \ell < K$, assuming that the ℓ_1 norm has not halved, we can lower bound the average score in T by the definition of K:

$$\sum_{i \in T} \frac{w_i^{(t,\ell)}}{\left\|w^{(t,\ell)}\right\|_1} \tau_i^{(t)} \ge \frac{1}{\bar{\beta}} \sum_{i \in T} w_i^{(t,\ell)} \tau_i^{(t)} \ge \frac{\gamma}{12}.$$

Hence, to prove that the scores are safe in iteration ℓ, it suffices to show that the average score in S is at most $\frac{\gamma}{24}$. Because the weights $w^{(t,\ell)}$ are monotone in ℓ, and the ℓ_1 norm of $w_S^{(t,\ell)}$ inductively does not change by more than a factor of $\sqrt{2}$ by the following Lemma 301, it suffices to show that

$$\sum_{i \in S} \frac{w_i^{(t,0)}}{\left\|w_S^{(t,0)}\right\|_1} \tau_i^{(t)} \le \frac{\gamma}{34} \implies \sum_{i \in S} \frac{w_i^{(t,\ell)}}{\left\|w_S^{(t,\ell)}\right\|_1} \tau_i^{(t)} \le \frac{\gamma\sqrt{2}}{34} < \frac{\gamma}{24}.$$

We now prove this bound on the average score in S with respect to $w^{(t,0)} = w^{(t)}$, which will conclude the proof. To see this bound, we have

$$\begin{aligned}
\sum_{i \in S} \frac{w_i^{(t)}}{\left\|w_S^{(t)}\right\|_1} \tau_i^{(t)} &\le 1.05 \left\langle \widehat{\mathbf{Y}}_t, \sum_{i \in S} \frac{w_i^{(t)}}{\left\|w_S^{(t)}\right\|_1} \left(X_i - \mu_{w^{(t)}}(T)\right) \left(X_i - \mu_{w^{(t)}}(T)\right)^\top \right\rangle \\
&= 1.05 \left\langle \widehat{\mathbf{Y}}_t, \mathrm{Cov}_{w^{(t)}}(S) \right\rangle + 1.05 \left\langle \widehat{\mathbf{Y}}_t, \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right) \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right)^\top \right\rangle \\
&\le 1.07k \left\|\mathrm{Cov}_{w^{(t)}}(S)\right\|_{\mathrm{op}} + 1.07 \left\| \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right) \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right)^\top \right\|_{\mathrm{op}} \\
&\le \frac{1.07k\alpha}{\left\|w_S^{(t)}\right\|_1} + \frac{9\gamma}{k\alpha} \le \frac{1.6k}{\sqrt{\bar{\beta}}} + \frac{9\gamma}{k\alpha} \le \frac{\gamma}{34}.
\end{aligned}$$

Here, the first inequality is by the approximation guarantees on the scores $\tau_i^{(t)}$. The second inequality used matrix Hölder twice, as well as trace and operator norm bounds on $\widehat{\mathbf{Y}}_t$ due to Proposition 70, and finally the fact that the trace and operator norm agree for any rank-1 matrix. The fourth inequality is by the helper Lemma 301 and saturation of $w^{(0)}$, and the fifth is by our choices of k and lower bound on $\gamma \ge \frac{110k}{\sqrt{\bar{\beta}}}$. The third inequality used Lemmas 297 and 298, the latter of which implies

$$\begin{aligned}
\left\| \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right) \left(\mu_{w^{(t)}}(S) - \mu_{w^{(t)}}(T)\right)^\top \right\|_{\mathrm{op}} &\le \frac{\left\|w^{(t)}\right\|_1}{\left\|w_S^{(t)}\right\|_1} \left\|\mathrm{Cov}_{w^{(t)}}(T)\right\|_{\mathrm{op}} \\
&\le \frac{1}{\alpha} \cdot \frac{4}{k} \left\|\mathrm{Cov}_{w^{(t)}}(T)\right\|_k \le \frac{8.4\gamma}{k\alpha}.
\end{aligned}$$

The second inequality used our assumption (G.6), and the last used that $\widetilde{\mathrm{Cov}}_{w^{(t)}}(T)$ is monotonically decreasing in the Loewner order, and thus since until termination, the normalization factor $\left\|w^{(t)}\right\|_1$

does not change by more than a factor of two, and γ is a 1.05-approximation to $\|\mathrm{Cov}_{w^{(0)}}(T)\|_k$,

$$\|\mathrm{Cov}_{w^{(t)}}(T)\|_k = \frac{1}{\left\|w^{(t)}\right\|_1}\left\|\widetilde{\mathrm{Cov}}_{w^{(t)}}(T)\right\|_k \le \frac{2}{\bar{\beta}}\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_k \le 2.1\gamma.$$

□

In proving Lemma 301, we used the following helper lemma.

Lemma 302. *Consider any algorithm of the form in Lemma 295. Suppose in some iteration t, $\left\|w^{(t)}\right\|_1 \ge \frac{1}{2}\left\|w^{(0)}\right\|_1$. Then, $\left\|w_S^{(t)}\right\|_1 \ge \frac{1}{\sqrt{2}}\left\|w_S^{(0)}\right\|_1$.*

Proof. This is immediate from telescoping (G.4), which was used in the proof of Lemma 295. □

Finally, we prove overall correctness of Algorithm 80.

Proposition 71. *Algorithm 80 succeeds with probability at least $1-\delta$, in the sense that each of Cases 1-3 returns correctly. The overall complexity is bounded by*

$$O\left(ndk\log^2(d)\log^2\left(\frac{dR}{\delta}\right)\right).$$

Proof. We will show correctness and complexity of Algorithm 80 separately.

Correctness guarantee. As argued in the proof of Lemma 301, with probability $1-\delta$ every weight removal is safe, so Lemma 301 shows that $w^{(t)}$ is saturated throughout the algorithm. By a union bound, we also assume that all approximations are correct in the remainder of the proof. It is obvious that if the algorithm terminates in Line 14, the requirement of Case 1 is met. If the algorithm terminates in Line 8, the guarantees of Power (Proposition 69) imply that

$$\begin{aligned}\lambda_1\left(\mathrm{Cov}_{w^{(t)}}(T)\right) &\ge \frac{1}{1.05}\tilde{\lambda}_1 \ge \frac{3.5}{1.05}\tilde{\lambda}_k \ge \frac{3.5}{1.05^2}\lambda_k\left(\mathrm{Cov}_{w^{(t)}}(T)\right)\\ &\ge \frac{3.5}{1.05^3}\left\|(\mathbf{I}-\mathbf{V}\mathbf{V}^\top)\mathrm{Cov}_{w^{(t)}}(T)(\mathbf{I}-\mathbf{V}\mathbf{V}^\top)\right\|_{\mathrm{op}} \ge 3\left\|\mathrm{Cov}^{\mathbf{P}_{\mathbf{V}}^\perp}_{w^{(t)}}(T)\right\|_{\mathrm{op}}.\end{aligned} \tag{G.7}$$

However, since the algorithm did not terminate on Line 14 in the previous iteration, we also have

$$\lambda_1\left(\mathrm{Cov}_{w^{(t)}}(T)\right) = \frac{1}{\left\|w^{(t)}\right\|_1}\lambda_1\left(\widetilde{\mathrm{Cov}}_{w^{(t)}}(T)\right) \le \frac{2}{\bar{\beta}}\lambda_1\left(\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right) = 2\lambda_1\left(\mathrm{Cov}_{w^{(0)}}(T)\right).$$

Combining the above two calculations gives the correctness proof for Case 2, as

$$\left\|\mathrm{Cov}^{\mathbf{P}_{\mathbf{V}}^\perp}_{w^{(t)}}(T)\right\|_{\mathrm{op}} \le \frac{1}{3}\lambda_1\left(\mathrm{Cov}_{w^{(t)}}(T)\right) \le \frac{2}{3}\lambda_1\left(\mathrm{Cov}_{w^{(0)}}(T)\right).$$

Finally, we show correctness in Case 3, where N iterations of the algorithm have passed without terminating on either of Lines 8 (which halves operator norm) or 14 (which halves weight). In

this case, we apply Proposition 70, which is valid since the $\mathbf{G}_t$ are monotonically decreasing, and $\eta\mathbf{G}_0 \preceq \frac{1}{2}\mathbf{I}$ by the approximation guarantee on ρ. Here, we also note that all our matrices $\mathbf{G}_t$ are covariance matrices with known weights, so they can be expressed in the form $\mathbf{M}_t^\top \mathbf{M}_t$ for explicitly given $\mathbf{M}_t \in \mathbb{R}^{n\times d}$. Proposition 70 additionally requires a bound on each $\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t \right\rangle$; to this end,

$$\begin{aligned}\left\langle \mathbf{G}_t, \widehat{\mathbf{Y}}_t \right\rangle &= \sum_{i\in T} w_i^{(t+1)} \left\langle (X_i - \mu_{w^{(t+1)}}(T)), \widehat{\mathbf{Y}}_t (X_i - \mu_{w^{(t+1)}}(T)) \right\rangle \\ &\le \sum_{i\in T} w_i^{(t+1)} \left\langle (X_i - \mu_{w^{(t)}}(T)), \widehat{\mathbf{Y}}_t (X_i - \mu_{w^{(t)}}(T)) \right\rangle \le 1.05 \sum_{i\in T} w_i^{(t+1)} \tau_i^{(t)} \le \frac{1.05\gamma\bar{\beta}}{12}.\end{aligned}$$

In the first inequality, we used Fact 39; in the second, we used the assumption on the scores $\tau^{(t)}$; and in the third, we used the second guarantee in (G.5) since we did not terminate on Line 14. Now, applying this bound in every iteration $0 \le t < N$ in Proposition 70, and defining $\mathbf{G}_N = \mathbf{G}_{N-1}$,

$$\begin{aligned}\|\mathbf{G}_N\|_k &\le \frac{1.05\gamma\bar{\beta}}{6} + \frac{2.1k\rho\log d}{N} + \frac{2.1k\rho}{200} \\ &\le \frac{1.05\gamma\bar{\beta}}{6} + \frac{2.21k\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{\mathrm{op}}\log d}{N} + \frac{2.21k\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{\mathrm{op}}}{200} \\ &\le \frac{1.05\gamma\bar{\beta}}{6} + \frac{9\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{k}\log d}{N} + \frac{9\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{k}}{200} \\ &\le \frac{1.05^2\left\|\mathrm{Cov}_{w^{(0)}}(T)\right\|_k\bar{\beta}}{6} + \frac{9\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{k}\log d}{N} + \frac{9\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_{k}}{200}.\end{aligned}$$

The first inequality was by Proposition 70 and the definition of η; the second was by the approximation guarantee on ρ; the third was by the fact that the first iteration did not terminate on Line 8, so we can apply the bound (G.6); and the fourth was by the definition of γ. Next, dividing both sides by $\bar{\beta}$ and using that termination on Line 14 has not occurred,

$$\begin{aligned}\frac{1}{2}\left\|\mathrm{Cov}_{w^{(N)}}(T)\right\|_k &= \frac{1}{2\left\|w^{(N)}\right\|_1}\left\|\widetilde{\mathrm{Cov}}_{w^{(N)}}(T)\right\|_k \\ &\le \frac{1}{\bar{\beta}}\|\mathbf{G}_N\|_k \le \left\|\mathrm{Cov}_{w^{(0)}}(T)\right\|_k\left(\frac{1.05^2}{6} + \frac{9\log d}{N} + \frac{9}{200}\right).\end{aligned}$$

Here, we used that $\frac{1}{\bar{\beta}}\left\|\widetilde{\mathrm{Cov}}_{w^{(0)}}(T)\right\|_k = \left\|\mathrm{Cov}_{w^{(0)}}(T)\right\|_k$ twice, by definition of $\bar{\beta}$. Rearranging and using the definition of $N \ge 425\log d$ then yields correctness of Case 3.

Complexity guarantee. For $N = O(\log d)$, the total cost of running ApproxKFMMW is

$$O\left(ndk\log^2(d)\log^2\left(\frac{d}{\delta}\right)\right),$$

as given by Proposition 70. It is straightforward to check that the costs of Lines 5 and 10, given by

Propositions 69 and 70, do not dominate this. Finally, since the cost of checking (G.5) for a value of K is linear in n, it suffices to provide an upper bound on K and then binary search. For this, we have

$$\sum_{i\in T} w_i^{(t,K)}\tau_i^{(t)} \le \sum_{i\in T}\exp\left(-\frac{K\tau_i^{(t)}}{\tau_{\max}^{(t)}}\right) w_i^{(t)}\tau_i^{(t)} \le \frac{1}{eK}\sum_{i\in T} w_i^{(t)}\tau_{\max}^{(t)} \le \frac{\tau_{\max}^{(t)}}{eK}.$$

Here, the first inequality used the definition of $w_i^{(t,K)}$, the second used that $x\exp(-Cx) \le \frac{1}{eC}$ for all nonnegative x, where we chose $C = \frac{K}{\tau_{\max}^{(t)}}$, and the third used $w^{(t)} \in \Delta^n$. Since the definition of saturated weights implies that $\sqrt{\bar{\beta}} \ge \alpha$, it follows that the threshold in Line 11 satisfies

$$\frac{\gamma\bar{\beta}}{12} \ge \frac{110k\alpha}{12} \ge 5000.$$

Also, Assumption 14 and $\left\|\widehat{\mathbf{Y}}_t\right\|_{\text{op}} \le 1.01$ imply that all scores are bounded by $1.01R^2$, so we conclude $K \le R^2$. Thus, the complexity of the binary search is $O(n\log R)$ and does not dominate. □

G.3.2 Analysis of ProduceGoodTuple

At this point, the statements and analyses of both BicriteriaFilter and ProduceGoodTuple are straightforward, as we have done most of the heavy lifting in proving Proposition 71. We state both here and prove their correctness and a runtime guarantee in Proposition 72.

Proposition 72. *Algorithm 82,* ProduceGoodTuple*, correctly outputs a good tuple with probability at least* $1-\delta$*. Its overall complexity is*

$$O\left(\frac{nd}{\alpha}\log^2(d)\log^2\left(\frac{dR}{\delta}\right)\log(R)\log\left(\frac{1}{\alpha}\right)\right).$$

Proof. We will show correctness and complexity of Algorithm 82 separately.

Correctness guarantee. We first claim that if BicriteriaFilter meets its specifications, then so does ProduceGoodTuple. This is since every time BicriteriaFilter returns in Case 1, the ℓ_1 norm of w is halved, but it can never be smaller than α^2 since w is always saturated, so Case 1 occurs $\le 2\log\frac{1}{\alpha}$ times. Finally, note that Case 2 of BicriteriaFilter indeed constitutes a good tuple, with $c = 128$.

It remains to prove that BicriteriaFilter meets its specifications. We first claim that the while loop of Lines 5-23 is not run more than M times. To see this, whenever DecreaseKFNorm returns in Case 1, the loop immediately terminates, so it suffices to bound the number of times DecreaseKFNorm returns in Case 2 or Case 3 before exiting on Line 13. Observe that every time Case 2 occurs, the operator norm of $\text{Cov}_w(T)$ is decreased by $\frac{1}{3}$, but by Assumption 14 it is bounded by R^2 initially, and as soon as it is smaller than 100, then the algorithm will exit on Line 13. Thus, the number of times Case 2 occurs is at most $3\log(\frac{R^2}{100})$; similarly, Case 3 occurs at most $2\log(\frac{R^2}{100})$ times since it halves the Ky Fan norm each time. Combining these yields the claimed bound of M loops.

Algorithm 81: BicriteriaFilter(T, δ, w)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumptions 13 and 14, $\delta \in (0,1)$, saturated w
Output: Saturated w', satisfying one of the following possibilities with probability $\geq 1-\delta$:

1. w' has $\|w'\|_1 \leq \frac{1}{2}\|w\|_1$ (marked "Case 1")
2. $\mathbf{B} \in \mathbb{R}^{d \times k'}$ is also outputted, for $k' = O(\frac{\log R}{\alpha})$, and

$$\left\|\mathrm{Cov}_{w'}^{\mathbf{P}_{\mathbf{B}}^{\perp}}(T)\right\|_{\mathrm{op}} \leq \frac{128}{\sqrt{\|w'\|_1}} \text{ (marked "Case 2")}$$

;
$\mathbf{B} \leftarrow []$, $k \leftarrow \lceil \frac{612}{\alpha} \rceil$, $\bar{\beta} \leftarrow \|w\|_1$;
$\delta' \leftarrow \frac{\delta}{M}$, for $M = 5\log(\frac{R^2}{100})$;
while ***true*** **do**
  if $\|w\|_1 \leq \frac{1}{2}\bar{\beta}$ **then**
    Return: $(w$, "Case 1");
  $T \leftarrow$ projection of T into orthogonal complement of $\mathbf{B}\mathbf{B}^\top$;
  $\gamma \leftarrow$ 1.05-approximation to $\|\mathrm{Cov}_w(T)\|_k$ with probability $\geq 1 - \frac{\delta'}{3(N+1)}$, for $N = \lceil 150 \log d \rceil$;
  if $\gamma < \frac{110k}{\sqrt{\|w\|_1}}$ **then**
    Append the columns of $\mathsf{Power}(\mathrm{Cov}_w(T), k, 0.05, \frac{\delta'}{3(N+1)})$ to $\mathbf{B}$;
    Return: $(w, \mathbf{B}$, "Case 2");
  if DecreaseKFNorm(T, w, γ, δ') *returns "Case 1"* **then**
    Return: (DecreaseKFNorm(T, w, γ, δ'), "Case 1")
  else if DecreaseKFNorm(T, w, γ, δ') *returns "Case 2"* **then**
    $(w, \mathbf{V}) \leftarrow$ DecreaseKFNorm(T, w, γ, δ');
    Append the columns of $\mathbf{V}$ to $\mathbf{B}$;
  else
    $w \leftarrow$ DecreaseKFNorm(T, w, γ, δ');

Algorithm 82: ProduceGoodTuple(T, δ)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumptions 13 and 14, $\delta \in (0,1)$;
Output: Good tuple $(\mathbf{B}, w)$ (cf. Definition 58) with probability $\geq 1-\delta$;
$w \leftarrow \frac{1}{n}\mathbf{1}$;
while ***true*** **do**
  if BicriteriaFilter $\left(T, \frac{\delta}{2\log\frac{1}{\alpha}}, w\right)$ *returns "Case 1"* **then**
    $w \leftarrow$ BicriteriaFilter $\left(T, \frac{\delta}{2\log\frac{1}{\alpha}}, w\right)$
  ;
  else
    Return: BicriteriaFilter $\left(T, \frac{\delta}{2\log\frac{1}{\alpha}}, w\right)$;

Thus, the failure probability of BicriteriaFilter is met; it remains to prove that in each case, it returns correctly. If the algorithm returns on Line 7, this is clear. If the algorithm returns on Line 16, note that its input w has $\|w\|_1 \le \bar{\beta}$ by monotonicity of filtering, so it must be that the output of DecreaseKFNorm has ℓ_1 norm at most $\frac{1}{2}\bar{\beta}$ by Case 1 of DecreaseKFNorm. The only other place the algorithm can return is in Line 13. However, in this case it is clear that $\mathbf{B}$ has at most $k \cdot (3\log(\frac{R^2}{100}) + 1) = O(\frac{\log R}{\alpha})$ columns, since every time Line 18 is executed only k columns are appended, and we earlier bounded the number of times Line 18 can occur. Finally, by combining the definition of γ, the fact that we always project T into the orthogonal complement of $\mathbf{B}\mathbf{B}^\top$ in Line 9, and the fact that Proposition 69 implies that $\lambda_{k+1}(\mathrm{Cov}_w(T)) \le (1.05)^2\frac{\gamma}{k}$ (see e.g. the calculation (G.7)), we see that when $(w, \mathbf{B})$ is returned,

$$\left\|\mathrm{Cov}_w^{\mathbf{P}_{\mathbf{B}}^\perp}(T)\right\|_{\mathrm{op}} \le \frac{(1.05)^3\gamma}{k} \le \frac{128}{\sqrt{\|w\|_1}}.$$

In the above equation, we overload T to mean the original dataset (rather than after projection in Line 9). This proves correctness of BicriteriaFilter in all cases. Finally, we remark that all parts of DecreaseKFNorm operate correctly after the projection in Line 9. The only place this may cause difficulty is in dependences on smallest eigenvalues in implementing ApproxKFMMW, because the gain matrices are not full rank. However, it is straightforward to check that the guarantees of the subroutines ApproxProject and Power as given in Section 7.3 will depend on the smallest eigenvalues of gain matrices restricted to $\mathrm{Span}(\mathbf{I} - \mathbf{B}\mathbf{B}^\top)$, if all operations are performed in this space.

Complexity guarantee. By our earlier analysis, ProduceGoodTuple incurs a multiplicative $O(\log \frac{1}{\alpha})$ overhead on the cost of BicriteriaFilter, so it suffices to understand this latter complexity. The dominant cost is clearly the (at most M) calls to DecreaseKFNorm, and the projection steps in Line 9. Line 9 involves orthogonalizing each of n vectors against $O(k \log R)$ vectors in d dimensions, so its complexity is $O(ndk \log R \cdot M)$, which does not dominate. The overall cost bound follows from combining Proposition 71 with a multiplicative $O(M \log \frac{1}{\alpha})$ overhead factor. □

By combining Lemma 300 with Proposition 72, we have the following guarantee on FastSIFT.

Corollary 66. *With probability $1 - \delta$, some $\hat{\mu} \in L$ outputted by Algorithm 79 satisfies*

$$\|\hat{\mu} - \mu^*\|_2^2 \le \frac{560}{\alpha}.$$

The overall runtime of Algorithm 79 is

$$O\left(\frac{nd}{\alpha}\log^2(d)\log^2\left(\frac{dR}{\delta}\right)\log(R)\log\left(\frac{1}{\alpha}\right) + \frac{1}{\alpha^6}\log^3(R)\log\left(\frac{dR}{\delta}\right)\right).$$

G.4 Cleanup

In this section, we give implementations of pre-processing and post-processing procedures on the dataset which will be used in attaining our final guarantees. In particular, Section G.4.1 shows how to reduce the size of our final output list, and Section G.4.2 shows how to naïvely cluster the dataset to have diameter polynomially bounded in problem parameters. Finally, we put all the pieces together in giving our final result on list decodable mean estimation in Section G.4.3, as well as a variant on this procedure which obtains a slight runtime-accuracy tradeoff, in Section G.4.4.

G.4.1 Merging candidate means

We give a simple greedy algorithm for taking the output of Algorithm 79 (FastSIFT) and reducing its size to be $O(\frac{1}{\alpha})$, without affecting the guarantee (G.2) by more than a constant factor. The algorithm and analysis bear some resemblance to the strategy in [179], but we include it for completeness. In this section, denote $k := \lceil \frac{4}{\alpha} \rceil$ as in Algorithm 78. We recall from the description of SIFT that the output L of FastSIFT has the property that elementwise, all $\hat{\mu} \in L$ are of the form

$$\mu_{\text{fixed}} + \mathbf{V}\mathbf{V}^\top \mathbf{B}\mathbf{B}^\top X_i = \mu_{\text{fixed}} + \mathbf{P}X_i, \text{ where } X_i \in T_{\text{slow}}, \ \mathbf{P} := \mathbf{V}\mathbf{V}^\top, \tag{G.8}$$

since columns of $\mathbf{V} \in \mathbb{R}^{d \times k}$ are contained in Span($\mathbf{B}$), and μ_{fixed} lies in the orthogonal complement of Span($\mathbf{V}$).[2] To see this, note that all input points to SIFT are of the form $\mathbf{B}\mathbf{B}^\top X_i$ (Line 4 of FastSIFT), and because SIFT then works in the coordinate system of $\mathbf{B}$, every element of the output list will have this form. In particular, μ_{fixed} is the sum of μ_{fast} (Line 3, Algorithm 79) and the empirical mean in the last iteration of SIFT projected into $(\mathbf{I} - \mathbf{P})$ (Line 12, Algorithm 78).

Because the proof of Lemma 300 (with $c = 128$, cf. Proposition 72) shows that

$$\|\mu_{\text{fixed}} - (\mathbf{I} - \mathbf{P})\mu^*\|_2^2 \le \frac{512 + 28}{\alpha} = \frac{540}{\alpha}, \ \|\mathbf{P}(X_i - \mu^*)\|_2^2 \le \frac{8}{\alpha} \text{ for some } \hat{\mu} \in L, \tag{G.9}$$

it suffices to reduce the number of $\mathbf{P}X_i$ while maintaining one with squared ℓ_2 distance $O(\frac{1}{\alpha})$ from $\mathbf{P}\mu^*$. We now give our post-processing procedure. In the following, define $n' := |T_{\text{slow}}| = O(\frac{\log R}{\alpha^2})$.

Lemma 303. *The output of Algorithm 83 has $|\widetilde{L}| \le \frac{2}{\alpha}$, and at least one $\hat{\mu} \in \widetilde{L}$ has*

$$\|\hat{\mu} - \mu^*\|_2^2 \le \frac{1052}{\alpha}.$$

The overall runtime of the algorithm is

$$O\left(\frac{1}{\alpha^4} \log(R) \log\left(\frac{1}{\delta}\right)\right).$$

[2] In the implementation, we will have $\mathbf{V} \in \mathbb{R}^{k' \times k}$ where k' is the column dimensionality of $\mathbf{B}$ since it is expressed in the coordinate system of $\mathbf{B}$, but we write it this way for consistency with the whole algorithm.

Algorithm 83: PostProcess(L, α)

1 **Input:** L, the output of Algorithm 79 (FastSIFT) decomposed as (G.8), satisfying (G.9);
2 **Output:** $\widetilde{L}$, a subset of L with $|\widetilde{L}| \le \frac{2}{\alpha}$;
3 $\widetilde{L} \leftarrow \emptyset$;
4 Let $\widetilde{L}$ be a maximal subset of L of points $\hat{\mu} = \mu_{\text{fixed}} + \mathbf{P}X_i$, such that $\|\mathbf{P}(X_i - X_j)\|_2^2 \le \frac{32}{\alpha}$ for at least $\frac{n'\alpha}{2}$ of the $X_j \in T_{\text{slow}}$, and $\|\hat{\mu} - \hat{\mu}'\|_2^2 \ge \frac{128}{\alpha}$, $\forall \hat{\mu}' \in \widetilde{L}$;
5 **Return:** $\widetilde{L}$;

Proof. We first prove the bound on the list size. Note that every element $\hat{\mu} \in \widetilde{L}$ is associated with at least $\frac{n'\alpha}{2}$ elements in T_{slow}; call this the "cluster" of $\hat{\mu}$. By the separation assumption on pairs in $\widetilde{L}$, the clusters of all $\hat{\mu}, \hat{\mu}' \in \widetilde{L}$ are distinct, so there can only be at most $\frac{2}{\alpha}$ clusters as desired.

We now show the error guarantee. By the decomposition (G.8), the assumption (G.9), and the Pythagorean theorem, it suffices to show that for some $\hat{\mu} = \mathbf{P}X_j + \mu_{\text{fixed}}$ in the output list,

$$\|\mathbf{P}(X_j - \mu^*)\|_2^2 \le \frac{512}{\alpha}. \tag{G.10}$$

By assumption, there is a particular $\hat{\mu} = \mathbf{P}X_i + \mu_{\text{fixed}} \in L$ which satisfies the bound (G.9). We will designate this $\hat{\mu}$ as $\hat{\mu}_{\text{good}}$ throughout the proof, and fix the index i to be associated with $\hat{\mu}_{\text{good}}$. Next, we recall that at least $\frac{n'\alpha}{2}$ of the points $X_j \in T_{\text{slow}}$ have

$$\|\mathbf{P}(X_j - \mu^*)\|_2^2 \le \frac{8}{\alpha}.$$

This was shown in the first part of Theorem 89, and is a straightforward application of Markov and Assumption 13. By triangle inequality to μ^* and the definition of $\hat{\mu}_{\text{good}}$, X_i satisfies

$$\|\mathbf{P}(X_i - X_j)\|_2^2 \le \frac{32}{\alpha} \text{ for at least } \frac{n'\alpha}{2} \text{ of the } X_j \in T_{\text{slow}}.$$

Now, assume that (G.10) does not occur; this clearly also means that $\hat{\mu}_{\text{good}}$ cannot belong to $\widetilde{L}$. However, this is a contradiction, since triangle inequality implies that if no point in $\widetilde{L}$ satisfies (G.10), then $\hat{\mu}_{\text{good}}$ would be added to the list by maximality of the subset.

Finally, we show the complexity guarantee. Throughout, we use the assumption that L has already been decomposed as (G.8), and all components in $\mathbf{P}$ are expressed in the coordinate system of $\mathbf{V}$, so all distance comparisons take time $O(k)$. We can first eliminate all points which do not meet the clustering criteria (e.g. do not have enough points nearby) in one pass, in time $O(|L||T_{\text{slow}}|k)$. Afterwards, a naïve greedy algorithm suffices for forming a list $\widetilde{L}$ in Line 4, e.g. iteratively looping over L and performing the check against points in $|\widetilde{L}|$ sequentially until a loop adds no elements to $\widetilde{L}$. This costs $O(|L||\widetilde{L}|^2 k)$, which yields the runtime since we argued $|\widetilde{L}| = O(k)$. □

G.4.2 Bounding dataset diameter

In Section G.3, we developed an algorithm for list-decodable mean estimation under Assumption 14. We now demonstrate how to reduce a general dataset satisfying Assumption 13 to this case. Our strategy will be to divide the original dataset into multiple portions of bounded diameter, such that with high probability all of the points in S satisfying Assumption 13 lie in the same set. To do so, we perform a random one-dimensional projection, which is likely to preserve distances up to a polynomial factor, and then use an equivalence class partition as our clustering. We state two simple facts which are helpful in the analysis.

Lemma 304. *No two points $X_i, X_j \in S$ have $\|X_i - X_j\|_2 \geq 2\sqrt{n}$.*

Proof. It suffices to show that every point in S has distance at most $\sqrt{n}$ from μ^*. If this were not the case, it is clear Assumption 13 cannot hold by virtue of the corresponding rank-one term. □

Lemma 305. *Let T be a set of n points in $\mathbb{R}^d$, and sample $g \sim \mathcal{N}(0, \mathbf{I})$. With probability at least $1 - \delta$, for every pair of distinct points $X_i, X_j \in T$,*

$$\frac{1}{4 \log \frac{n}{\delta}} \left(\langle g, X_i - X_j \rangle\right)^2 \leq \|X_i - X_j\|_2^2 \leq \frac{n^4}{\delta^2} \left(\langle g, X_i - X_j \rangle\right)^2 . \tag{G.11}$$

Proof. Fix a pair $X_i, X_j \in T$; we show that each of the bounds in (G.11) holds with probability at least $1 - \frac{\delta}{n^2}$, and then the conclusion holds by a union bound over both tails and all pairs. Since the distribution of $\langle g, X_i - X_j \rangle$ is $\mathcal{N}(0, \|X_i - X_j\|_2^2)$, the lower bound in (G.11) is a straightforward application of sub-Gaussian concentration. The upper bound comes from the fact that the probability mass of $\mathcal{N}(0, 1)$ in the range $[-\sqrt{\epsilon}, \sqrt{\epsilon}]$ is bounded by

$$\frac{1}{\sqrt{2\pi}} \int_{-\sqrt{\epsilon}}^{\sqrt{\epsilon}} \exp\left(-\frac{1}{2} t^2\right) dt \leq \sqrt{\epsilon}.$$

Hence, the probability that $Z \sim \mathcal{N}(0, \|X_i - X_j\|_2^2)$ has $Z^2 \leq \frac{\delta^2}{n^4}$ is bounded by $\frac{\delta}{n^2}$. □

At this point, we are ready to give our pre-processing procedure.

Lemma 306. PreProcess *meets its output specifications. The overall runtime is*

$$O\left(nd + n \log n\right).$$

Proof. The runtime bound is immediate; Lines 3 and 4 clearly take time $O(nd)$, and Line 5 can be performed by sorting the values $\{v_i\}_{i \in T}$ and greedily forming clusters, creating disjoint paths from the smallest value to the largest. To show correctness, condition on the conclusion of Lemma 305 occuring (giving the failure probability). We begin with the claim that all of S is contained in a

Algorithm 84: PreProcess(T, δ)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumption 13, $\delta \in (0,1)$;
Output: Partition of T into disjoint clusters $\{T_j\}_{j\in[m]}$, such that all of S is contained in a single cluster, and every cluster has radius $\le \frac{4n^4}{\delta^2}$, with probability $1-\delta$;
$g \sim \mathcal{N}(0, \mathbf{I})$;
$v_i \leftarrow \langle g, X_i\rangle$ for all $X_i \in T$;
Partition T into equivalence classes $\{T_j\}_{j\in[m]}$, where indices i, i' are in the same T_j if there is a path of distinct $i_1 = i, i_2, \dots i_\ell = i'$ so that each consecutive $|v_{i_a} - v_{i_{a+1}}| \le 4\sqrt{n \log \frac{n}{\delta}}$;
Return: Clusters in $\{T_j\}_{j\in[m]}$ with at least αn points;

single cluster; to see this, if $X_i, X_j \in S$, then combining Lemma 304 and Lemma 305 implies that

$$(v_i - v_j)^2 \le \left(4 \log \frac{n}{\delta}\right)(4n) \implies |v_i - v_j| \le 4\sqrt{n \log \frac{n}{\delta}}.$$

Furthermore, suppose two points X_i, $X_{i'}$ are in the same cluster, witnessed by a path of length $\ell \le n$ starting at $i_1 = i$ and ending at $i_\ell = i'$. Then, by triangle inequality

$$|v_i - v_{i'}| \le \sum_{a=1}^{\ell-1} |v_{i_a} - v_{i_{a+1}}| \le 4\sqrt{n^3 \log \frac{n}{\delta}}$$
$$\implies \left(\langle g, X_i - X_{i'}\rangle\right)^2 \le 16 n^3 \log \frac{n}{\delta} \implies \|X_i - X_{i'}\|_2^2 \le \frac{16n^7}{\delta^2} \log \frac{n}{\delta} \le \frac{16n^8}{\delta^4}.$$

In the last implication, we used the upper bound in Lemma 305. □

G.4.3 Putting it all together

Finally, we put together the pieces we have developed to give our final algorithm.

Algorithm 85: ListDecodableMeanEstimation(T, δ)

Input: $T \subset \mathbb{R}^d$ with $|T| = n$ satisfying Assumption 13, $T_{\text{slow}} \subset \mathbb{R}^d$ with $|T_{\text{slow}}| = O(\frac{\log d/\delta}{\alpha^2})$ satisfying Assumption 13 for $\frac{1}{\alpha}$ fixed $O(\frac{\log d/\delta}{\alpha})$-dimensional subspaces (cf. Remark 14, where we use $R = \text{poly}(d, \delta^{-1})$ as below, where $n = \text{poly}(d)$), $\delta \in (0,1)$;
Output: $L \subset \mathbb{R}^d$ with $|L| \le \frac{2}{\alpha}$ satisfying (G.2) with probability $\ge 1 - \delta$;
$\{T_j\}_{j\in[m]} \leftarrow$ PreProcess$(T, \frac{\delta}{2})$;
$L_j \leftarrow$ FastSIFT$(T_j, \frac{\delta\alpha}{2}, $ProduceGoodTuple$)$, for all $j \in [m]$, with $R = \frac{4n^4}{\delta^2}$, and $\alpha_j = \frac{\alpha|T|}{|T_j|}$, reusing the same datapoints T_{slow} for each call to FastSIFT;
$L_j \leftarrow$ PostProcess(L_j, α_j), for all $j \in [m]$;
Return: $L \leftarrow \bigcup_{j\in P} L_j$, where $P = \{j \in [m] \mid |L_j| \le \frac{2}{\alpha_j}\}$;

Theorem 90. *Under Assumption 13, with probability at least* $1 - \delta$, ListDecodableMeanEstimation

outputs a list of size at most $\frac{2}{\alpha}$*, and attains error*

$$\min_{\mu \in L} \|\mu - \mu^*\|_2 = O\left(\frac{1}{\sqrt{\alpha}}\right).$$

The overall runtime is

$$O\left(\frac{nd}{\alpha}\log^2(d)\log^3\left(\frac{d}{\delta}\right)\log\left(\frac{1}{\alpha}\right) + \frac{1}{\alpha^6}\log^4\left(\frac{d}{\delta}\right)\right).$$

Proof. We will show correctness and complexity of Algorithm 85 separately.

Correctness guarantee. First, note there are at most α^{-1} clusters outputted by PreProcess, so by a union bound, with probability at least $1-\delta$, both PreProcess and all FastSIFT calls succeed. Note that whichever cluster T_j that contains all of S indeed satisfies Assumption 13, with $|S| = \alpha_j |T_j|$, by definition of α_j. Thus, Corollary 66 and Lemma 303 imply that index j will belong to the output set P, and an element of L_j will meet the error guarantee (G.2). The list size follows from

$$|L| \le \sum_{j \in [m]} \frac{2}{\alpha_j} = \frac{2}{\alpha}.$$

Finally, we remark that we can reuse the same slow dataset T_{slow} for each of the at most $\frac{1}{\alpha}$ runs of FastSIFT in Line 4, corresponding to different clusters, up to a $\frac{1}{\alpha}$ factor in the failure probability of Proposition 68. This is because (as in Remark 14), the low-dimensional subspaces produced by ProduceGoodTuple are each independent of any randomness used in generating the set T_{slow}.

Complexity guarantee. The cost of PostProcess given in Lemma 303 never dominates the cost of FastSIFT given in Corollary 66; similarly, it is clear that the cost of PreProcess given in Lemma 306 never dominates. Thus, it suffices to bound the costs of all calls to FastSIFT in Line 4. To this end, we bound contributions of the two terms in the runtime of Corollary 66. Because each $\alpha_j \ge \alpha$ and the sum of the sizes of the $\{T_j\}_{j \in [m]}$ is n,

$$\sum_{j \in [m]} \frac{|T_j| d}{\alpha_j} \le \frac{|T| d}{\alpha} = \frac{nd}{\alpha}.$$

Similarly, denoting $k_j = \frac{1}{\alpha_j}$ and $k = \frac{1}{\alpha}$, since $\sum_{j \in [m]} k_j = k$ by design,

$$\sum_{j \in [m]} k_j^6 \le \left(\sum_{j \in [m]} k_j\right)^6 = \frac{1}{\alpha^6}.$$

□

G.4.4 Trading off accuracy for runtime

In this section, we give a simple alternative to the algorithm ListDecodableMeanEstimation which removes the lower-order term in the runtime (so that the complexity is just the cost of polylogarithmically many calls to a k-PCA routine), at the cost of a slight loss in the accuracy term. We first note that unless $\alpha^{-1} = \omega\left(\sqrt{d}\right)$, the term with dependence α^{-6} will not dominate the complexity of Theorem 90. This is because we choose our sample complexity (following Assumption 13) to be on the order of $\frac{d}{\alpha}$, so that asymptotically,

$$\frac{1}{\alpha^6} > \frac{nd}{\alpha} \implies d^2 < \frac{1}{\alpha^4}.$$

We now give the main result of this section, which shows in this regime of α^{-1}, it suffices to randomly sample in the last stage of each run of FastSIFT rather than apply SIFT. The following Algorithm 86 (FasterSIFT) is a simple modification of FastSIFT, which is the same for the first three lines, as well as the last. The only difference is that in Line 4, the list L_{slow} is formed by random sampling points from T_{slow} and projecting into the subspace $\mathbf{B}\mathbf{B}^\top$.

Algorithm 86: FasterSIFT$(T, \delta,$ ProduceGoodTuple$)$

1 **Input:** $T = T_{\text{fast}} \cup T_{\text{slow}} \subset \mathbb{R}^d$ with $|T_{\text{fast}}| = n$ satisfying Assumptions 13 and 14, $|T_{\text{slow}}| = O(\frac{\log R}{\alpha^2})$ satisfying Assumption 13 for a fixed $O(\frac{\log R}{\alpha})$-dimensional subspace, $\delta \in (0,1)$, subroutine ProduceGoodTuple which returns a good tuple with specified failure probability;
2 $(\mathbf{B}, w) \leftarrow$ ProduceGoodTuple$(T_{\text{fast}}, \frac{\delta}{2})$;
3 $\mu_{\text{fast}} \leftarrow (\mathbf{I} - \mathbf{B}\mathbf{B}^\top)\mu_w(T)$;
4 $L_{\text{slow}} \leftarrow \{\mathbf{B}\mathbf{B}^\top X_i$ where $i \in T_{\text{slow}}$ is sampled uniformly at random$\}$, with list size $|L_{\text{slow}}| = \lceil \frac{2}{\alpha} \log \frac{4}{\delta\alpha} \rceil$;
5 **Return:** $L \leftarrow \{\mu_{\text{slow}} + \mu_{\text{fast}} \mid \mu_{\text{slow}} \in L_{\text{slow}}\}$;

Corollary 67. *Consider running* ListDecodableMeanEstimation *with a modification: in Line 4, use* FasterSIFT *(Algorithm 86) in place of* FastSIFT *(Algorithm 79). The resulting list has size at most* $\frac{2}{\alpha}$*. Under Assumption 13, with probability at least* $1 - \delta$*, the overall runtime is*

$$O\left(\frac{nd}{\alpha} \log^2(d) \log^3\left(\frac{d}{\delta}\right) \log\left(\frac{1}{\alpha}\right)\right),$$

and the error guarantee is

$$\min_{\mu \in L} \|\mu - \mu^*\|_2 = O\left(\sqrt{\frac{\log \frac{1}{\delta\alpha}}{\alpha}}\right).$$

Proof. We first discuss list size and error guarantee. It suffices to show that for the cluster T_j containing all of S, we can modify Lemma 303 to obtain a list size $\frac{2}{\alpha_j}$ and error guarantee on

the order of $\sqrt{\log(1/\delta\alpha)/\alpha}$. To see this, all arguments in Lemma 303 follow identically, except that the random sampling occured in a $O(\frac{\log R}{\alpha_j})$-dimensional space. Hence, the error guarantee is correspondingly amplified, where we recall $R = \text{poly}(d, \delta^{-1})$, but the list size argument is the same (e.g. we only keep means which contain at least $O(|T_j|\alpha_j)$ points within their cluster, and all clusters are disjoint).

We now discuss runtime. The cost of all runs of FastSIFT remains the same, up until the step where SIFT is run; clearly, the cost of random sampling is cheaper than running ProduceGoodTuple, once the projections into the coordinate system of $\mathbf{B}$ have already been formed. Finally, the only place that we can lose runtime due to working in a larger-dimensional subspace is in the complexity of PostProcess, where operations are done in $O(\frac{\log R}{\alpha})$ dimensions. Mirroring the proof of Lemma 303, this only adds a $\log R$ overhead, and it is straightforward to check that the cost of all runs of PostProcess do not dominate, since for $d \geq \alpha^{-1}$ and our choice of n, $\frac{nd}{\alpha} \geq \frac{1}{\alpha^4}$. □

G.5 Warmup: fast Gaussian multifilter

As a warmup to our (stronger) developments in Section 8.4, we give a complete algorithm for list-decodable mean estimation in the Gaussian case, i.e. where the "true" distribution $\mathcal{D}$ is drawn from a Gaussian with covariance bounded by $\mathbf{I}$. We encourage the interested reader to read this section before Section 8.4. Conceptually, the types of statements Gaussian concentration (rather than heavy-tailed concentration) allow us to make let us simplify several of the technical difficulties alluded to at the end of Section 8.3.3, in particular the following.

1. Instead of a "covariance bound" statement such as (8.3) to use in our potential proof, we will simply guarantee that the multifilter returns sets of points which lie in short intervals along a number of random directions given by a Johnson-Lindenstrauss sketch.

2. Instead of a randomly subsampled filtering step to remove outliers without soft downweighting (to preserve truly small subsets), it will be enough to deterministically set thresholds along 1-dimensional projections to safely remove the outliers.

3. The definition of the Gaussian multifilter (see Section G.5.2) will be substantially simpler, since we have more explicit tail bounds to check for outliers.

The strength of the error guarantees of the simpler algorithm in this section are somewhat weaker than those of Section 8.4 even when specialized to the Gaussian case, but we include this section as an introductory exposition of our techniques. We will use the stronger Gaussian concentration assumption in this section, a tightening of Assumption 13.

Throughout this section, we will assume that $\alpha \in [1/d, 1/\log^C d]$, for some constant $C > 0$ without loss of generality, as discussed in Section 8.5.

We now formally define the regularity condition which we will use throughout this section.

Assumption 15. *There is a subset $S \subseteq T \subset \mathbb{R}^d$ of size $\alpha n = \Theta(d \cdot \mathrm{polylog}(d))$, and a vector $\mu^* \in \mathbb{R}^d$, such that for all unit vectors $v \in \mathbb{R}^d$ and thresholds $t \in \mathbb{R}_{\geq 0}$,*

$$\Pr_{i \sim_{\mathrm{unif}} S}[\langle X_i - \mu^*, v\rangle > t] \leq \exp(-\Omega(t^2)) + \frac{1}{\Omega\left(\log^3 d\right)}.$$

Here, the notation $i \sim_{\mathrm{unif}} S$ means that i is a uniformly random sampled index from S.

This assumption is standard in the literature, and follows when the true distribution which S is sampled from is Gaussian with identity-bounded covariance (see e.g. Definition A.4, Lemma A.5 [177]). We remark that the sample complexity of Assumption 15 is worse than that of Assumption 13 by a polylogarithmic factor. This lossiness is just to simplify exposition in this warmup section, and indeed in the following Section 8.4 we give an algorithm which recovers stronger guarantees than Theorem 91, this section's main export, under only Assumption 13.

In Section G.5.1, we first give our main subroutine, GaussianPartition, which takes a candidate set and produces a number of children candidate sets which each satisfy a progress guarantee similar to (8.3). The main difficulty will be in guaranteeing that the children sets are sufficiently small, and that if the parent set was "good" (had large overlap with S), then at least one child set will as well. We reduce GaussianPartition to a number of one-dimensional clustering steps, which we implement as GaussianSplitOrCluster in Section G.5.2. Finally, we use the guarantees of GaussianPartition within our potential-based framework outlined in Section 8.3.3, giving our final algorithm FastGaussianMultifilter in Section G.5.3. Throughout, sets S and T are fixed and satisfy Assumptions 13 and 15.

G.5.1 Reducing GaussianPartition to GaussianSplitOrCluster

Our final algorithm creates a tree of candidate sets. Every node p in the tree is associated with a subset T_p. In order to progress down the tree, at a given node p we form children $\{c_\ell\}_{\ell \in [k]}$ with associated sets $\{T_{c_\ell}\}_{\ell \in [k]}$; we call the procedure which produces the children node GaussianPartition, and develop it in this section. There are three key properties of GaussianPartition which we need.

1. The sum of the cardinalities of $\{T_{c_\ell}\}_{\ell \in [k]}$ is not too large compared to $|T_p|$. This is to guarantee that at each layer of the tree, we perform about the same amount of work, namely $\widetilde{O}(nd)$. We formalize this with a parameter $\beta \in (0, 1]$ throughout the rest of this section, and will guarantee that every time GaussianPartition is called on a parent node p,

$$\sum_{\ell \in [k]} |T_{c_\ell}|^{1+\beta} \leq |T_p|^{1+\beta}. \tag{G.12}$$

2. If the parent vertex p has substantial overlap with S (at least $\frac{1}{2}|S|$ points), then at least one of the produced children continues to retain all but a small fraction of points in S.

3. Defining the matrices

$$\begin{aligned}\mathbf{M}_p &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_p),\ \mathbf{Y}_p := \mathbf{M}_p^{\log d},\\ \mathbf{M}_{c_\ell} &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{c_\ell}),\ \mathbf{Y}_{c_\ell} := \mathbf{M}_{c_\ell}^{\log d} \text{ for all } \ell \in [k],\end{aligned} \tag{G.13}$$

every $\mathbf{M}_{c_\ell}$ satisfies the bound

$$\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \right\rangle \le R^2 \mathrm{Tr}\left(\mathbf{Y}_p^2\right), \tag{G.14}$$

for some (polylogarithmic) value R we will specify. Note the similarity between this and (8.3); this will be used in a potential analysis to bound progress on covariance operator norms.

We are now ready to state the algorithm GaussianPartition.

Algorithm 87: GaussianPartition$(T_p, \alpha, \beta, C, R)$

Input: $T_p \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $\beta \in (0, 1]$, $C, R \in \mathbb{R}_{\ge 0}$ satisfying (for sufficiently large constants)

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log\left(C\alpha^{-1}\right)}{\beta}\right),\ C = \Omega\left(\log^2 d\right).$$

;

Output: With failure probability $\le \frac{1}{d^3}$: subsets $\{T_{c_\ell}\}_{\ell\in[k]}$ of T_p, satisfying (G.12). Every child satisfies (G.14) (using notation (G.13)). If $|T_p \cap S| \ge (\frac{1}{2} + \frac{1}{C})|S|$, at least one child T_{c_ℓ} satisfies

$$|T_{c_\ell} \cap S| \ge |T_p \cap S| - \frac{1}{C}|S|. \tag{G.15}$$

;

Sample $N_{\mathrm{dir}} = \Theta(\log d)$ vectors $\{u_j\}_{j\in[N_{\mathrm{dir}}]} \in \mathbb{R}^d$ each with independent entries ± 1. Following notation (G.13), let $v_j \leftarrow \mathbf{Y}_p u_j$ for all $j \in [N_{\mathrm{dir}}]$.;

$\mathcal{S}_0 \leftarrow T_p$;

for $j \in [N_{\mathrm{dir}}]$ **do**

  $\mathcal{S}_j \leftarrow \emptyset$;

  for $T' \in \mathcal{S}_{j-1}$ **do**

    $\mathcal{T} \leftarrow$ Gaussian1DPartition$(T', \alpha, v_j, \beta, CN_{\mathrm{dir}}, R)$;

    $\mathcal{S}_j \leftarrow \mathcal{S}_j \cup \mathcal{T}$;

Return: $\mathcal{S}_{N_{\mathrm{dir}}}$;

It heavily relies on a subroutine, Gaussian1DPartition(T', v) which takes a subset T' and a vector $v \in \mathbb{R}^d$, and produces children subsets of T' satisfying the first two conditions above, and also guarantees that along the direction v, each child subset is contained in a relatively short interval.

Once again, Gaussian1DPartition heavily relies on a subroutine, GaussianSplitOrCluster, which we implement in Section G.5.2. It takes as input a set T'' and either produces one or two subsets of T''

Algorithm 88: Gaussian1DPartition$(T', \alpha, v, \beta, C, R)$

Input: $T' \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $C, R \in \mathbb{R}_{\geq 0}$ satisfying (for sufficiently large constants)

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log\left(C\alpha^{-1}\right)}{\beta}\right),\ C = \Omega\left(\log^3 d\right).$$

;

Output: Subsets $\{T''_\ell\}_{\ell \in [k]} \subseteq T'$, such that

$$\sum_{\ell \in [k]} |T''_\ell|^{1+\beta} \leq |T'|^{1+\beta}. \tag{G.16}$$

If $|T' \cap S| \geq (\frac{1}{2} + \frac{1}{C})|S|$, at least one child T''_ℓ satisfies

$$|T''_\ell \cap S| \geq |T' \cap S| - \frac{1}{C}|S|.$$

Every child has all values $\{\langle v, X_i \rangle \mid i \in T''_\ell\}$ contained in an interval of length $R\|v\|_2$.;

$\mathcal{S}_{\text{in}} \leftarrow \{T'\}$, $\mathcal{S}_{\text{out}} \leftarrow \emptyset$;

while $\mathcal{S}_{in} \neq \emptyset$ **do**

  $T'' \leftarrow$ the first element of $\mathcal{S}_{\text{in}}$;

  $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \setminus T''$;

  if GaussianSplitOrCluster$(T'', \alpha, \beta, R, \frac{1}{Cn})$ *returns one set* $T^{(0)}_{out}$ **then**

    $\mathcal{S}_{\text{out}} \leftarrow \mathcal{S}_{\text{out}} \cup \left\{T^{(0)}_{\text{out}}\right\}$;

  else

    $T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}} \leftarrow$ GaussianSplitOrCluster$(T'', \alpha, \beta, R, \frac{1}{Cn})$;

    $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \cup \left\{T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}}\right\}$;

as output. If it outputs one set, that set has length at most $R\|v\|_2$ in the direction v; otherwise, Gaussian1DPartition simply recurses on the additional two sets. Crucially, GaussianSplitOrCluster guarantees that if T'' has substantial overlap with S, then so does at least one child; moreover, when GaussianSplitOrCluster returns two sets, they satisfy a size potential such as (G.16). We now demonstrate correctness of GaussianPartition, assuming that Gaussian1DPartition is correct.

Lemma 307. *The output of* GaussianPartition *satisfies the guarantees given in Line 2 of Algorithm 87, assuming correctness of* Gaussian1DPartition*.*

Proof. First, to demonstrate that the subsets satisfy (G.12), we observe that we can view GaussianPartition as always maintaining a set of subsets, $\mathcal{S}_j$ (in the beginning, $\mathcal{S}_0 = T_p$). The set $\mathcal{S}_j$ is formed by calling Gaussian1DPartition on elements of $\mathcal{S}_{j-1}$, each of which satisfy (G.16), so inductively $\mathcal{S}_{N_{\text{dir}}} = \{T_{c_l}\}_{l\in[k]}$ will satisfy (G.12) with respect to $\mathcal{S}_0 = T_p$ as desired.

Next, by recursively using the guarantee of Gaussian1DPartition, every $T_{c_l} \in \mathcal{S}_{N_{\text{dir}}}$ will satisfy

$$\text{all values } \{\langle v_j, X_i\rangle \mid i \in T_{c_l}\} \text{ are contained in an interval of length } R\|v_j\|_2, \text{ for all } j \in [N_{\text{dir}}].$$

In other words, this set is short along all the directions $\{\mathbf{Y}_p u_j = v_j\}_{j\in[N_{\text{dir}}]}$. This lets us conclude

$$\begin{aligned}
\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_l}\right\rangle &= \frac{1}{2n\left|T_{c_l}\right|}\left\langle \mathbf{Y}_p^2, \sum_{i,i'\in T_{c_l}} (X_i - X_{i'})(X_i - X_{i'})^\top \right\rangle \\
&= \frac{1}{2n\left|T_{c_l}\right|}\sum_{i,i'\in T_{c_l}} \|\mathbf{Y}_p(X_i - X_{i'})\|_2^2 \\
&\le \frac{1.4}{2n\left|T_{c_l}\right| N_{\text{dir}}}\sum_{i,i'\in T_{c_l}}\sum_{j\in[N_{\text{dir}}]} \langle \mathbf{Y}_p u_j, X_i - X_{i'}\rangle^2 \\
&\le \frac{1.4}{2n\left|T_{c_l}\right| N_{\text{dir}}}\sum_{i,i'\in T_{c_l}}\sum_{j\in[N_{\text{dir}}]} R^2\|\mathbf{Y}_p u_j\|_2^2 \\
&\le \frac{1.4}{2N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]} R^2\|\mathbf{Y}_p u_j\|_2^2 \le R^2\text{Tr}\left(\mathbf{Y}_p^2\right),
\end{aligned}$$

with probability at least $1 - \frac{1}{2d^3}$. Here, we used Fact 16 in the first line and linearity of trace in the second line. The third line used the Johnson-Lindenstrauss lemma of [6] which says that for any vector v, $\frac{1}{N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]}\langle u_j, v\rangle^2 \in [0.6, 1.4]\|v\|_2^2$ for a sufficiently large $N_{\text{dir}} = \Theta(\log(d))$ with probability at least $1 - \frac{1}{2d^6}$, which we union bound over all $|T_{c_l}|^2 \le n^2 \le d^4$ pairs of points. The fourth line used the radius guarantee of Gaussian1DPartition, and the fifth used $|T_{c_l}| \le n$ and the Johnson-Lindenstrauss lemma guarantee that $\frac{1}{N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]}\|\mathbf{Y}_p u_j\|_2^2 \in [0.6, 1.4]\text{Tr}(\mathbf{Y}_p^2)$ with probability at least $1 - \frac{1}{2d^3}$, which can be deduced by the guarantee of [6] applied to the rows of $\mathbf{Y}_p$. Union bounding over the two applications of [6] yields the claim.

Finally, to demonstrate that at least one child satisfies (G.15), suppose p satisfies $|T_p \cap S| \ge (\frac{1}{2} +$

$\frac{1}{C}$)$|S|$ (i.e. it has substantial overlap with S). Then by applying the guarantee of Gaussian1DPartition inductively, every $\mathcal{S}_j$ will have at least one element T' satisfying $|T' \cap S| \geq \frac{1}{2}|S|$. Every call to Gaussian1DPartition only removes $\frac{1}{CN_{\text{dir}}}|S|$ points in S, so overall only $\frac{1}{C}|S|$ points are removed. □

G.5.2 Implementation of GaussianSplitOrCluster

In this section, we first state GaussianSplitOrCluster and analyze its correctness. We conclude with a full runtime analysis of Gaussian1DPartition, using our GaussianSplitOrCluster implementation.

Algorithm 89: GaussianSplitOrCluster$(T_{\text{in}}, \alpha, v, \beta, R, \Delta)$

1 **Input:** $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$, $\Delta \in (0, 1)$;

2 **Output:** Either one subset $T^{(0)}_{\text{out}} \subset T_{\text{in}}$, or two subsets $T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}} \subset T_{\text{in}}$. In the one subset case, $T^{(0)}_{\text{out}}$ has $\left\{\langle v, X_i\rangle \mid i \in T^{(0)}_{\text{out}}\right\}$ contained in an interval of length $R\|v\|_2$. In the two subsets case, they take the form, for some threshold value $\tau \in \mathbb{R}$ and $r := \frac{R}{4k_{\max}}, k_{\max} = \Theta\left(\frac{\log\log(\frac{1}{\alpha\Delta})}{\beta}\right)$

$$T^{(1)}_{\text{out}} := \{X_i \mid \langle v, X_i\rangle \leq \tau + r\|v\|_2\},\ T^{(2)}_{\text{out}} := \{X_i \mid \langle v, X_i\rangle \geq \tau - r\|v\|_2\}, \tag{G.17}$$

and satisfy

$$\left|T^{(1)}_{\text{out}}\right|^{1+\beta} + \left|T^{(2)}_{\text{out}}\right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}. \tag{G.18}$$

;

3 $Y_i \leftarrow \langle v, X_i\rangle$ for all $i \in T_{\text{in}}$;

4 $T^{(0)}_{\text{out}} \leftarrow$ indices in the middle $1 - \alpha\Delta$ quantiles of $\{Y_i\}_{i \in T_{\text{in}}}$;

5 **if** $\left\{Y_i \mid i \in T^{(0)}_{\text{out}}\right\}$ *is contained in an interval of length* $R\|v\|_2$ **then**

6 **Return:** $T^{(0)}_{\text{out}}$;

7 **else**

8 $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$, where med returns the median;

9 $\tau_k \leftarrow \tau_{\text{med}} + 2kr\|v\|_2$ for all integers $-k_{\max} \leq k \leq k_{\max}$;

10 **Return:** $T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}}$ defined in (G.17) for any threshold τ_k inducing sets satisfying (G.18);

To analyze Algorithm 89 we first demonstrate that it always returns in at least one case. In particular, we demonstrate that whenever the set $T^{(0)}_{\text{out}}$ is not sufficiently short, then there will be a threshold parameter k such that the induced sets in (G.17) satisfy the size bound (G.18).

Lemma 308. *Suppose Algorithm 89 does not return on Line 6. Then, there exists a $k \in \mathbb{Z}$ in the range $-k_{\max} \leq k \leq k_{\max}$ such that Algorithm 89 is able to return on Line 10.*

Proof. We instead prove that if there is no such k, then we will have a contradiction on the length of the set $T^{(0)}_{\text{out}}$ in the direction v. We first lower bound the length of the $[\frac{1}{2}, 1 - \frac{\alpha\Delta}{2}]$ quantiles

of $\{Y_i \mid i \in T_{\text{in}}\}$ by $\frac{1}{2}R\|v\|_2$; the lower bound for the $[\frac{\alpha\Delta}{2}, \frac{1}{2}]$ quantiles will follow analogously. Combining shows that if no threshold works, then the algorithm should have returned $T_{\text{out}}^{(0)}$.

For any threshold τ, define $g(\tau) \in [0,1]$ to be the proportion of $\{Y_i \mid i \in T_{\text{in}}\}$ which are $\geq \tau$. Moreover, define for all $1 \leq k \leq k_{\max}$,

$$\gamma_k := g(\tau_k - r\|v\|_2) = g(\tau_{\text{med}} + (2k-1)r\|v\|_2),$$

and note that $\gamma_1 \leq \frac{1}{2}$ by definition, since τ_{med} was the median. Now, for each $1 \leq k \leq k_{\max}$, since τ_k was not a valid threshold, the sets

$$T_k^{(1)} := \{X_i \mid Y_i \leq \tau_{\text{med}} + (2k+1)r\|v\|_2\},\ T_k^{(2)} := \{X_i \mid Y_i \geq \tau_{\text{med}} + (2k-1)r\|v\|_2\}$$

do not satisfy the size bound (G.18). Normalizing both sides of (G.18) by $|T_{\text{in}}|^{1+\beta}$ and using the definitions of $\{\gamma_k\}$, we obtain the following recursion:

$$(1-\gamma_{k+1})^{1+\beta} + \gamma_k^{1+\beta} = \left(\frac{\left|T_k^{(1)}\right|}{|T_{\text{in}}|}\right)^{1+\beta} + \left(\frac{\left|T_k^{(2)}\right|}{|T_{\text{in}}|}\right)^{1+\beta} \geq 1 \implies \gamma_{k+1} \leq \gamma_k^{1+\beta}. \tag{G.19}$$

To obtain the above implication, we used $1-(1-x)^{1+\beta} > x^{1+\beta}$ for all $x, \beta \in [0,1]$. By repeatedly applying the recursion (G.19), we have

$$\gamma_{k_{\max}} \leq \gamma_1^{(1+\beta)^{(k_{\max}-1)}} \leq \left(\frac{1}{2}\right)^{(1+\beta)^{(k_{\max}-1)}} \leq \frac{\alpha\Delta}{2},$$

where we use the definition of $k_{\max}$ and $\gamma_1 \leq \frac{1}{2}$. Thus, the $[\frac{1}{2}, 1-\frac{\alpha\Delta}{2}]$ quantiles are contained between τ_{med} and $\tau_{\text{med}} + (2k_{\max}-1)r\|v\|_2 \leq \tau_{\text{med}} + \frac{1}{2}R\|v\|_2$. By repeating this argument in the range $-k_{\max} \leq k \leq -1$, we obtain a contradiction (as Algorithm 89 should have returned $T_{\text{out}}^{(0)}$). □

We next prove that if the input T' to Gaussian1DPartition has large overlap with S, then the algorithm always returns some child T'' which removes at most $\frac{1}{C}|S|$ points from this overlap. This proof uses the implementation of GaussianSplitOrCluster in a white-box way, as well as Assumption 15.

Lemma 309. *Whenever* Gaussian1DPartition *is called on T' with $|T' \cap S| \geq \left(\frac{1}{2} + \frac{1}{C}\right)|S|$ with parameters R, C satisfying (for sufficiently large constants)*

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log(Cd)}{\beta}\right),\ C = \Omega\left(\log^3 d\right),$$

it produces some child T'' satisfying $|T'' \cap S| \geq |T' \cap S| - \frac{1}{C}|S|$.

Proof. We first discuss the structure of Gaussian1DPartition. We say a call to GaussianSplitOrCluster is a "split step" if it produces two sets, and otherwise we call it a "cluster step." Every output

child of Gaussian1DPartition is the result of a consecutive number of split steps, and then one cluster step. Also, every split step replaces an interval with its intersections with two half-lines which overlap by $2r \|v\|_2 = \Omega(\sqrt{\log(C)} \|v\|_2)$. Assume for simplicity that $\|v\|_2 = 1$ in this proof; analogous arguments hold for all v by scaling everything appropriately. Finally, we recall that all calls to GaussianSplitOrCluster in Gaussian1DPartition are with $\Delta = \frac{1}{Cn}$.

Our key technical claim is that after any number of split steps forming a partition of the real line, there is always some interval such that $\langle v, \mu^* \rangle$ is r away from both endpoints (in this proof, we allow intervals to have endpoints at $\pm\infty$). This is clearly true at the beginning, since the only interval is $(-\infty, \infty)$. Next, we induct and assume that on the current partition, after some number of split steps, there is an interval $[a, b]$ in the partition such that $\langle v, \mu^* \rangle \in [a + r, b - r]$. Consider the intersection of this interval with any split step, parameterized by the half-lines $(-\infty, \tau + r]$ and $[\tau - r, \infty)$ for some $\tau \in \mathbb{R}$. If $\langle v, \mu^* \rangle \geq \tau$, then one of the resulting intervals is

$$[\max(a, \tau - r), b]$$

where we note that this interval is non-degenerate by assumption; $\tau \leq \langle v, \mu^* \rangle \leq b - r \implies \tau - r \leq b$. If the result of the max is $[a, b]$, then the claim holds; otherwise, the interval is $[\tau - r, b]$ and the claim holds by induction ($\langle v, \mu^* \rangle \leq b - r$) and the assumption $\langle v, \mu^* \rangle \geq \tau$. The other case when $\langle v, \mu^* \rangle \leq \tau$ follows symmetrically by considering the interval $[a, \min(b, \tau + r)]$.

Now, consider the partition of the real line which is induced by the eventual children outputted by Gaussian1DPartition, *right before* the last cluster step is applied to them (in other words, this partition is formed only by split steps). Using the above argument, there is some element of this partition $[a, b]$ so that $\langle v, \mu^* \rangle \in [a + r, b - r]$. Applying Assumption 15 shows that if we consider the effects of truncating the set $\{Y_i \mid i \in S\}$ at the endpoints of this interval, we remove at most a $\frac{1}{2C}$ fraction of the points from S. Finally, the interval that is returned is the result of a cluster step applied to this interval. This can only remove at most an $\alpha\Delta \leq \frac{\alpha}{2C}$ fraction of the overall points, which is at most $\frac{1}{2C}|S|$. Combining these two bounds yields the claim. □

Finally, we conclude with a runtime analysis of Gaussian1DPartition.

Lemma 310. *Let $n' := |T'|$ for some $T' \subseteq T$.* Gaussian1DPartition *called on input T' with parameter C can be implemented to run in time*

$$O\left(n'd + (n')^{1+\beta} \log n' \cdot \frac{\log\log(Cd)}{\beta}\right).$$

Proof. We begin by forming all of the one-dimensional projections $\langle v, X_i \rangle$ for all $i \in T'$, and sorting these values. We also store the quantile of each point (i.e. the number of points larger than it). The total cost of these operations is $O(n'd + n' \log n')$.

Next, given this total ordering, observe that the structure of Gaussian1DPartition means that every

set in $\mathcal{S}_{\text{in}}$ is a subinterval of T', since this is inductively preserved by calls to GaussianSplitOrCluster; hence, we can represent every set implicitly by its endpoints. Moreover, given access to the initial quantile information we can implement every call to GaussianSplitOrCluster in time $O(k_{\max} \log n') = O(\log n' \cdot \frac{\log\log(Cd)}{\beta})$, since the cost of checking the length of $T_{\text{out}}^{(0)}$ is constant, and the cost of checking each candidiate τ_k is dominated by determining the thresholds of the corresponding induced sets $T_{\text{out}}^{(1)}$ and $T_{\text{out}}^{(2)}$. These can be performed via binary searches in $O(\log n')$ time.

It remains to bound the number of calls to GaussianSplitOrCluster throughout the execution of Gaussian1DPartition. To this end, we bound the number of times GaussianSplitOrCluster can return one set, and the number of times it can return two sets. Every time GaussianSplitOrCluster returns one set, it adds it to $\mathcal{S}_{\text{out}}$, and by using the guarantee (G.18) recursively, there can only ever be $(n')^{1+\beta}$ such sets. Similarly, every time it returns two sets it increases $|\mathcal{S}_{\text{in}}| + |\mathcal{S}_{\text{out}}|$ by one, but we know at termination this is at most $(n')^{1+\beta}$, and this potential never decreases. Thus, the total number of calls to GaussianSplitOrCluster is bounded by $O((n')^{1+\beta})$, as desired. □

As an immediate corollary, we obtain a runtime bound on GaussianPartition.

Corollary 68. *Let $n_p := |T_p|$ for some $T_p \subseteq T$.* GaussianPartition *called on input T_p with parameter C can be implemented to run in time*

$$O\left(n_p^{1+\beta} d \log^2(d) + n_p^{1+\beta} \log^2(d) \cdot \frac{\log\log(Cd))}{\beta}\right).$$

Proof. First, consider the cost of computing all vectors $\mathbf{Y}_p u_j$. It is straightforward to implement matrix-vector multiplications through $\mathbf{M}_p$ in time $O(n_p d)$, so this cost is $O(n_p d \log^2(d))$.

We next require a bound on the cost of $N_{\text{dir}} = \Theta(\log d)$ consecutive calls to Gaussian1DPartition. The cost of each is given by Lemma 310, and the result follows by summing this cost over all elements of each $\mathcal{S}_j$, which can be bounded since for all $j \in [N_{\text{dir}}]$, the cardinalities of all sets contained in $\mathcal{S}_j$ have $1+\beta$ powers bounded by $n_p^{1+\beta}$ by repeatedly using the guarantee (G.16). □

G.5.3 Full Gaussian algorithm

Finally, we are ready to give our full algorithm for list-decodable mean estimation under Assumptions 13 and 15. We begin by reducing the original problem to a number of subproblems of bounded diameter (following Section G.4), and then showing that for each of these subproblems, polylogarithmic calls to GaussianPartition yield subsets of bounded covariance operator norm. We conclude by recalling that a covariance operator norm bound suffices to yield guarantees on mean estimation.

We begin by restating the guarantees of NaiveCluster, used in Line 3 of FastGaussianMultifilter.

Lemma 311 (Restatement, Lemma 306)**.** *There is a randomized algorithm,* NaiveCluster*, which takes as input $T \subset \mathbb{R}^d$ satisfying Assumption 13 and partitions it into disjoint subsets $\{T_i'\}_{i\in[k]}$ such*

Algorithm 90: FastGaussianMultifilter(T, α)

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumptions 13 and 15 with parameter $\alpha \in (0, \frac{1}{2})$;
Output: With failure probability $\leq \frac{1}{d}$: L with $|L| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L$ satisfies

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\frac{\log(d)\log\log^{1.5}(d)}{\sqrt{\alpha}}\right). \tag{G.20}$$

;
$\{T_i'\}_{i\in[k]} \leftarrow$ NaiveCluster(T);
$\alpha_i \leftarrow \frac{|T|}{|T_i'|}\alpha$ for all $i \in [k]$;
Return: $\bigcup_{i\in[k]}$ FastGaussianMultifilterBoundedDiameter(T_i', α_i);

Algorithm 91: FastGaussianMultifilterBoundedDiameter(T, α)

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumptions 13 and 15 with parameter $\alpha \in (0, \frac{1}{2})$;
Output: With failure probability $\leq \frac{1}{d}$: L_{out} with $|L_{\text{out}}| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L_{\text{out}}$ satisfies

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\frac{\log(d)\log\log^{1.5}(d)}{\sqrt{\alpha}}\right).$$

;
$L^{(0)} \leftarrow \{T\}$, $L_{\text{out}} \leftarrow \emptyset$;
For sufficiently large constants,

$$R \leftarrow \Theta\left(\log(d)\log\log^{1.5}(d)\right),\ C \leftarrow \Theta(\log^2 d),\ D \leftarrow \Theta(\log^2 d)$$

;
for $\ell \in [D]$ **do**
  $L^{(\ell)} \leftarrow \emptyset$;
  for $T' \in L^{(\ell-1)}$ **do**
    Append all elements of GaussianPartition$(T', \alpha, \frac{1}{\log d}, C, R)$ to $L^{(\ell)}$ with size at least $\frac{\alpha n}{2}$;
Return: List of empirical means of all sets in $L^{(D)}$;

that with probability at least $1 - \frac{1}{d^2}$*, all of* S *is contained in the same subset, and every subset has diameter bounded by* $O(d^{12})$*. The runtime of* NaiveCluster *is* $O(nd + n \log n)$*.*

We next demonstrate that if the operator norm of the (unnormalized) covariance matrix of a set of points T' is bounded, and T' has sufficient overlap with S, then its empirical mean is close to μ^*.

Lemma 312. *For* $T' \subset T$ *with empirical mean* $\hat{\mu}$*, if* $|T' \cap S| \geq \frac{1}{2}|S|$ *and* $\widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T') \leq R^2$*,*

$$\|\hat{\mu} - \mu^*\|_2 = O\left((1+R) \cdot \frac{1}{\sqrt{\alpha}}\right).$$

Proof. Let w place weight $\frac{1}{n}$ on coordinates in T', and 0 on all other coordinates. Clearly this w satisfies the assumption of Lemma 113, since its ℓ_1 norm is simply $\frac{|T'|}{|T|} \geq \frac{1}{2}\alpha$. The conclusion follows by applying Lemma 113, where we use $\|w\|_1 \mathrm{Cov}_w(T) = \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T')$, and the assumed bound. □

We now give a full analysis of FastGaussianMultifilterBoundedDiameter.

Proposition 73. FastGaussianMultifilterBoundedDiameter *meets its output specifications with probability at least* $1 - \frac{1}{d}$*, within runtime*

$$O\left(nd \log^4(d) + n \log^5(d) \log\log(d)\right).$$

Proof. Throughout, we denote $\beta := \frac{1}{\log d}$. There are three main guarantees of the algorithm: that the list size is $O(\frac{1}{\alpha})$, that some list element satisfies (G.20), and that the runtime is as claimed.

We first bound the list size. We can view FastGaussianMultifilterBoundedDiameter as producing a tree of subsets, of depth D. Each layer of the tree is composed by the sets in $L^{(\ell)}$ where $0 \leq \ell \leq D$, and $L^{(0)}$ is the root node. The children of each node are the results of calling GaussianPartition on the associated subset. Moreover, by repeatedly using the guarantee (G.12) inductively, the total cardinality of all sets at layer ℓ is bounded by $n^{1+\beta} = O(n)$. Since we only return means from sets with size at least $\frac{\alpha n}{2}$ on layer D, there can only be $O(\frac{1}{\alpha})$ such sets.

Next, we bound error rate. Consider some leaf node, and its path to the root; call the sets associated with these vertices $T_0, T_1, \ldots T_D$, where T_D is the leaf node and $T_0 = T$ is the original set. Define the potential function at each layer $0 \leq \ell \leq D$,

$$\Phi_\ell := \mathrm{Tr}\left(\mathbf{M}_\ell^{2\log d}\right), \text{ where } \mathbf{M}_\ell := \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_\ell).$$

Note that every parent-child pair along this path satisfies the guarantee (G.14). We thus conclude

that for each $0 \le \ell < D$, we have the recurrence (analogously to (8.4))

$$\begin{aligned}\Phi_{\ell+1} = \operatorname{Tr}\left(\mathbf{M}_{\ell+1}^{2\log d}\right) &\le \frac{1}{2R^2}\operatorname{Tr}\left(\mathbf{M}_{\ell+1}^{2\log d+1}\right) + d(2R^2)^{2\log d} \\ &\le \frac{1}{2R^2}\operatorname{Tr}\left(\mathbf{M}_{\ell}^{2\log d}\mathbf{M}_{\ell+1}\right) + d(2R^2)^{2\log d} \\ &\le \frac{1}{2}\operatorname{Tr}\left(\mathbf{M}_{\ell}^{2\log d}\right) + d(2R^2)^{2\log d} = \frac{1}{2}\Phi_\ell + d(2R^2)^{2\log d}.\end{aligned}$$

The first line used Fact 18 with $\gamma = 2R^2$, the second used Fact 17, and the third used the guarantee (G.14). Thus, as long as at a layer ℓ we have

$$\Phi_\ell > 4d(2R^2)^{2\log d},$$

we have $\Phi_{\ell+1} \le \frac{3}{4}\Phi_\ell$, and so the potential is decreasing by at least a constant factor. The potential Φ_0 is bounded by $d^{O(\log d)}$, because we assumed the input set has polynomially bounded diameter, so within $D = \Omega(\log^2 d)$ layers, every node on layer D must have $\Phi_D \le 4d(2R^2)^{2\log d}$. This implies that the operator norm of $\widetilde{\operatorname{Cov}}_{\frac{1}{n}\mathbf{1}}(T')$ for every node T' on layer D is $O(R^2)$.

We next show at least one node T' on every layer has $|T' \cap S| \ge \frac{1}{2}|S|$. By inductively using (G.15) with our chosen value of C, summing over the $O(\log^2 d)$ layers guarantees that we only remove at most $\frac{1}{2}|S|$ points from the intersection throughout the root-to-leaf path, for some path. We can now apply Lemma 312 to guarantee (G.20). To obtain the high-probability bound, note that the number of times we call GaussianPartition is bounded by $O(\frac{1}{\alpha}\log^2 d)$, since at each layer we prune every node with less than $\frac{\alpha n}{2}$ points; there can only be $O(\frac{1}{\alpha})$ surviving nodes per layer (since the total cardinalities of the layer is bounded by $n^{1+\beta} = O(n)$), and taking a union bound over all calls to GaussianPartition shows the failure probability is at most $\frac{1}{d}$.

Finally, we discuss runtime. We simply apply Corollary 68 to each layer, which bounds the runtime of each layer by $O(nd\log(d) + n\log^3(d)\log\log(d))$, since the sets on that layer satisfy (G.12) inductively. Summing over all layers yields the desired runtime guarantee. □

Theorem 91. FastGaussianMultifilter *meets its output specifications with probability at least* $1 - \frac{1}{d}$*, within runtime*

$$O\left(nd\log^4(d) + n\log^5(d)\log\log(d)\right).$$

Proof. We apply Proposition 73 to the relevant call of FastGaussianMultifilterBoundedDiameter. Note that all $\alpha_i \ge \alpha$, giving the error guarantee (G.20), and

$$\sum_{i\in[k]} \frac{1}{\alpha_i} = \frac{1}{\alpha},$$

giving the list size guarantee. The runtime follows from $\sum_{i\in[k]} |T_i'| = |T|$. □

G.6 Deferred proofs from Section 3.2

G.6.1 Proof of Proposition 33

In this section, we prove Proposition 33, restated here for convenience.

Proposition 33. *Let $\alpha \geq 1$ and let $\epsilon > 0$ be sufficiently small. Let $\{(X_i, y_i)\}_{i\in[n]} \subset \mathbb{R}^d \times \mathbb{R}$ be an ϵ-corrupted set of samples from a distribution $\mathcal{D}_{Xy}$ as in Model 3. Then, if*

$$n = O\left(\frac{d\alpha^2 \log d}{\epsilon^4} + \frac{d^2\alpha^{1.5}\log(d/\epsilon)}{\epsilon^3}\right) ,$$

the set $\{(X_i, y_i)\}_{i\in[n]}$ is $(2\epsilon, \frac{\epsilon^2}{\alpha})$-good for linear regression with probability at least $\frac{9}{10}$.

Before we prove this lemma, we need the following useful technical lemmata. The first shows that given a large enough sample of points from a distribution with bounded second moment, there is a large subset of points with bounded second moment.

Lemma 313 (Lemma A.20 in [177])**.** *Let $X_1, \ldots, X_n$ be independent samples from a distribution $\mathcal{D}$ with second moment matrix $\mathbf{\Sigma}^\star$, and let $\epsilon > 0$ be sufficiently small. There exists a universal constant $c > 0$ so that if $n \geq c\frac{d\log d}{\epsilon}$, we have that with probability 0.99, there exists a subset S of size $(1-\epsilon)n$ satisfying*

$$\frac{1}{|S|}\sum_{i\in S} X_i X_i^\top \preceq \frac{3}{2}\mathbf{\Sigma}^\star .$$

We note that Lemma A.20 is stated for covariance as opposed to second moment, but the same proof immediately implies the same result for second moment.

We also require the following bound.

Lemma 314 (Lemma 5.1 in [134])**.** *Let $X_1, \ldots, X_n$ be independent samples from a distribution $\mathcal{D}$ with second moment $\mathbf{\Sigma}^\star$, and let $\epsilon > 0$ be sufficiently small. Assume $\mathcal{D}$ is 2-to-4 hypercontractive with parameter $C = O(1)$. There exists a universal constant $c > 0$ so that if $n \geq \frac{cd\log(d/\epsilon)}{\epsilon}$, then with probability $1 - \frac{1}{d^2}$, we have that for any $S \subset [n]$ of size $(1-\epsilon)n$,*

$$\frac{1}{|S|}\sum_{i\in S} X_i X_i^\top \succeq (1-\sqrt{\epsilon})\mathbf{\Sigma}^\star .$$

Finally, we show our main helper lemma, which is used to prove Assumption 7.2 holds.

Lemma 315. *Let $\epsilon > 0$ be sufficiently small. Let $X_1, \ldots, X_n$ be n samples from a 2-to-4 hypercontractive distribution $\mathcal{D}$ with parameter C and second moment $\mathbf{\Sigma}^\star$. Then, there exist universal constants $c, C_{\mathrm{est}} > 0$ so that if*

$$n \geq c\left(\frac{d\log d}{\epsilon^4} + \frac{d^2\log(d/\epsilon)}{\epsilon^3}\right),$$

then with probability 0.99*, for every* $u \in \mathbb{R}^d$*, there exists an* $G \subseteq [n]$ *satisfying* $|G| \geq (1-\epsilon^2)n$*, and*

$$\left\| \frac{1}{|G|} \sum_{i \in G} \langle X_i, u \rangle^2 X_i X_i^\top \right\|_{\mathrm{op}} \leq C_{\mathrm{est}} L \|u\|_{\mathbf{\Sigma}^\star}^2 \ .$$

Proof. Without loss of generality (by scale invariance), it suffices to prove this for all u with $\|u\|_{\mathbf{\Sigma}^\star} = 1$. First, by Markov's inequality with $\mathbb{E}_{X \sim \mathcal{D}}[\|X\|_2^2] = \mathrm{Tr}[\mathbf{\Sigma}^\star] \leq Ld$, we have that

$$\Pr_{X \sim \mathcal{D}} \left[\|X\|_2^2 \geq \frac{20Ld}{\epsilon^2} \right] \leq \frac{\epsilon^2}{20} \ .$$

Hence, by Bernstein's inequality, we have that with probability 0.999,

$$\frac{|\{i : \|X_i\|_2^2 \geq 20Ld\epsilon^{-2}\}|}{n} \leq \frac{\epsilon^2}{10} \ . \tag{G.21}$$

Condition on this event holding for the rest of the proof.

For any vector $u \in \mathbb{R}^d$ with $\|u\|_{\mathbf{\Sigma}^\star} = 1$, let $H_u \subset \mathbb{R}^d$ be the set given by

$$H_u = \left\{ x \in \mathbb{R}^d : \langle x, u \rangle^2 \geq \frac{10 C_{2\to 4}^{1/2}}{\epsilon} \right\} \ .$$

Note that by Chebyshev's inequality, since $\mathbb{E}_{X \sim \mathcal{D}}[\langle x, u \rangle^4] \leq C_{2\to 4} \|u\|_{\mathbf{\Sigma}^\star}^4 \leq C_{2\to 4}$, $\Pr_{X \sim D}[X \in H_u] \leq \frac{\epsilon^2}{100}$. Furthermore, the collection of sets $\{H_u\}_{\|u\|_{\mathbf{\Sigma}^\star}=1}$ has VC dimension $O(d)$, as each H_u can be expressed as a restricted intersection of parallel halfspaces, and it is well-known that VC dimension is additive under intersection. Therefore, by the VC inequality, we know that if $n \geq c \frac{d \log d}{\epsilon^4}$ for sufficiently large constant c, with probability 0.999, we have that

$$\sup_{H_u} \frac{|\{i : X_i \in H_u\}|}{n} \leq \frac{\epsilon^2}{50} \ . \tag{G.22}$$

Condition on this event holding for the rest of the proof. All expectations throughout the remainder of the proof are taken with respect to $X \sim \mathcal{D}$ for notational simplicity.

For any fixed u, we define the truncated fourth moment (contracted in the direction u) by

$$\mathbf{A}_i = \mathbf{A}_i(u) = \langle X_i, u \rangle^2 X_i X_i^\top \mathbf{1} \left[\langle X_i, u \rangle^2 \leq \frac{20 C_{2\to 4}^{1/2}}{\epsilon} \text{ and } \|X_i\|_2^2 \leq \frac{20Ld}{\epsilon^2} \right] .$$

Note that

$$\|\mathbb{E}[\mathbf{A}_i]\|_{\mathrm{op}} \leq \left\| \mathbb{E}\left[\langle X_i, u \rangle^2 X_i X_i^\top \right] \right\|_{\mathrm{op}} = \sup_{\|v\|_2 = 1} \mathbb{E}\left[\langle X_i, u \rangle^2 \langle X_i, v \rangle^2 \right] \leq C_{2\to 4} L \|u\|_{\mathbf{\Sigma}^\star}^2 \ ,$$

by Cauchy-Schwarz and hypercontractivity. Moreover, by construction, the spectral norm of $\mathbf{A}_i$ is bounded almost surely by $\frac{400LC_{2\to4}^{1/2}d}{\epsilon^3}$. Hence, by a matrix Chernoff bound, we get that if $n \geq c\frac{d^2\log(d/\epsilon)}{\epsilon^3}$ for a sufficiently large constant c, then with probability 0.999, we have that

$$\left\|\frac{1}{n}\sum_{i=1}^{n}\mathbf{A}_i(u)\right\|_{\mathrm{op}} \leq 2C_{2\to4}L\left\|u\right\|_{\mathbf{\Sigma}^\star}^2 . \tag{G.23}$$

for all u in a $\mathrm{poly}(\frac{\epsilon}{d})$-net of the unit sphere in the $\mathbf{\Sigma}^\star$ norm (which has cardinality $(\frac{d}{\epsilon})^{O(d)}$ by Theorem 1.13 of [456]). Because we are union bounding over $\mathrm{poly}(d,\epsilon^{-1})$ samples, we have with high probability that all $\left\|X_i\right\|_{(\mathbf{\Sigma}^\star)^{-1}} = \mathrm{poly}(d,\epsilon^{-1})$. Hence, it is straightforward to show that the bound (G.23) over our net implies for all $\left\|u\right\|_{\mathbf{\Sigma}^\star} = 1$,

$$\left\|\frac{1}{n}\sum_{i=1}^{n}\left\langle X_i,u\right\rangle^2 X_iX_i^\top \mathbf{1}\left[X_i \notin H_u \text{ and } \left\|X_i\right\|_2^2 \leq \frac{10C_{2\to4}^{1/2}Ld}{\epsilon}\right]\right\|_{\mathrm{op}} \leq 2C_{2\to4}L\,, \tag{G.24}$$

Combining (G.21), (G.22), and (G.24) implies that for every u, the set $G = \{i : X_i \in H_u \text{ and } \left\|X_i\right\|_2^2 \leq \frac{20Ld}{\epsilon^2}\}$ satisfies the conditions of the lemma. □

We are now ready to prove Proposition 33.

Proof of Proposition 33. Condition 3 of Assumption 7 follows directly from Markov's inequality, since it is asking about the empirical average over G of $\delta^2 \sim \mathcal{D}_\delta$; the adversary removing points can only affect this upper bound by a constant factor (due to renormalization).

Next, let $[n] = G \cup B$ be the canonical decomposition of the corrupted set of samples. By two applications of Lemma 313, with probability at least 0.99, there exists a set $G' \subset G$ of size $|G'| \geq (1-\epsilon)|G| \geq (1-2\epsilon)n$ so that

$$\frac{1}{|G'|}\sum_{i\in G'}X_iX_i^\top \preceq \frac{3}{2}\mathbf{\Sigma}^\star\,, \tag{G.25}$$

$$\frac{1}{|G'|}\sum_{i\in G'}\delta_i^2X_iX_i^\top \preceq \frac{3}{2}\sigma^2\mathbf{\Sigma}^\star\,. \tag{G.26}$$

Condition on the event that such a G' exists for the remainder of the proof, and also condition on the event that Lemma 315 is satisfied. By a union bound, these events happen together with probability at least 0.9. We will show that this G' will satisfy the conditions of the lemma. The upper bound in Condition 1 of Assumption 7 is immediate, and similarly, the lower bound follows from Lemma 314 and a standard convexity argument (since the vertices of the polytope defining saturated weights are subsets of cardinality $(1-O(\epsilon))|G|$).

It thus remains to prove Condition 2 of Assumption 7. To do so, we will first prove (8.39) is satisfied with high probability. By Lemma 315 (adjusting by a factor of α in the definition of ϵ^2),

there exists a set $G'' \subseteq G'$ so that $|G''| \geq (1 - \frac{\epsilon^2}{\alpha})|G'|$ so that

$$\left\| \frac{1}{|G''|} \sum_{i \in G''} \langle X_i, \theta - \theta^\star \rangle^2 X_i X_i^\top \right\|_{\mathrm{op}} \leq C_{\mathrm{est}} L \left\| \theta - \theta^\star \right\|_{\mathbf{\Sigma}^\star}^2 .$$

Hence, for this choice of G'', we have

$$\begin{aligned}
\mathrm{Cov}_{\tilde{w}} \left(\{ g_i(\theta) \}_{i \in G''} \right) &= \sum_{i \in G''} \tilde{w}_i \left(\langle X_i, \theta - \theta^\star \rangle + \delta_i \right)^2 X_i X_i^\top \\
&\preceq 2 \sum_{i \in G''} \tilde{w}_i \langle X_i, \theta - \theta^\star \rangle^2 X_i X_i^\top + 2 \sum_{i \in G} \tilde{w}_i \delta_i^2 X_i X_i^\top \\
&\preceq \frac{2(1+2\epsilon)}{|G''|} \sum_{i \in G''} \langle X_i, \theta - \theta^\star \rangle^2 X_i X_i^\top + \frac{2(1+2\epsilon)}{|G'|} \sum_{i \in G'} \delta_i^2 X_i X_i^\top \\
&\preceq \frac{2(1+2\epsilon)}{|G'|} \sum_{i \in G''} \langle X_i, \theta - \theta^\star \rangle^2 X_i X_i^\top + 3\sigma^2 \mathbf{\Sigma}^\star \\
&\preceq 2(1+2\epsilon) C_{\mathrm{est}} L \left(\left\| \theta - \theta^\star \right\|_{\mathbf{\Sigma}^\star}^2 + \sigma^2 \right) \mathbf{I} .
\end{aligned}$$

where the last line follows from (G.26). By suitably adjusting the choice of C_{est}, this proves that (8.39) is satisfied for this choice of G''. Finally, we claim that (8.39) implies (8.38) via standard techniques from the robust mean estimation literature, e.g. in the proof of Lemma 3.2 in [194]. □

G.6.2 Proof of Proposition 36

In this section, we state FastCovFilter and prove Proposition 36, restated for convenience.

Proposition 36. *There is an algorithm,* FastCovFilter *(Algorithm 92), taking inputs* $\mathbf{V} := \{v_i\}_{i \in [n]} \in \mathbb{R}^{n \times d}$*, saturated weights* $w \in \Delta^n$ *with respect to bipartition* $[n] = G \cup B$ *with* $|B| = \epsilon n$*,* $\delta \in (0, 1]$*, and* $R \geq 0$ *with the promise that*

$$\left\| \sum_{i \in G} \frac{1}{|G|} v_i v_i^\top \right\|_{\mathrm{op}} \leq R.$$

Then, with probability at least $1 - \delta$*,* FastCovFilter *returns saturated* $w' \in \Delta^n$ *such that*

$$\left\| \sum_{i \in [n]} w_i' v_i v_i^\top \right\|_{\mathrm{op}} \leq 5R.$$

The runtime of FastCovFilter *is*

$$O\left(nd \log^3(n) \log\left(\frac{n}{\delta} \right) \right).$$

Before proving Proposition 36, we require three helper facts.

Algorithm 92: $\mathsf{FastCovFilter}(\mathbf{V}, w, \delta, R)$

Input: $\mathbf{V} := \{v_i\}_{i\in[n]} \in \mathbb{R}^{n\times d}$, saturated weights $w \in \Delta^n$ with respect to bipartition $[n] = G \cup B$ with $|B| = \epsilon n$ for sufficiently small ϵ, $\delta \in (0,1)$, $R \geq \left\|\sum_{i\in G} \frac{1}{|G|} v_i v_i^\top\right\|_{\mathrm{op}}$;

Output: With probability $\geq 1-\delta$, saturated w' with respect to bipartition $G \cup B$, with

$$\left\|\sum_{i\in[n]} w_i' v_i v_i^\top\right\|_{\mathrm{op}} \leq 5R.$$

;

Remove all $i \in [n]$ with $\|v_i\|_2^2 \geq nR$, $n \leftarrow$ new dataset size;

$T \leftarrow O(\log^2 n)$ (for a sufficiently large constant), $t \leftarrow 0$, $w^{(0)} \leftarrow w$;

while $t < T$ *and* $\mathsf{Power}\left(\sum_{i\in[n]} w_i^{(t)} v_i v_i^\top, \frac{\delta}{2T}\right) > 2R$ **do**

  $\mathbf{M}_t \leftarrow \sum_{i\in[n]} w_i^{(t)} v_i v_i^\top$, $\mathbf{Y}_t \leftarrow \mathbf{M}_t^{\log d}$;

  Sample $N_{\mathrm{dir}} = O(\log \frac{n}{\delta})$ (for a sufficiently large constant) vectors $\{u_j\}_{j\in[N_{\mathrm{dir}}]} \in \mathbb{R}^d$ each with independent entries ± 1. Let $\tilde{u}_j \leftarrow \mathbf{Y}_t u_j$ for all $j \in [N_{\mathrm{dir}}]$.;

  for $j \in [N_{\mathrm{dir}}]$ **do**

    $\tau_i \leftarrow \langle v_i, \tilde{u}_j\rangle^2$ for all $i \in [n]$;

    $\tau_{\max} \leftarrow \max_{i\in[n]|w_i\neq 0} \tau_i$;

    while $\sum_{i\in[n]} w_i^{(t)} \tau_i \geq 2R\|\tilde{u}_j\|_2^2$ **do**

      $w_i^{(t)} \leftarrow \left(1 - \frac{\tau_i}{\tau_{\max}}\right) w_i^{(t)}$ for all $i \in [n]$;

  $w^{(t+1)} \leftarrow w^{(t)}$, $t \leftarrow t+1$;

Return: $w^{(t)}$;

Fact 40 (Theorem 1, [404])**.** *For any $\delta \in (0,1]$ and $\mathbf{M} \in \mathbb{S}^d_{\geq 0}$, there is an algorithm, $\mathsf{Power}(\mathbf{M}, \delta)$, which returns with probability at least $1-\delta$ a value V such that $\lambda_{\max}(\mathbf{M}) \geq V \geq 0.9\lambda_{\max}(\mathbf{M})$. The algorithm costs $O(\log \frac{d}{\delta})$ matrix-vector products through $\mathbf{M}$ plus $O(d \log \frac{d}{\delta})$ additional runtime.*

Fact 41 (Lemma 7, [289])**.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d_{\geq 0}$ with $\mathbf{A} \succeq \mathbf{B}$, and $p \in \mathbb{N}$. Then $\mathrm{Tr}(\mathbf{A}^{p-1}\mathbf{B}) \geq \mathrm{Tr}(\mathbf{B}^p)$.*

Fact 42. *For any $\alpha \geq 0$ and $\mathbf{A} \in \mathbb{S}^d_{\geq 0}$,*

$$\alpha \mathrm{Tr}(\mathbf{A}^{2\log d}) \leq \mathrm{Tr}(\mathbf{A}^{2\log d+1}) + d\alpha^{2\log d+1}.$$

Proof. Every eigenvalue λ of $\mathbf{A}$ is either at least α (and hence $\lambda^{2\log d+1} \geq \alpha\lambda^{2\log d}$) or not (and hence $\alpha^{2\log d+1} \geq \alpha\lambda^{2\log d}$). Both of these cases are accounted for by the right hand side. □

Proof of Proposition 36. We discuss correctness, runtime, and the failure probability separately.

Correctness. First, Line 3 is correct because these indices cannot belong to G as they would certify a violation to the operator norm bound in the direction of v, so this preserves saturation. Next, it is clear by Fact 40 that if the algorithm ever ends because Power returns too small a

number, the output is correct, so it suffices to handle the other case. Define the potential function $\Phi_t := \mathrm{Tr}(\mathbf{Y}_t^2)$. Our main goal is to show that in every iteration the algorithm runs, Φ_t decreases substantially. To this end, we claim that after all runs of Lines 8-14 of FastCovFilter have finished, we have in all randomly sampled directions $j \in [N_{\mathrm{dir}}]$ the guarantee

$$\left\langle \tilde{u}_j \tilde{u}_j^\top, \sum_{i \in [n]} w_i^{(t)} v_i v_i^\top \right\rangle \le 2R \left\| \tilde{u}_j \right\|_2^2 . \tag{G.27}$$

This is immediate from the termination condition on Line 11 for each $j \in [N_{\mathrm{dir}}]$, the fact that weights are monotone nonincreasing throughout the whole algorithm, and that the left hand side of (G.27) is monotone nonincreasing as a function of the weights. Next, by the Johnson-Lindenstrauss lemma of [6], for a sufficiently large N_{dir} with probability at least $1 - \frac{\delta}{4T}$,

$$\frac{1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \langle v_i, \tilde{u}_j \rangle^2 = \frac{1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} v_i^\top \mathbf{Y}_t \tilde{u}_j \tilde{u}_j^\top \mathbf{Y}_t v_i \in [0.95, 1.05] \left\langle v_i v_i^\top, \mathbf{Y}_t^2 \right\rangle \text{ for all } i \in [n].$$

Condition on this event for all runs of Lines 8-14 throughout the algorithm for the remainder of the proof. Combining this guarantee with (G.27), we have that after Lines 8-14 terminate,

$$\begin{aligned}
\left\langle \mathbf{Y}_t^2, \sum_{i \in [n]} w_i^{(t)} v_i v_i^\top \right\rangle &= \sum_{i \in [n]} w_i^{(t)} \left\langle v_i v_i^\top, \mathbf{Y}_t^2 \right\rangle \\
&\le \frac{1.1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \sum_{i \in [n]} w_i^{(t)} \left\langle v_i, \tilde{u}_j^2 \right\rangle \\
&= \frac{1.1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \left\langle \tilde{u}_j \tilde{u}_j^\top, \sum_{i \in [n]} w_i^{(t)} v_i v_i^\top \right\rangle \le \frac{2.2R}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \left\| \tilde{u}_j \right\|_2^2 .
\end{aligned}$$

Next, by the Johnson-Lindenstrauss lemma of [6], since all $\{u_j\}_{j \in [N_{\mathrm{dir}}]}$ were sampled independently of $\mathbf{M}_t$, we have with probability at least $1 - \frac{\delta}{4T}$ that

$$\frac{1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} \left\| \tilde{u}_j \right\|_2^2 = \frac{1}{N_{\mathrm{dir}}} \sum_{j \in [N_{\mathrm{dir}}]} u_j^\top \mathbf{Y}_t^2 u_j \le 1.1 \mathrm{Tr}(\mathbf{Y}_t^2).$$

Conditioning on this event in every iteration, at the start of the next iteration, we will have

$$\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \le 2.5 R \mathrm{Tr}(\mathbf{Y}_t^2). \tag{G.28}$$

We now show how (G.28) implies a rapid potential decrease:

$$\begin{aligned}\Phi_{t+1} = \mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d}\right) &\le \frac{1}{2.8R}\mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d+1}\right) + d(2.8R)^{2\log d} \\ &\le \frac{1}{2.8R}\mathrm{Tr}\left(\mathbf{Y}_t^2\mathbf{M}_{t+1}\right) + d(2.8R)^{2\log d} \le 0.9\Phi_t + d(2.8R)^{2\log d}.\end{aligned}$$

In the first inequality, we used Lemma 42 with $\alpha = 2.8R$; in the second, we used Fact 41 with $\mathbf{A} = \mathbf{M}_t$, $\mathbf{B} = \mathbf{M}_{t+1}$, and $p = 2\log d + 1$. The last inequality applied (G.28). The above display implies that until $\Phi_t \le 20d(2.8R)^{2\log d}$, the potential is decreasing by a constant factor every iteration, and $\Phi_0 \le d\lambda_{\max}(\mathbf{M}_0)^{2\log d} \le d(nR)^{2\log d}$, so within $O(\log^2 n)$ iterations we will have

$$\Phi_t = \mathrm{Tr}(\mathbf{M}_t^{2\log d}) \le 20d(2.8R)^{2\log d}.$$

At this point, it is clear the operator norm of $\mathbf{M}_t$ achieves the desired bound of $5R$. It remains to show that all weight removals in Lines 11-13 were safe throughout the algorithm. Here we use Lemma 130: it suffices to show that throughout the algorithm,

$$\sum_{i\in G} w_i^{(t)}\tau_i \le R\left\|\tilde{u}_j\right\|_2^2, \tag{G.29}$$

because then whenever Line 11 fails, the scores are safe with respect to the weights and Lemma 130 applies. However, (G.29) follows from the assumption on $\left\|\sum_{i\in G}\frac{1}{|G|}v_iv_i^\top\right\|_{\mathrm{op}}$, yielding

$$\sum_{i\in G} w_i^{(t)}\tau_i \le \sum_{i\in G}\frac{1}{|G|}\tau_i = \left\langle \tilde{u}_j\tilde{u}_j^\top, \sum_{i\in G}\frac{1}{|G|}v_iv_i^\top\right\rangle \le R\left\|\tilde{u}_j\right\|_2^2.$$

Runtime. The cost of all lines other than Lines 6-7 and the repeated loops of Lines 11-13 clearly fall within the budget. To implement Lines 6-7, we never need to form the matrices $\mathbf{M}_t$ or $\mathbf{Y}_t$, and instead form all $\{\tilde{u}_j\}_{j\in[N_{\mathrm{dir}}]}$ in time

$$O\left(nd\log d\log\frac{n}{\delta}\right)$$

implicitly through matrix-vector multiplications with $\mathbf{M}_t$, each of which take time $O(nd)$. To implement Lines 11-13, let $\bar{w}$ denote the value of $w^{(t)}$ right after an execution of Line 10. We wish to determine the smallest value K such that

$$\sum_{i\in[n]}\left(1-\frac{\tau_i}{\tau_{\max}}\right)^K \bar{w}_i\tau_i \le 2R\left\|\tilde{u}_j\right\|_2^2.$$

Checking if the above display holds for a particular guess of K clearly takes $O(n)$ time, and we can

upper bound K by the following inequality:

$$\sum_{i\in[n]}\left(1-\frac{\tau_i}{\tau_{\max}}\right)^K \bar{w}_i\tau_i \le \sum_{i\in[n]}\exp\left(-\frac{K\tau_i}{\tau_{\max}}\right)\bar{w}_i\tau_i \le \frac{1}{eK}\sum_{i\in[n]}\bar{w}_i\tau_{\max} \le \frac{\tau_{\max}}{K}.$$

Here the second inequality used $x\exp(-Cx) \le \frac{1}{eC}$ for all nonnegative x, C, where we chose $C = \frac{K}{\tau_{\max}}$ and $x = \tau_i$. Now since $\tau_{\max} \le \|\tilde{u}_j\|_2^2 \|v_i\|_2^2 \le nR\|\tilde{u}_j\|_2^2$ by Cauchy-Schwarz, we have that $K = O(n)$ as desired. At this point, a binary search on K suffices, so all loops take time $O(n\log n)$.

Failure probability. The only randomness used in the algorithm appears in the guarantees of Power and the guarantees of the Johnson-Lindenstrauss projections. Taking a union bound over T iterations shows these all succeed with probability at least $1-\delta$. □

G.7 Concentration

G.7.1 Sub-Gaussian concentration

We use the following concentration facts on sub-Gaussian distributions following from standard techniques, and give an application bounding Schatten-norm deviations.

Lemma 316. *Under Assumption 10, there are universal constants C_1, C_2 such that*

$$\Pr\left[\sup_{\substack{v\in\mathbb{R}^d\\ \|v\|_2=1}}\left|v^\top\left(\frac{1}{n}\sum_{i\in G'}X_iX_i^\top - \mathbf{\Sigma}\right)v\right| - tv^\top\mathbf{\Sigma}v > 0\right] \le \exp\left(C_1d - C_2n\min(t,t^2)\right).$$

Proof. By observing (8.71), it is clear that the random vector $\widetilde{X} = \mathbf{\Sigma}^{-\frac{1}{2}}X$ for $X\sim\mathcal{D}$ has covariance $\mathbf{I}$ and sub-Gaussian proxy $c\mathbf{I}$. For any fixed unit vector u, by Lemma 1.12 of [456], the random variable $(u^\top\widetilde{X})^2 - 1$ is sub-exponential with parameter $16c$, so by Bernstein's inequality (Theorem 1.13, [456]), defining $\widetilde{X}_i = \mathbf{\Sigma}^{-\frac{1}{2}}X_i$ for each $X_i\sim\mathcal{D}$,

$$\Pr\left[\left|u^\top\left(\frac{1}{n}\sum_{i\in G'}\widetilde{X}_i\widetilde{X}_i^\top - \mathbf{I}\right)u\right| > \frac{t}{2}\right] \le \exp\left(-\frac{n}{2^{11}c^2}\min(t,t^2)\right).$$

For shorthand define $\mathbf{M} := \frac{1}{n}\sum_{i\in G'}\widetilde{X}_i\widetilde{X}_i^\top$, and let $\mathcal{N}$ be a maximal $\frac{1}{4}$-net of the unit ball (as measured in ℓ_2 distance). By Lemma 1.18 of [456], $|\mathcal{N}| \le 12^d$, so by a union bound,

$$\Pr\left[\sup_{u\in\mathcal{N}}\left|u^\top(\mathbf{M}-\mathbf{I})u\right| > \frac{t}{2}\right] \le \exp\left(3d - \frac{n}{2^{11}c^2}\min(t,t^2)\right).$$

Next, by a standard application of the triangle inequality (see e.g. Exercise 4.3.3, [522])

$$\sup_{\substack{v\in\mathbb{R}^d\\ \|v\|_2=1}} \left|v^\top(\mathbf{M}-\mathbf{I})v\right| \le 2\sup_{u\in\mathcal{N}} \left|u^\top(\mathbf{M}-\mathbf{I})u\right| \le t$$

with probability at least $1-\exp\left(C_1 d - C_2 n\min(t,t^2)\right)$ for appropriate C_1, C_2. The conclusion follows since its statement is scale invariant, so it suffices to show as we have that

$$\Pr\left[\sup_{\substack{v\in\mathbb{R}^d\\ \|v\|_{\mathbf{\Sigma}}=1}} \left| v^\top\left(\frac{1}{n}\sum_{i\in G'} X_iX_i^\top - \mathbf{\Sigma}\right)v\right| - tv^\top\mathbf{\Sigma}v > 0\right] \le \exp\left(C_1 d - C_2 n\min(t,t^2)\right).$$

□

Corollary 69. *Let $p\ge 2$. Under Assumption 10, there are universal constants C_1, C_2 with*

$$\Pr\left[\left\|\frac{1}{n}\sum_{i\in G'} X_iX_i^\top - \mathbf{\Sigma}\right\|_p > t\left\|\mathbf{\Sigma}\right\|_p\right] \le \exp\left(C_1 d - C_2 n\min(t,t^2)\right).$$

Proof. Suppose the event in Lemma 316 does not hold, which happens with probability at least $1-\exp(C_1 d - C_2 n\min(t,t^2))$. Define for shorthand $\mathbf{M} := \frac{1}{n}\sum_{i\in G'} X_iX_i^\top - \mathbf{\Sigma}$ and let its spectral decomposition be $\sum_{j\in[d]}\lambda_j v_jv_j^\top$. By the triangle inequality and Fact 22,

$$\begin{aligned}\left\|\mathbf{M}\right\|_p &\le \sum_{j\in[d]}\frac{|\lambda_j|^{p-1}}{\left\|\mathbf{M}\right\|_p^{p-1}}\left|v_j^\top\left(\frac{1}{n}\sum_{i\in G'} X_iX_i^\top - \mathbf{\Sigma}\right)v_j\right| \\ &\le t\sum_{j\in[d]}\frac{|\lambda_j|^{p-1}}{\left\|\mathbf{M}\right\|_p^{p-1}}v_j^\top\mathbf{\Sigma}v_j = t\left\langle\sum_{j\in[d]}\frac{|\lambda_j|^{p-1}}{\left\|\mathbf{M}\right\|_p^{p-1}}v_jv_j^\top, \mathbf{\Sigma}\right\rangle \le t\left\|\mathbf{\Sigma}\right\|_p.\end{aligned}$$

In the last inequality, we used that $\sum_{j\in[d]}\frac{|\lambda_j|^{p-1}}{\left\|\mathbf{M}\right\|_p^{p-1}}v_jv_j^\top$ has unit ℓ_q norm, and applied Fact 22. □

G.7.2 Concentration under weightings in $\mathfrak{S}_\epsilon^n$

We consider concentration of the empirical covariance under weightings which are not far from uniform, in spectral and Schatten senses.

Lemma 317. *Under Assumption 10, let $\delta\in[0,1]$, $p\ge 2$, and $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$ for a sufficiently large constant. Then for a universal constant C_3,*

$$\Pr\left[\exists w\in\mathfrak{S}_\epsilon^n \;\middle|\; \left\|\sum_{i\in G'} w_iX_iX_i^\top - \mathbf{\Sigma}\right\|_p > C_3\cdot\epsilon\log\epsilon^{-1}\left\|\mathbf{\Sigma}\right\|_p\right] \le \frac{\delta}{2}.$$

Proof. Because the vertices of $\mathfrak{S}_\epsilon^n$ are uniform over sets $S \subseteq G'$ with $|S| = (1-\epsilon)n$ (see e.g. Section 4.1, [175]), by convexity of the Schatten-p norm it suffices to prove

$$\Pr\left[\exists S \text{ with } |S| = (1-\epsilon)n \;\middle|\; \left\|\frac{1}{(1-\epsilon)n}\sum_{i\in S} X_i X_i^\top - \mathbf{\Sigma}\right\|_p > C_3 \cdot \epsilon \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p\right] \leq \frac{\delta}{4}.$$

For any fixed S, and recalling $|S^c| = \epsilon n$, we can decompose this sum as

$$\frac{1}{(1-\epsilon)n}\sum_{i\in S} X_i X_i^\top = \frac{1}{1-\epsilon}\left(\frac{1}{n}\sum_{i\in G'} X_i X_i^\top\right) - \frac{\epsilon}{1-\epsilon}\left(\frac{1}{|S^c|}\sum_{i\in S^c} X_i X_i^\top\right). \tag{G.30}$$

By applying Corollary 69, it follows that by setting $t = \frac{1-\epsilon}{2}\cdot \epsilon \log \epsilon^{-1}$ and our choice of n that

$$\Pr\left[\left\|\frac{1}{n}\sum_{i\in G'} X_i X_i^\top - \mathbf{\Sigma}\right\|_p > \frac{1-\epsilon}{2}\cdot \epsilon \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p\right] \leq \frac{\delta}{4}. \tag{G.31}$$

Moreover, for any fixed S^c, setting $t = \frac{1-\epsilon}{2}\cdot C_3 \log \epsilon^{-1}$ where C_3 is a sufficiently large constant, so that for sufficiently small ϵ, $t = \min(t, t^2)$,

$$\begin{aligned}\Pr\left[\left\|\frac{1}{\epsilon n}\sum_{i\in S^c} X_i X_i^\top - \mathbf{\Sigma}\right\|_p > \frac{1-\epsilon}{2}\cdot C_3 \cdot \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p\right] &\leq \exp\left(C_1 d - C_2 \epsilon n t\right) \\ &\leq \exp\left(-\left(\log \delta^{-1} + n\epsilon \log \epsilon^{-1}\right)\right) \\ &\leq \frac{\delta}{4\binom{n}{\epsilon n}}.\end{aligned} \tag{G.32}$$

Here, we used that $\log\binom{n}{\epsilon n} = O\left(n\epsilon \log \epsilon^{-1}\right)$. Finally, union bounding over all possible sets S^c imply that with probability at least $1 - \frac{\delta}{2}$, the following events hold:

$$\left\|\frac{1}{n}\sum_{i\in G'} X_i X_i^\top - \mathbf{\Sigma}\right\|_p < \frac{1-\epsilon}{2}\cdot \epsilon \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p,$$

$$\left\|\frac{1}{|S^c|}\sum_{i\in S^c} X_i X_i^\top - \mathbf{\Sigma}\right\|_p < \frac{1-\epsilon}{2}\cdot C_3 \cdot \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p \text{ for all } S \text{ with } |S| = (1-\epsilon)n.$$

Combining these bounds in the context of (G.30) after applying the triangle inequality, we have with probability at least $1 - \frac{\delta}{2}$ for all S the desired conclusion,

$$\left\|\frac{1}{(1-\epsilon)n}\sum_{i\in S} X_i X_i^\top - \mathbf{\Sigma}\right\|_p < C_3 \cdot \epsilon \log \epsilon^{-1} \left\|\mathbf{\Sigma}\right\|_p.$$

□

Corollary 70. *Under Assumption 10, let* $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$ *for a sufficiently large constant. For universal* C_3 *and all* $w \in \mathfrak{S}_\epsilon^n$*, with probability at least* $1-\frac{\delta}{2}$*,*

$$C_3\cdot\epsilon\log\epsilon^{-1}\mathbf{\Sigma} \succeq \sum_{i\in G'} w_i X_i X_i^\top - \mathbf{\Sigma} \succeq -C_3\cdot\epsilon\log\epsilon^{-1}\mathbf{\Sigma}.$$

Proof. Consider any unit vector $v\in\mathbb{R}^d$. By similar arguments as in (G.31), (G.32), and applying a union bound over all S with $|S| = (1-\epsilon)n$, with probability at least $1-\frac{\delta}{2}$, it follows from Lemma 316 that

$$\left|v^\top\left(\frac{1}{n}\sum_{i\in G'} X_iX_i^\top - \mathbf{\Sigma}\right)v\right| < \frac{1-\epsilon}{2}\cdot\epsilon\log\epsilon^{-1}v^\top\mathbf{\Sigma}v, \tag{G.33}$$

$$\left|v^\top\left(\frac{1}{|S^c|}\sum_{i\in S^c} X_iX_i^\top - \mathbf{\Sigma}\right)v\right| < \frac{1-\epsilon}{2}\cdot C_3\cdot\log\epsilon^{-1}v^\top\mathbf{\Sigma}v\,. \tag{G.34}$$

Therefore, again using the formula (G.30) and the triangle inequality yields the desired conclusion for all directions v, which is equivalent to the spectral bound of the lemma statement. □

G.8 Deferred proofs from Section 8.11.1

G.8.1 Robust univariate variance estimation

In this section, we prove Lemma 147, which allows us to robustly estimate the quadratic form of a vector in the covariance of a sub-Gaussian distribution from corrupted samples. Algorithm 93 is folklore, and intuitively very simple; it projects all samples onto u, throws away the 2ϵ fraction of points with largest magnitude in this direction, and takes the mean of the remaining set.

Algorithm 93: Univariate variance estimation: $\mathsf{1DRobustVariance}(\{X_i\}_{i\in[n]}, \epsilon, u)$

Input: $\{X_i\}_{i\in[n]}$, $\epsilon > 0$, and a unit vector u;
Let $a_i = \langle X_i, u\rangle^2$ for $i = 1,\dots,n$;
Sort the a_i in increasing order. WLOG assume $a_1 \le a_2 \le \dots \le a_n$.;
Return: $\sigma_u^2 = \frac{1}{(1-2\epsilon)n}\sum_{i=1}^{(1-2\epsilon)n} a_i$;

We require the following helper fact.

Fact 43. *Let* Z *be a sub-exponential random variable with parameter at most* λ[3]*, and let* $\epsilon\in[0,1]$*. Then, for any event* E *with* $\Pr[Z\in E]\le\epsilon$*,* $|\mathbb{E}[Z\cdot_{1,1}[Z\in E]]| \le 2\lambda\epsilon\log\epsilon^{-1}$*.*

[3]We say mean-zero Z is sub-exponential with parameter λ if $\forall|s|\le\lambda^{-1}$, $\mathbb{E}[\exp(sZ)]\le\exp(\frac{s^2\lambda^2}{2})$.

Proof. We have by Hölder's inequality that for any $p, q \geq 1$ with $p^{-1} + q^{-1} = 1$,

$$|\mathbb{E}\left[Z \cdot_{1,1} [Z \in E]\right]| \leq \mathbb{E}[|Z|^p]^{1/p} \cdot \epsilon^{1/q} \leq 2\lambda p \cdot \epsilon^{1/q}.$$

The second inequality is Lemma 1.10 [456]. Setting $p = \log \epsilon^{-1}$ yields the result. □

Lemma 147. *Under Assumption 10, let $\delta \in [0,1]$, $n = \Omega\left(\frac{\log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$, and $u \in \mathbb{R}^d$ be a fixed unit vector. Algorithm 93,* 1DRobustVariance*, takes input $\{X_i\}_{i\in[n]}$, u, and ϵ, and outputs σ_u^2 with $|u^\top \mathbf{\Sigma} u - \sigma_u^2| < C u^\top \mathbf{\Sigma} u \cdot \epsilon \log \epsilon^{-1}$ with probability at least $1 - \delta$, and runs in time $O(nd + n \log n)$, for C a fixed multiple of the parameter c in Assumption 10.*

Proof. The runtime claim is immediate; we now turn our attention to correctness. We follow notation of Assumption 10, and in a slight abuse of notation, also define $a_i = \langle X_i, u\rangle^2$ for $i \in G'$. First, for $X \sim \mathcal{D}$, then $\langle u, X\rangle^2 - u^\top \mathbf{\Sigma} u$ is sub-exponential with parameter at most $16cu^\top \mathbf{\Sigma} u$ (Lemma 1.12, [456]). By Bernstein's inequality, we have that if $X \sim \mathcal{D}$, then for all $t \geq 1$,

$$\Pr\left[\langle X, u\rangle^2 > 32ctu^\top \mathbf{\Sigma} u\right] \leq \exp(-t)\ . \tag{G.35}$$

Using this in a standard Chernoff bound, we have that with probability $1 - \frac{\delta}{2}$,

$$\frac{\left|\{i \in G' : a_i > 64c \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u\}\right|}{n} \leq \epsilon\ . \tag{G.36}$$

Let $T = 64c \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u$, and let Y be distributed as $(\langle u, X\rangle^2 - u^\top \mathbf{\Sigma} u) \cdot_{1,1} [\langle u, X\rangle^2 \leq T]$, where $X \sim \mathcal{D}$. We observe $Y - \mathbb{E}[Y]$ is also sub-exponential with parameter $16cu^\top \mathbf{\Sigma} u$, and that by Fact 43,

$$|\mathbb{E}[Y]| \leq 32cu^\top \mathbf{\Sigma} u \epsilon \log \epsilon^{-1}. \tag{G.37}$$

Define the interval $I = [0, T]$ and let S be the set of points in $[n]$ that survive the truncation procedure, so that $\sigma_u^2 = \frac{1}{|S|}\sum_{i\in S} a_i$. Given event (G.36), $a_i \in I$ for all $i \in S$, since there are at most ϵn points in G outside I, and $|B| \leq \epsilon n$. We decompose the deviation as follows:

$$\begin{aligned}
\sum_{i\in S} a_i - |S| u^\top \mathbf{\Sigma} u &= \sum_{i\in G\cap S} (a_i - u^\top \mathbf{\Sigma} u) + \sum_{i\in B\cap S} (a_i - u^\top \mathbf{\Sigma} u) \\
&= \sum_{i\in G'\cap I} (a_i - u^\top \mathbf{\Sigma} u) + \sum_{i\in B\cap S} (a_i - u^\top \mathbf{\Sigma} u) \\
&\quad - \sum_{i\in (G'\setminus G)\cap I} (a_i - u^\top \mathbf{\Sigma} u) - \sum_{i\in (G\cap I)\setminus S} (a_i - u^\top \mathbf{\Sigma} u).
\end{aligned} \tag{G.38}$$

Here we overloaded $i \in I$ to mean that a_i lies in the interval I, and conditioned on S lying entirely in I. We bound each of these terms individually. First, for all $i \in G' \cap I$, conditioning on (G.36) (i.e. all $a_i \in I$), $a_i - u^\top \mathbf{\Sigma} u$ is an independent sample from Y. Thus, by (G.37) and Bernstein's

inequality,

$$\left|\frac{1}{|G'\cap I|}\sum_{i\in G'\cap I}(a_i-u^\top\mathbf{\Sigma}u)\right| \le \left|\frac{1}{|G'\cap I|}\sum_{i\in G'\cap I}(a_i-u^\top\mathbf{\Sigma}u)-\mathbb{E}[Y]\right| + 32cu^\top\mathbf{\Sigma}u\epsilon\log\epsilon^{-1} \le 64c\cdot u^\top\mathbf{\Sigma}u\epsilon\log\epsilon^{-1}, \tag{G.39}$$

with (conditional) probability at least $1-\frac{\delta}{2}$. By a union bound, both events occur with probability at least $1-\delta$; condition on this for the remainder of the proof. Under this assumption, we control the other three terms of (G.38). Observe that $|B\cap S|\le\epsilon n$, $|(G'\setminus G)\cap I|\le\epsilon n$, and $|(G\cap I)\setminus S|\le\epsilon n$. Further, by definition of I, every summand is at most $64c\log\epsilon^{-1}\cdot u^\top\mathbf{\Sigma}u$. Thus,

$$\left|\sum_{i\in B\cap S}(a_i-u^\top\mathbf{\Sigma}u)\right| \le 64c\epsilon n\log\epsilon^{-1}\cdot u^\top\mathbf{\Sigma}u, \tag{G.40}$$

$$\left|\sum_{i\in (G'\setminus G)\cap I}(a_i-u^\top\mathbf{\Sigma}u)\right| \le 64c\epsilon n\log\epsilon^{-1}\cdot u^\top\mathbf{\Sigma}u. \tag{G.41}$$

$$\left|\sum_{i\in (G'\cap I)\setminus S}(a_i-u^\top\mathbf{\Sigma}u)\right| \le 64c\epsilon n\log\epsilon^{-1}\cdot u^\top\mathbf{\Sigma}u. \tag{G.42}$$

Combining (G.39), (G.40), (G.41), and (G.42) in derivation (G.38) and dividing by $|S|$ yields the claim. □

Finally, we also give an alternative set of conditions under which we can certify correctness of 1DRobustVariance. Specifically, this assumption will be useful in lifting indpendence assumptions between u and our samples $\{X_i\}_{i\in[n]}$ in repeated calls within Algorithm 94.

Assumption 16. *Under Assumption 10, let the following conditions hold for universal constant C_4:*

$$C_4\epsilon\log\epsilon^{-1}\cdot\mathbf{\Sigma}\succeq\frac{1}{n}\sum_{i\in G'}X_iX_i^\top-\mathbf{\Sigma}\succeq-C_4\epsilon\log\epsilon^{-1}\cdot\mathbf{\Sigma}, \tag{G.43}$$

$$C_4\log\epsilon^{-1}\cdot\mathbf{\Sigma}\succeq\sum_{i\in G'}w_i\left(X_iX_i^\top-\mathbf{\Sigma}\right)\succeq-C_4\log\epsilon^{-1}\cdot\mathbf{\Sigma}\ \textit{for all}\ w\in\mathfrak{S}^n_{1-\epsilon}. \tag{G.44}$$

Note that (G.44) is a factor ϵ weaker in its guarantee than Corollary 70, and is over weights in a different set $\mathfrak{S}^n_{1-\epsilon}$. Standard sub-Gaussian concentration (i.e. an unweighted variant of Corollary 70) and modifying the proof of Corollary 70 to take the constraint set $\mathfrak{S}^n_{1-\epsilon}$ and normalizing over vertex sets of size ϵn yield the following conclusion.

Lemma 318. *Let $n=\Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$ for a sufficiently large constant. Assumption 16 holds with probability at least $1-\frac{\delta}{2}$.*

We give a variant of Lemma 147 with slightly stronger guarantees for 1DRobustVariance; specifically, it holds for all u simultaneously for a fixed set of samples satisfying Assumption 16.

Corollary 71. *Under Assumption 16, Algorithm 93 outputs σ_u^2 with $|u^\top \mathbf{\Sigma} u - \sigma_u^2| < C u^\top \mathbf{\Sigma} u \cdot \epsilon \log \epsilon^{-1}$, for C a fixed multiple of the parameter c in Assumption 10, and runs in time $O(nd+n\log n)$.*

Proof. We discuss how to modify the derivations from Lemma 147 appropriately in the absence of applications of Bernstein's inequality. First, note that appropriately combining (G.43) and (G.44) in a derivation such as (G.30) yields the following bound (deterministically under Assumption 16):

$$C_4 \epsilon \log \epsilon^{-1} \cdot \mathbf{\Sigma} \succeq \sum_{i \in G'} w_i \left(X_i X_i^\top - \mathbf{\Sigma} \right) \succeq -C_4 \epsilon \log \epsilon^{-1} \mathbf{\Sigma} \text{ for all } w \in \mathfrak{S}_{3\epsilon}^n. \tag{G.45}$$

Now, consider the decomposition (G.38). We claim first that similarly to (G.40), (G.41), (G.42) we can bound each summand in the latter three terms by $O(u^\top \mathbf{\Sigma} u \log \epsilon^{-1})$; to prove this, it suffices to show that at least one filtered a_i attains this bound, as then by definition of the algorithm, each non-filtered a_i will as well. Note that a fraction between ϵ and 2ϵ of points in $G \subset G'$ is filtered (since there are only ϵn points from B). The assumption (G.44) then implies precisely the desired bound on some filtered a_i by placing uniform mass on filtered points from G, and applying pigeonhole. So, all non-filtered a_i are bounded by $O(u^\top \mathbf{\Sigma} u \log \epsilon^{-1})$, yielding analogous statements to (G.40), (G.41), (G.42).

Finally, an analogous derivation to (G.39) follows via an application of the bound (G.45), where we place uniform mass on the set $G' \cap I$ and adjust constants appropriately, since the above argument shows that under the assumption (G.44), we have that at most $2\epsilon n$ indices $i \in G'$ have $a_i \notin I$. □

G.8.2 Preliminaries

For convenience, we give the following preliminaries before embarking on our proof of Theorem 60 and giving guarantees on Algorithm 94. First, we state a set of assumptions which augments Assumption 16 with one additional condition, used in bounding the iteration count of our algorithm.

Assumption 17. *Under Assumption 10, let Assumption 16 hold, as well as the following additional condition for the same universal constant C_4:*

$$\|X_i\|_2^2 \le C_4 \log \frac{n}{\delta} \cdot \mathrm{Tr}(\mathbf{\Sigma}) \text{ for all } i \in G. \tag{G.46}$$

Standard sub-Gaussian concentration inequalities and a union bound, combined with our earlier claim Lemma 318, then yield the following guarantee.

Lemma 319. *Let $n = \Omega\left(\frac{d+\log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$ for a sufficiently large constant. Assumption 17 holds with probability at least $1-\delta$.*

G.8.3 Analysis of PCAFilter

For this section, for any nonnegative weights w, define $\mathbf{M}(w) := \sum_{i\in[n]} w_i X_i X_i^\top$. We now state our algorithm, PCAFilter. At all iterations t, it maintains a current nonnegative weight vector $w^{(t)}$ (initialized to be the uniform distribution on $[n]$), preserving the following invariants for all t:

$$w_i^{(t-1)} \geq w_i^{(t)} \text{ for all } i \in [n], \sum_{i\in B} w_i^{(t-1)} - w_i^{(t)} \geq \sum_{i\in G} w_i^{(t-1)} - w_i^{(t)}. \tag{G.47}$$

We now state our method as Algorithm 94; note that the update to $w^{(t)}$ is of the form in Lemma 148.

Algorithm 94: PCAFilter$(\{X_i\}_{i\in[n]}, \epsilon)$

1 Remove all points $i \in [n]$ with $\|X_i\|_2^2 > c\log(\frac{n}{\delta}) \cdot \mathrm{Tr}(\mathbf{\Sigma})$;
2 $w_i^{(0)} \leftarrow \frac{1}{n}$ for all $i \in [n]$, $t \leftarrow 1$;
3 $u_1 \leftarrow$ approximate top eigenvector of $\mathbf{M}(w^{(0)})$;
4 $\sigma_1^2 \leftarrow$ 1DRobustVariance$(\{X_i\}_{i\in[n]}, \epsilon, u_1)$;
5 **while** $u_t^\top \mathbf{M}(w^{(t-1)}) u_t > (1 + 5C_5\epsilon\log\epsilon^{-1})\sigma_t^2$, *where* $C_5 = \max(C, C_4)$ *from constants in Assumption 16, Corollary 71* **do**
6 $a_i \leftarrow \langle u_t, X_i\rangle^2$ for all $i \in [n]$;
7 Sort (permute) the indices $[n]$ so the a_i are in increasing order (with a_1 smallest, a_n largest);
8 Let ℓ be the largest index with $\sum_{i=\ell}^n w_i \geq 2\epsilon$;
9 Define

$$w_i^{(t)} \leftarrow \begin{cases} \left(1 - \frac{a_i}{a_n}\right) w_i^{(t-1)} & \ell \leq i \leq n \\ w_i^{(t-1)} & i < \ell \end{cases}$$

;
10 $u_t \leftarrow$ approximate top eigenvector of $\mathbf{M}(w^{(t)})$;
11 $\sigma_t^2 \leftarrow$ 1DRobustVariance$(\{X_i\}_{i\in[n]}, u_t, \epsilon)$;
12 $t \leftarrow t + 1$;
13 **Return:** u_t;

We assume that in Line 8, we also have $\sum_{i=\ell}^n w_i \leq 3\epsilon$, as we can assume at least one point is corrupted i.e. $\epsilon \geq \frac{1}{n}$ (else standard algorithms suffice for our setting), so adding an additional w_i can only change the sum by ϵ. We first prove invariants (G.47) are preserved; at a high level, we simply demonstrate that Lemma 148 holds via concentration on G and lack of termination.

Lemma 320. *Under Assumption 16, for any iteration t of Algorithm 94, suppose* (G.47) *held for all iterations $t' \leq t-1$. Then,* (G.47) *holds at iteration t.*

Proof. The first part of (G.47) is immediate by observing the update in Line 9, so we show the

second. We drop subscripts and superscripts for conciseness and focus on a single iteration t. Let $I_B = \{\ell, \dots, n\} \cap B$, and $I_G = \{\ell, \dots, n\} \cap G$. By Lemma 148, it suffices to demonstrate that

$$\sum_{i \in I_B} w_i a_i > \sum_{i \in I_G} w_i a_i. \tag{G.48}$$

First, $\sum_{i \in I_B} w_i \leq \epsilon$, so by definition of index ℓ, we have $\epsilon \leq \sum_{i \in I_G} w_i \leq 2\epsilon$. Define $\tilde{w}_i = \frac{w_i}{\sum_{i \in I_G} w_i}$ if $i \in I_G$, and 0 otherwise, and observe $\widetilde{w} \in \mathfrak{S}^n_{1-2\epsilon}$. By modifying constants appropriately from (G.44), it follows from definition of $a_i = u^\top X_i X_i^\top u$ that

$$\sum_{i \in I_G} w_i a_i \leq \left(\sum_{i \in I_G} w_i \right) \cdot C_4 \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u \leq 2 C_4 \epsilon \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u. \tag{G.49}$$

On the other hand, by (G.45) we know that the total quadratic form over G is bounded as

$$\sum_{i \in G} w_i a_i < \left(\sum_{i \in G} w_i \right) \left(1 + C_4 \epsilon \log \epsilon^{-1}\right) u^\top \mathbf{\Sigma} u < \left(1 + C_4 \epsilon \log \epsilon^{-1}\right) u^\top \mathbf{\Sigma} u. \tag{G.50}$$

Here, we applied the observation that the normalized w_i restricted to G are in $\mathfrak{S}^n_{1-3\epsilon}$ (e.g. using Lemma 321 inductively). However, since we did not terminate (Line 5), we must have by u_t being a top eigenvector and Corollary 71 (we defer discussions of inexactness to Theorem 60) that

$$\begin{aligned} \sum_{i \in [n]} w_i a_i \geq (1 + 5 C_5 \epsilon \log \epsilon^{-1}) \sigma_t^2 \geq (1 + 4 C_4 \epsilon \log \epsilon^{-1}) \cdot u^\top \mathbf{\Sigma} u \\ \implies \sum_{i \in B} w_i a_i > 3 C_4 \epsilon \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u. \end{aligned}$$

To obtain the last conclusion, we used (G.50). Finally, note that for all $i \in B \setminus I_B$,

$$a_i \leq a_\ell \leq \sum_{i \in I_G} \widetilde{w}_i a_i \leq C_4 \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u$$

by rearranging (G.49). This implies that

$$\sum_{i \in B \setminus I_B} w_i a_i \leq \left(\sum_{i \in B \setminus I_B} w_i \right) \cdot C_4 \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u \leq C_4 \epsilon \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u.$$

Thus, the desired inequality (G.48) follows from combining the above derivations, e.g. using (G.49) and

$$\sum_{i \in I_B} w_i a_i = \sum_{i \in B} w_i a_i - \sum_{i \in B \setminus I_B} w_i a_i > 2 C_4 \epsilon \log \epsilon^{-1} \cdot u^\top \mathbf{\Sigma} u.$$

□

Lemma 320 yields for all t that $\sum_{i\in B} w_i^{(0)} - w_i^{(t)} \geq \sum_{i\in G} w_i^{(0)} - w_i^{(t)}$ by telescoping. Note that we can only remove at most 2ϵ mass from w total, as $\sum_{i\in B} w_i^{(0)} - w_i^{(t)} \leq \epsilon$. Denote for shorthand normalized weights $v^{(t)} := \frac{w^{(t)}}{\left\|w^{(t)}\right\|_1}$. Then, the following is immediate by $\left\|w^{(t)}\right\|_1 \geq 1 - 2\epsilon$.

Lemma 321. *Under Assumption 16, in all iterations t of Algorithm 94, $v^{(t)} \in \mathfrak{S}_{2\epsilon}^n$.*

Using Lemma 321, we show that the output has the desired quality of being a large eigenvector.

Lemma 322. *Under Assumption 16, let the output of Algorithm 94 be u_T. Then for a universal constant $C^\star$, $u_T^\top \mathbf{\Sigma} u_T \geq (1 - C^\star \epsilon \log \epsilon^{-1}) \|\mathbf{\Sigma}\|_\infty$.*

Proof. We assume for now that u_T is an exact top eigenvector, and discuss inexactness while proving Theorem 60. By (G.45) and Lemma 321, as then the normalized restriction of $w^{(T)}$ to G is in $\mathfrak{S}_{3\epsilon}^n$,

$$\mathbf{M}(w^{(T)}) \succeq \sum_{i\in G} w_i^{(T)} X_i X_i^\top \succeq \left(1 - 2C_4 \epsilon \log \epsilon^{-1}\right) \mathbf{\Sigma}$$
$$\implies u_T^\top \mathbf{M}(w^{(T)}) u_T \geq \left(1 - 2C_4 \epsilon \log \epsilon^{-1}\right) \|\mathbf{\Sigma}\|_\infty .$$

We used the Courant-Fischer characterization of eigenvalues, and that u_T is a top eigenvector of $\mathbf{M}(w^{(T)})$. Moreover, by termination conditions and Corollary 71 (correctness of 1DRobustVariance),

$$(1 + C\epsilon \log \epsilon^{-1}) u_T^\top \mathbf{\Sigma} u_T \geq \sigma_T^2 \geq (1 + 5C_5 \epsilon \log \epsilon^{-1})^{-1} u_T^\top \mathbf{M}(w^{(T)}) u_T.$$

Combining these two bounds and rescaling yields the conclusion. □

Finally, we prove our main guarantee about Algorithm 94.

Theorem 60. *Under Assumption 10, let $\delta \in [0, 1]$, and $n = \Omega\left(\frac{d + \log \delta^{-1}}{(\epsilon \log \epsilon^{-1})^2}\right)$. Algorithm 94 runs in time $O(\frac{nd^2}{\epsilon} \log \frac{n}{\delta\epsilon} \log \frac{n}{\delta})$, and outputs u with $u^\top \mathbf{\Sigma} u > (1 - C^\star \epsilon \log \epsilon^{-1}) \|\mathbf{\Sigma}\|_\infty$, for $C^\star$ a fixed multiple of parameter c in Assumption 10, with probability at least $1 - \delta$.*

Proof. First, we will operate under Assumption 17, which holds with probability at least $1 - \delta$. It is clear that the analyses of Lemma 320 and 322 hold with $1 - \Theta(\epsilon \log \epsilon^{-1})$ multiplicative approximations of top eigenvector computation, which the power method approximates with high probability. Thus, each iteration takes time $O\left(\frac{nd}{\epsilon} \log \frac{n}{\delta\epsilon}\right)$, where we will union bound over the number of iterations.

We now give an iteration bound: in any iteration where we do not terminate, Lemma 148 implies

$$\begin{aligned}\sum_{i=1}^{n} w_i^{(t-1)} - w_i^{(t)} &\geq \frac{1}{2\max_{i\in[n]} \langle u_t, X_i\rangle^2} \sum_{i=\ell}^{n} w_i a_i \\ &\geq \frac{1}{2C_4 \log\frac{n}{\delta} \cdot \mathrm{Tr}(\mathbf{\Sigma})} \sum_{i=\ell}^{n} w_i a_i \\ &\geq \frac{1}{2C_4 \log\frac{n}{\delta} \cdot \mathrm{Tr}(\mathbf{\Sigma})} \left(\frac{\sum_{i=\ell}^{n} w_i}{\sum_{i\in[n]} w_i}\right) \sum_{i\in[n]} w_i a_i \\ &= \Omega\left(\epsilon \cdot \frac{\|\mathbf{\Sigma}\|_\infty}{\log\frac{n}{\delta} \cdot \mathrm{Tr}(\mathbf{\Sigma})}\right) = \Omega\left(\frac{\epsilon}{d\log\frac{n}{\delta}}\right).\end{aligned}$$

Here, the second line used Assumption 17, the third used that the a_i are in sorted order, and the last used the definition of ℓ as well as the derivations of Lemma 322 (specifically, that $\mathbf{M}(w)$ spectrally dominates $(1 - O(\epsilon \log \epsilon^{-1}))\mathbf{\Sigma}$ for roughly uniform w). The conclusion follows since there can be at most $O(d \log \frac{n}{\delta})$ iterations, as the algorithm terminates when a 2ϵ fraction of the mass is removed, giving the overall runtime claim. □

G.9 Deferred proofs from Section 8.11.2

G.9.1 Proof of Proposition 40

Proposition 40. *There is an algorithm* Power *(Algorithm 1, [404]), parameterized by $t \in [d]$, tolerance $\tilde{\epsilon} > 0$, $p \geq 1$, and $\mathbf{A} \in \mathbb{S}^d_{\geq 0}$, which outputs orthonormal $\{z_j\}_{j\in[t]}$ with the guarantee*

$$\begin{cases} \left|z_j^\top \mathbf{A}^p z_j - \lambda_j^p(\mathbf{A})\right| & \leq \tilde{\epsilon}\lambda_j^p(\mathbf{A}) \\ \left|z_j^\top \mathbf{A}^{p-1} z_j - \lambda_j^{p-1}(\mathbf{A})\right| & \leq \tilde{\epsilon}\lambda_j^{p-1}(\mathbf{A}) \end{cases} \text{ for all } j \in [t]. \tag{8.73}$$

Here, $\lambda_j(\mathbf{A})$ is the j^{th} largest eigenvalue of $\mathbf{A}$. The total time required by the method is $O(\mathrm{nnz}(\mathbf{A})\frac{tp\log d}{\varepsilon})$.

Proof. We claim that Algorithm 1 in [404] applied to the matrix $\mathbf{A}^p$ with a careful choice of exponent q in their Algorithm 1 yields this guarantee. Specifically, we choose q_1, q_2, both of which satisfy the criteria in their main theorem, such that the iterates produced by simultaneous power iteration $\mathbf{M}^p$ with exponent q_1 and $\mathbf{M}^{p-1}$ with exponent q_2 are identical; it suffices to choose q a multiple of $p(p-1)$. Thus, we can also apply their guarantees to $\mathbf{A}^{p-1}$ and apply a union bound. Notice that their Algorithm 1 also contains some postprocessing to ensure that they obtain singular values in the right space, which is unnecessary for us, as our matrices are Hermitian. □

G.9.2 Proof of Lemma 149

Lemma 149. *Let $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$. With probability $1-\frac{\delta}{2}$, the uniform distribution over G attains value $(1+\frac{\tilde{\epsilon}}{2})\|\mathbf{\Sigma}\|_p$ for objective (8.72), where $\tilde{\epsilon} = C'\epsilon\log\epsilon^{-1}$ for a universal constant $C' > 0$.*

Proof. Lemma 317 implies that letting w^* be the uniform distribution over the uncorrupted samples amongst $X_1, \ldots, X_n$, we have with probability at least $1-\frac{\delta}{2}$, and denoting $\tilde{\epsilon} := 2C_3 \cdot \epsilon\log\epsilon^{-1}$,

$$\left\|\sum_{i\in[n]} w_i^* X_i X_i^\top\right\|_p \le \left(1+\frac{\tilde{\epsilon}}{2}\right)\|\mathbf{\Sigma}\|_p.$$

Therefore, the mixed ℓ_∞-ℓ_p packing semidefinite program

$$\exists w^* \in \Delta^n \text{ with } \|w^*\|_\infty \le \frac{1}{(1-\epsilon)n}, \left\|\sum_{i\in[n]} w_i^* X_i X_i^\top\right\|_p \le \left(1+\frac{\tilde{\epsilon}}{2}\right)\|\mathbf{\Sigma}\|_p$$

is feasible. This completes the proof. □

G.9.3 Proof of Lemma 150

Lemma 150. *Let $n = \Omega\left(\frac{d+\log\delta^{-1}}{(\epsilon\log\epsilon^{-1})^2}\right)$. With probability at least $1-\frac{\delta}{2}$, $(1+\tilde{\epsilon})\mathbf{\Sigma} \succeq \mathbf{M}_G \succeq (1-\tilde{\epsilon})\mathbf{\Sigma}$.*

Proof. We follow the notation of (8.74). First, by the guarantees of Corollary 40,

$$w_G = 1 - w_B \ge 1 - \frac{\epsilon n}{(1-2\epsilon)n} = \frac{1-3\epsilon}{1-2\epsilon} \ge 1-2\epsilon.$$

Therefore, again applying Corollary 40, for all $i \in G$,

$$\frac{w_i}{w_G} \le \frac{1}{(1-2\epsilon)n}\cdot\frac{1-2\epsilon}{1-3\epsilon} = \frac{1}{(1-3\epsilon)n}.$$

We conclude that the set of weights $\{\frac{w_i}{w_G}\}_{i\in G}$ belong to $\mathfrak{S}_{3\epsilon}^{(1-\epsilon)n}$. By applying Corollary 70 to these weights and adjusting the definition of C_3 by a constant, we conclude with probability at least $1-\frac{\delta}{2}$

$$\left(1+C_3\cdot\epsilon\log\epsilon^{-1}\right)\mathbf{\Sigma} \succeq \sum_{i\in G}\frac{w_i}{w_G}X_iX_i^\top \succeq \left(1-C_3\cdot\epsilon\log\epsilon^{-1}\right)\mathbf{\Sigma}.$$

The conclusion follows by multiplying through by w_G, and using the definition $\tilde{\epsilon} = 2C_3\cdot\epsilon\log\epsilon^{-1}$. □

Appendix H

Deferred proofs from Chapter 9

H.1 Deferred proofs from Section 9.2

H.1.1 Polynomial approximation to the square root

We give a proof of Fact 23.

Fact 23 (Polynomial approximation of $\sqrt{\cdot}$)**.** *Let $\mathbf{M} \in \mathbb{S}^d_{\succ \mathbf{0}}$ have $\mu\mathbf{I} \preceq \mathbf{M} \preceq \kappa\mu\mathbf{I}$ where μ is known. Then for any $\delta \in (0,1)$, there is an explicit polynomial p of degree $O(\sqrt{\kappa}\log\frac{\kappa}{\delta})$ with*

$$(1-\delta)\mathbf{M}^{\frac{1}{2}} \preceq p(\mathbf{M}) \preceq (1+\delta)\mathbf{M}^{\frac{1}{2}}.$$

Proof. We will instead prove the following fact: for any $\epsilon \in (0,1)$, there is an explicit degree-$O\left(\sqrt{\kappa}\log\frac{\kappa}{\epsilon}\right)$ polynomial p satisfying

$$\max_{x\in[\frac{1}{\kappa},1]} |p(x) - \sqrt{x}| \leq \epsilon.$$

The conclusion for arbitrary scalars with multiplicative range $[\mu, \kappa\mu]$ will then follow from setting $\epsilon = \delta\kappa^{-\frac{1}{2}}$ (giving a multiplicative error guarantee), and the fact that rescaling the range $[\frac{1}{\kappa}, 1]$ will preserve this multiplicative guarantee (adjusting the coefficients of the polynomial as necessary, since μ is known). Finally, the conclusion for matrices follows since $p(\mathbf{M})$ and $\mathbf{M}^{\frac{1}{2}}$ commute.

Denote $\gamma = \frac{1}{\kappa}$ for convenience. We first shift and scale the function $\sqrt{x}$ to adjust the region of approximation from $[\gamma, 1]$ to $[-1, 1]$. In particular, let $h(x) = \sqrt{\frac{1-\gamma}{2}x + \frac{\gamma+1}{2}}$. If we can find some degree-Δ polynomial $g(x)$ with $|g(x) - h(x)| \leq \epsilon$ for all $x \in [-1, 1]$, then

$$p(x) = g\left(\frac{2}{1-\gamma}x \ - \ \frac{1+\gamma}{1-\gamma}\right)$$

provides the required approximation to $\sqrt{x}$.

To construct g, we take the Chebyshev interpolant of $h(x)$ on the interval $[-1,1]$. Since h is analytic on $[-1,1]$, we can apply standard results on the approximation of analytic functions by polynomials, and specifically Chebyshev interpolants. Specifically, by Theorem 8.2 in [509], if $h(z)$ is analytic in an open Bernstein ellipse with parameter ρ in the complex plane, then:

$$\max_{x\in[-1,1]} |g(x) - h(x)| \leq \frac{4M}{\rho - 1}\rho^{-\Delta},$$

where M is the maximum of $|h(z)|$ for z in the ellipse. It can be checked that $h(x)$ is analytic on an open Bernstein ellipse with parameter $\rho = \frac{1+\sqrt{\gamma}}{1-\sqrt{\gamma}}$ — i.e. with major axis length $\rho + \rho^{-1} = 2\frac{1+\gamma}{1-\gamma}$. We can then check that $M = \sqrt{1+\gamma} \leq \sqrt{2}$ and $\rho - 1 \geq 2\sqrt{\gamma}$. Since for all $\gamma < 1$,

$$\left(\frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}\right)^{1/2\gamma} \leq \frac{1}{e},$$

we conclude that $\frac{4M}{\rho-1}\rho^{-\Delta} \leq \epsilon$ as long as $\Delta \geq \frac{1}{2\gamma}\log\left(\frac{\epsilon}{\sqrt{2}\gamma}\right)$, which completes the proof. □

H.1.2 Deferred proofs from Section 9.2.2

Lemma 153. *Let* $\mathbf{B} \in \mathbb{R}^{d\times d}$ *and let* $\mathbf{A} \in \mathbb{S}^d_{\succ \mathbf{0}}$. *Then* $\min\left(\|\mathbf{AB}\|_2, \|\mathbf{BA}\|_2\right) \geq \frac{1}{\kappa(\mathbf{A})}\|\mathbf{B}\|_2\|\mathbf{A}\|_2$.

Proof. We begin with the first entry in the above minimum. Let v be the unit vector with $\|\mathbf{B}v\|_2 = \|\mathbf{B}\|_\infty$, and note $\|\mathbf{ABB}v\|_2 \geq \frac{1}{\kappa(\mathbf{A})}\|\mathbf{A}\|_\infty\|\mathbf{B}v\|_2$ by definition of $\kappa(\mathbf{A})$. Hence,

$$\|\mathbf{AB}\|_\infty \geq \|\mathbf{ABB}v\|_2 \geq \frac{1}{\kappa(\mathbf{A})}\|\mathbf{A}\|_\infty\|\mathbf{B}v\|_2 = \frac{1}{\kappa(\mathbf{A})}\|\mathbf{A}\|_\infty\|\mathbf{B}\|_\infty.$$

We move onto the second entry. Let v be a vector such that $\|\mathbf{A}v\|_2$ and $\|\mathbf{BA}v\|_2 = \|\mathbf{B}\|_\infty$; note that $\|v\|_2 \leq \frac{\kappa(\mathbf{A})}{\|\mathbf{A}\|_\infty}$. The conclusion follows from rearranging the following display:

$$\frac{\kappa(\mathbf{A})\|\mathbf{BA}\|_\infty}{\|\mathbf{A}\|_\infty} \geq \|\mathbf{BA}\|_\infty\|v\|_2 \geq \|\mathbf{BA}v\|_2 = \|\mathbf{B}\|_\infty.$$

□

Lemma 161. *For any matrix* $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ *and* $\lambda \geq 0$, $\kappa^\star_o(\mathbf{K} + \lambda\mathbf{I}) \leq \kappa^\star_o(\mathbf{K})$.

Proof. By scaling $\mathbf{K}$ by λ appropriately (since $\kappa^\star_o$ is invariant under scalar multiplication), it suffices to take $\lambda = 1$. The definition of $\kappa^\star_o$ implies there exists a diagonal matrix $\mathbf{W}$ such that

$$\mathbf{I} \preceq \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}} \preceq \kappa^\star_o(\mathbf{K})\mathbf{I} \iff \mathbf{W}^{-1} \preceq \mathbf{K} \preceq \kappa^\star_o(\mathbf{K})\mathbf{W}^{-1}. \tag{H.1}$$

Thus, to demonstrate $\kappa_o^\star(\mathbf{K}+\mathbf{I}) \le \kappa_o^\star(\mathbf{K})$ it suffices to exhibit a diagonal $\widetilde{\mathbf{W}}$ such that

$$\widetilde{\mathbf{W}} \preceq \mathbf{K}+\mathbf{I} \preceq \kappa_o^\star(\mathbf{K})\widetilde{\mathbf{W}}.$$

We choose $\widetilde{\mathbf{W}} = \mathbf{W}^{-1}+\mathbf{I}$; then, the above display follows from (H.1) and $\mathbf{I} \preceq \mathbf{I} \preceq \kappa_o^\star(\mathbf{K})\mathbf{I}$. □

Lemma 162. *Let $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$. Then, for $\lambda \ge \frac{1}{\epsilon}\lambda_{\max}(\mathbf{K})$, $\kappa(\mathbf{K}+\lambda\mathbf{I}) \le 1+\epsilon$. Moreover, given a diagonal $\mathbf{W} \in \mathbb{S}^d_{\succeq 0}$ such that $\kappa(\mathbf{W}^{\frac12}(\mathbf{K}+\lambda\mathbf{I})\mathbf{W}^{\frac12}) \le \kappa_{\text{scale}}$ for $0 \le \lambda \le \frac{\epsilon\lambda_{\min}(\mathbf{K})}{1+\epsilon}$, $\kappa(\mathbf{W}^{\frac12}\mathbf{K}\mathbf{W}^{\frac12}) \le (1+\epsilon)\kappa_{\text{scale}}$.*

Proof. To see the first claim, the largest eigenvalue of $\mathbf{K}+\lambda\mathbf{I}$ is at most $\lambda+\lambda_{\max}(\mathbf{K})$ and the smallest is at least λ, so the condition number is at most $1+\epsilon$ as desired.

To see the second claim, it follows from the fact that outer rescalings preserve Loewner order, and then combining

$$\begin{aligned}\mathbf{K} \preceq \mathbf{K}+\lambda\mathbf{I} &\implies \lambda_{\max}\left(\mathbf{W}^{\frac12}\mathbf{K}\mathbf{W}^{\frac12}\right) \le \lambda_{\max}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\lambda\mathbf{I}\right)\mathbf{W}^{\frac12}\right),\\ \mathbf{K} \succeq \frac{1}{1+\epsilon}\left(\mathbf{K}+\lambda\mathbf{I}\right) &\implies \lambda_{\min}\left(\mathbf{W}^{\frac12}\mathbf{K}\mathbf{W}^{\frac12}\right) \ge \frac{1}{1+\epsilon}\lambda_{\min}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\lambda\mathbf{I}\right)\mathbf{W}^{\frac12}\right).\end{aligned}$$

□

Lemma 163. *Let $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$, and let $\mathbf{W} \in \mathbb{S}^d_{\succeq 0}$ be diagonal. Then for any $\lambda > 0$,*

$$\kappa\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\lambda\mathbf{I}\right)\mathbf{W}^{\frac12}\right) \le 2\kappa\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\frac{\lambda}{2}\mathbf{I}\right)\mathbf{W}^{\frac12}\right).$$

Proof. First, because outer rescalings preserve Loewner order, it is immediate that

$$\mathbf{K}+\frac{\lambda}{2}\mathbf{I} \preceq \mathbf{K}+\lambda\mathbf{I} \implies \lambda_{\max}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\frac{\lambda}{2}\right)\mathbf{W}^{\frac12}\mathbf{I}\right) \le \lambda_{\max}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\lambda\mathbf{I}\right)\mathbf{W}^{\frac12}\right).$$

Moreover, the same argument shows that

$$\frac12\mathbf{K}+\frac{\lambda}{2}\mathbf{I} \preceq \mathbf{K}+\frac{\lambda}{2}\mathbf{I} \implies \lambda_{\min}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\frac{\lambda}{2}\mathbf{I}\right)\mathbf{W}^{\frac12}\right) \ge \frac12\lambda_{\min}\left(\mathbf{W}^{\frac12}\left(\mathbf{K}+\mathbf{I}\right)\mathbf{W}^{\frac12}\right).$$

Combining the above two displays yields the conclusion. □

H.1.3 Normalizing the diagonal

In this section, we analyze a popular heuristic for computing diagonal preconditioners. Given a positive definite matrix $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$, consider applying the outer scaling

$$\mathbf{W}^{\frac12}\mathbf{K}\mathbf{W}^{\frac12}, \text{ where } \mathbf{W} = \mathbf{diag}\,(w) \text{ and } w_i := \mathbf{K}_{ii}^{-1} \text{ for all } i \in [d]. \tag{H.2}$$

In other words, the result of this scaling is to simply normalize the diagonal of $\mathbf{K}$ to be all ones; we remark $\mathbf{W}$ has strictly positive diagonal entries, else $\mathbf{K}$ is not positive definite. Also called the Jacobi preconditioner, a result of Van de Sluis [251, 518] proves that *for any matrix* this scaling leads to a condition number that is within an m factor of optimal, where $m \leq d$ is the maximum number of non-zeros in any row of $\mathbf{K}$. For completeness, we state a generalization of Van de Sluis's result below. We also require a simple fact; both are proven at the end of this section.

Fact 44. *For any* $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d_{\succ \mathbf{0}}$, $\kappa(\mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}) \leq \kappa(\mathbf{A})\kappa(\mathbf{B})$.

Proposition 74. *Let* $\mathbf{W}$ *be defined as in* (H.2) *and let* m *denote the maximum number of non-zero's in any row of* $\mathbf{K}$*. Then,*

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) \leq \min\left(m, \sqrt{\mathrm{nnz}(\mathbf{K})}\right) \cdot \kappa_o^\star(\mathbf{K}).$$

Note that m and $\sqrt{\mathrm{nnz}(\mathbf{K})}$ are both $\leq d$, so it follows that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) \leq d \cdot \kappa_o^\star(\mathbf{K})$. While the approximation factor in Proposition 74 depends on the dimension or sparsity of $\mathbf{K}$, we show that a similar analysis actually yields a *dimension-independent* approximation. Specifically, the Jacobi preconditioner always obtains condition number no worse than the optimal squared. To the best of our knowledge, this simple but powerful bound has been observed in prior work.

Proposition 75. *Let* $\mathbf{W}$ *be defined as in* (H.2)*. Then,*

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) \leq \left(\kappa_o^\star(\mathbf{K})\right)^2.$$

Proof. Let $\mathbf{W}_\star$ attain the minimum in the definition of $\kappa_o^\star$, i.e. $\kappa(\mathbf{K}_\star) = \kappa_o^\star(\mathbf{K})$ for $\mathbf{K}_\star := \mathbf{W}_\star^{\frac{1}{2}}\mathbf{K}\mathbf{W}_\star^{\frac{1}{2}}$. Note that since $[\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}]_{ii} = 1$ by definition of $\mathbf{W}$ it follows that for all i

$$[\mathbf{W}_\star\mathbf{W}^{-1}]_{ii} = [\mathbf{W}_\star\mathbf{W}^{-1}]_{ii} \cdot [\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}]_{ii} = [\mathbf{K}_\star]_{ii} = e_i^\top \mathbf{K}_\star e_i \in [\lambda_{\min}(\mathbf{K}_\star), \lambda_{\max}(\mathbf{K}_\star)]$$

where the last step used that $\lambda_{\min}(\mathbf{K}_\star)\mathbf{I} \preceq \mathbf{K}_\star \preceq \lambda_{\max}(\mathbf{K}_\star)\mathbf{I}$. Consequently, for $\widetilde{\mathbf{W}} := \mathbf{W}_\star^{-1}\mathbf{W}$ it follows that $\kappa(\widetilde{\mathbf{W}}) = \kappa(\widetilde{\mathbf{W}}^{-1}) \leq \lambda_{\max}(\mathbf{K}_\star)/\lambda_{\min}(\mathbf{K}_\star) = \kappa(\mathbf{K}_\star)$. The result follows from Fact 44 as

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) = \kappa\left(\widetilde{\mathbf{W}}^{\frac{1}{2}}\mathbf{K}_\star\widetilde{\mathbf{W}}^{\frac{1}{2}}\right) \leq \kappa\left(\widetilde{\mathbf{W}}\right)\kappa(\mathbf{K}_\star) \leq \left(\kappa_o^\star\right)^2.$$

□

Next, we demonstrate that Proposition 75 is essentially tight by exhibiting a family of matrices which attain the bound of Proposition 75 up to a constant factor. At a high level, our strategy is to create two blocks where the "scales" of the diagonal normalizing rescaling are at odds, whereas a simple rescaling of one of the blocks would result in a quadratic savings in conditioning.

Proposition 76. *Consider a $2d \times 2d$ matrix $\mathbf{M}$ such that*

$$\mathbf{K} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}, \ \mathbf{A} = \sqrt{d}\mathbf{I} + \mathbf{1}\mathbf{1}^\top, \ \mathbf{B} = \mathbf{I} - \frac{1}{\sqrt{d}+d}\mathbf{1}\mathbf{1}^\top,$$

where $\mathbf{A}$ and $\mathbf{B}$ are $d \times d$. Then, defining $\mathbf{W}$ as in (H.2)*,*

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) = \Theta(d), \ \kappa_o^\star(\mathbf{K}) = \Theta\left(\sqrt{d}\right).$$

Proof. Because $\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}$ is blockwise separable, to understand its eigenvalue distribution it suffices to understand the eigenvalues of the two blocks. First, the upper-left block (the rescaling of the matrix $\mathbf{A}$) is multiplied by $\frac{1}{\sqrt{d}+1}$. It is straightforward to see that the resulting eigenvalues are

$$\frac{\sqrt{d}}{\sqrt{d}+1} \text{ with multiplicity } d-1, \ \sqrt{d} \text{ with multiplicity } 1.$$

Similarly, the bottom-right block is multiplied by $\frac{d+\sqrt{d}}{d+\sqrt{d}-1}$, and hence its rescaled eigenvalues are

$$\frac{d+\sqrt{d}}{d+\sqrt{d}-1} \text{ with multiplicity } d-1, \ \frac{\sqrt{d}}{d+\sqrt{d}-1} \text{ with multiplicity } 1.$$

Hence, the condition number of $\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}$ is $d+\sqrt{d}-1 = \Theta(d)$. However, had we rescaled the top-left block to be a $\sqrt{d}$ factor smaller, it is straightforward to see the resulting condition number is $O(\sqrt{d})$. On the other hand, since the condition number of $\mathbf{K}$ is $O(d)$, Proposition 75 shows that the optimal condition number $\kappa_o^\star(\mathbf{K})$ is $\Omega(\sqrt{d})$, and combining yields the claim. We remark that as $d \to \infty$, the constants in the upper and lower bounds agree up to a low-order term. □

Finally, we provide the requisite proofs of Fact 44 and Proposition 74.

Fact 44. *For any $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d_{\succ \mathbf{0}}$, $\kappa(\mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}) \le \kappa(\mathbf{A})\kappa(\mathbf{B})$.*

Proof. It is straightforward from $\lambda_{\min}(\mathbf{A})\mathbf{I} \preceq \mathbf{A} \preceq \lambda_{\max}(\mathbf{A})\mathbf{I}$ that

$$\sqrt{\lambda_{\min}(\mathbf{A})}\,\|u\|_2 \le \left\|\mathbf{A}^{\frac{1}{2}}u\right\|_2 \le \sqrt{\lambda_{\max}(\mathbf{A})}\,\|u\|_2,$$

and an analogous fact holds for $\mathbf{B}$. Hence, we can bound the eigenvalues of $\mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}$:

$$\lambda_{\max}\left(\mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}\right) = \max_{\|u\|_2=1} u^\top \mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}\mathbf{B}u \le \lambda_{\max}(\mathbf{A}) \max_{\|v\|_2=1} v^\top \mathbf{B}v = \lambda_{\max}(\mathbf{A})\lambda_{\max}(\mathbf{B}),$$
$$\lambda_{\min}\left(\mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}\right) = \min_{\|u\|_2=1} u^\top \mathbf{A}^{\frac{1}{2}}\mathbf{B}\mathbf{A}^{\frac{1}{2}}\mathbf{B}u \ge \lambda_{\min}(\mathbf{A}) \min_{\|v\|_2=1} v^\top \mathbf{B}v = \lambda_{\min}(\mathbf{A})\lambda_{\min}(\mathbf{B}).$$

Dividing the above two equations yields the claim. □

Proposition 74. *Let* $\mathbf{W}$ *be defined as in* (H.2) *and let* m *denote the maximum number of non-zero's in any row of* $\mathbf{K}$*. Then,*

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) \leq \min\left(m, \sqrt{\mathrm{nnz}(\mathbf{K})}\right) \cdot \kappa_o^\star(\mathbf{K}).$$

Proof. Throughout let $\kappa_o^\star := \kappa_o^\star(\mathbf{K})$ for notational convenience. Let $\mathbf{W}_\star$ obtain the minimum in the definition of $\kappa_o^\star$ and let $\mathbf{B} = \mathbf{W}_\star^{\frac{1}{2}}\mathbf{K}\mathbf{W}_\star^{\frac{1}{2}}$. Also let $\mathbf{W}_\mathbf{B}$ be the inverse of a diagonal matrix with the same entries as $\mathbf{B}$'s diagonal. Note that $\kappa(\mathbf{B}) = \kappa_o^\star$ and $\mathbf{W}_\mathbf{B}^{\frac{1}{2}}\mathbf{B}\mathbf{W}_\mathbf{B}^{\frac{1}{2}} = \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}$. So, to prove Proposition 74, it suffices to prove that

$$\kappa\left(\mathbf{W}_\mathbf{B}^{\frac{1}{2}}\mathbf{B}\mathbf{W}_\mathbf{B}^{\frac{1}{2}}\right) \leq \min\left(m, \sqrt{\mathrm{nnz}(\mathbf{K})}\right) \cdot \kappa_o^\star.$$

Let $d_{\max}$ denote the largest entry in $\mathbf{W}_\mathbf{B}^{-1}$. We have that $d_{\max} \leq \lambda_{\max}(\mathbf{B})$. Then let $\mathbf{M} = (d_{\max}\mathbf{W}_\mathbf{B})^{\frac{1}{2}}\mathbf{B}(d_{\max}\mathbf{W}_\mathbf{B})^{\frac{1}{2}}$ and note that all of $\mathbf{M}$'s diagonal entries are equal to $d_{\max}$ and $\kappa(\mathbf{M}) = \kappa(\mathbf{W}_\mathbf{B}^{\frac{1}{2}}\mathbf{B}\mathbf{W}_\mathbf{B}^{\frac{1}{2}})$. Moreover, since $d_{\max}\mathbf{W}_\mathbf{B}$ has all entries ≥ 1, $\lambda_{\min}(\mathbf{M}) \geq \lambda_{\min}(\mathbf{B})$. Additionally, since a PSD matrix must have its largest entry on the diagonal, we have that $\|\mathbf{M}\|_{\mathrm{F}}^2 \leq \mathrm{nnz}(\mathbf{M})d_{\max}^2 \leq \mathrm{nnz}(\mathbf{M})\lambda_{\max}(\mathbf{B})^2$. Accordingly, $\lambda_{\max}(\mathbf{M}) = \|\mathbf{M}\|_2 \leq \|\mathbf{M}\|_{\mathrm{F}} \leq \sqrt{\mathrm{nnz}(\mathbf{M})}\lambda_{\max}(\mathbf{B})$.

From this lower bound on $\lambda_{\min}(\mathbf{M})$ and upper bound on $\lambda_{\max}(\mathbf{M})$, we have that

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) = \kappa(\mathbf{M}) \leq \frac{\sqrt{\mathrm{nnz}(\mathbf{M})}\lambda_{\max}(\mathbf{B})}{\lambda_{\min}(\mathbf{B})} = \sqrt{\mathrm{nnz}(\mathbf{M})} \cdot \kappa(\mathbf{B}).$$

This proves one part of the minimum in Proposition 74. The second, which was already proven in [518] follows similarly. In particular, by the Gershgorin circle theorem we have that $\lambda_{\max}(\mathbf{M}) \leq \max_{i\in[d]} \|\mathbf{M}_{i:}\|_1$, where $\mathbf{M}_{i:}$ denotes the i^{th} row for $\mathbf{M}$. Since all entries in $\mathbf{M}$ are bounded by $d_{\max} \leq \lambda_{\max}(\mathbf{B})$, we have that $\max_{i\in[d]} \|\mathbf{M}_{i:}\|_1 \leq m\lambda_{\max}(\mathbf{B})$, and thus

$$\kappa\left(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}\right) = \kappa(\mathbf{M}) \leq \frac{m\lambda_{\max}(\mathbf{B})}{\lambda_{\min}(\mathbf{B})} = m \cdot \kappa(\mathbf{B}).$$

□

H.1.4 Faster scalings with a conjectured subroutine

In this section, we demonstrate algorithms which achieve runtimes which scale as $\widetilde{O}(\sqrt{\kappa^\star})$[1] matrix-vector multiplies for computing approximately optimal scalings, assuming the existence of a sufficiently general *width-independent* mixed packing and covering (MPC) SDP solver. Such runtimes (which improve each of Theorems 62 and 63 by roughly a $\kappa^\star$ factor) would nearly match the cost of the fastest solvers *after* rescaling, e.g. conjugate gradient methods. We also demonstrate that we

[1] Throughout this section for brevity, we use $\kappa^\star$ to interchangeably refer to the quantities $\kappa_i^\star$ or $\kappa_o^\star$ of a particular appropriate inner or outer rescaling problem.

can achieve near-optimal algorithms for computing constant-factor optimal scalings for average-case notions of conditioning under this assumption.

We first recall the definition of the general MPC SDP feasibility problem.

Definition 59 (MPC feasibility problem)**.** *Given sets of matrices* $\{\mathbf{P}_i\}_{i\in[n]} \in \mathbb{S}^{d_p}_{\geq 0}$ *and* $\{\mathbf{C}_i\}_{i\in[n]} \in \mathbb{S}^{d_c}_{\geq 0}$*, and error tolerance* $\epsilon \in (0,1)$*, the mixed packing-covering (MPC) feasibility problem asks to return weights* $w \in \mathbb{R}^n_{\geq 0}$ *such that*

$$\lambda_{\max}\left(\sum_{i\in[n]} w_i\mathbf{P}_i\right) \leq (1+\epsilon)\lambda_{\min}\left(\sum_{i\in[n]} w_i\mathbf{C}_i\right), \tag{H.3}$$

or conclude that the following is infeasible for $w \in \mathbb{R}^n_{\geq 0}$*:*

$$\lambda_{\max}\left(\sum_{i\in[n]} w_i\mathbf{P}_i\right) \leq \lambda_{\min}\left(\sum_{i\in[n]} w_i\mathbf{C}_i\right). \tag{H.4}$$

If both (H.3) *is feasible and* (H.4) *is infeasible, either answer is acceptable.*

Throughout this section, we provide efficient algorithms under Assumption 18: namely, that there exists a solver for the MPC feasibility problem at constant ϵ with polylogarithmic iteration complexity and sufficient approximation tolerance. Such a solver would improve upon our algorithm in Section 7.4 both in generality (i.e. without the restriction that the constraint matrices are multiples of each other) and in the number of iterations.

Assumption 18. *There is an algorithm* MPC *which takes inputs* $\{\mathbf{P}_i\}_{i\in[n]} \in \mathbb{S}^{d_p}_{\geq 0}$*,* $\{\mathbf{C}_i\}_{i\in[n]} \in \mathbb{S}^{d_c}_{\geq 0}$*, and error tolerance* ϵ*, and solves problem* (H.3)*,* (H.4)*, in* $\mathrm{poly}(\log(nd\rho), \epsilon^{-1})$ *iterations, where* $d := \max(d_p, d_c)$*,* $\rho := \max_{i\in[n]} \frac{\lambda_{\max}(\mathbf{C}_i)}{\lambda_{\max}(\mathbf{P}_i)}$*. Each iteration uses* $O(1)$ *n-dimensional vector operations, and for* $\epsilon' = \Theta(\epsilon)$ *with an appropriate constant, additionally requires computation of*

$$\begin{aligned} &\epsilon'\text{-multiplicative approximations to } \left\langle \mathbf{P}_i, \frac{\exp\left(\sum_{i\in[n]} w_i\mathbf{P}_i\right)}{\mathrm{Tr}\exp\left(\sum_{i\in[n]} w_i\mathbf{P}_i\right)} \right\rangle \ \forall i \in [n], \\ &\left(\epsilon', e^{\frac{-\log(nd\rho)}{\epsilon'}}\mathrm{Tr}(\mathbf{C}_i)\right)\text{-approximations to } \left\langle \mathbf{C}_i, \frac{\exp\left(-\sum_{i\in[n]} w_i\mathbf{C}_i\right)}{\mathrm{Tr}\exp\left(-\sum_{i\in[n]} w_i\mathbf{C}_i\right)} \right\rangle \ \forall i \in [n], \end{aligned} \tag{H.5}$$

for $w \in \mathbb{R}^n_{\geq 0}$ *with* $\lambda_{\max}\left(\sum_{i\in[n]} w_i\mathbf{P}_i\right), \lambda_{\min}\left(\sum_{i\in[n]} w_i\mathbf{C}_i\right) \leq R$ *for* $R = O(\frac{\log(nd\rho)}{\epsilon})$*.*

In particular, we observe that the number of iterations of this conjectured subroutine depends polylogarithmically on ρ, i.e. the runtime is *width-independent*.[2] In our settings computing optimal

[2]The literature on approximate solvers for positive linear programs and semidefinite programs refer to logarithmic dependences on ρ as width-independent, and we follow this convention in our exposition.

rescaled condition numbers, $\rho = \Theta(\kappa^\star)$; our solver in Section 7.4 has an iteration count depending linearly on ρ. Such runtimes are known for MPC linear programs [379], however, such rates have been elusive in the SDP setting. While the form of requirements in (H.5) may seem somewhat unnatural at first glance, we observe that this is the natural generalization of the error tolerance of known width-independent MPC LP solvers [379]. Moreover, these approximations mirror the tolerances of our width-dependent solver in Section 7.4 (see Line 6 and Corollary 43).

We first record the following technical lemma, which we will repeatedly use.

Lemma 323. *Given a matrix $\mathbf{0} \preceq \mathbf{M} \preceq R\mathbf{I}$ for some $R > 0$, sufficiently small constant ϵ, and $\delta \in (0,1)$, we can compute ϵ-multiplicative approximations to the quantities*

$$\left\langle a_i a_i^\top, \exp(\mathbf{M}) \right\rangle \text{ for all } i \in [n], \text{ and } \operatorname{Tr}\exp(\mathbf{M})$$

in time $O((\mathcal{T}_{\mathrm{mv}}(\mathbf{M})R + \mathrm{nnz}(\mathbf{A}))\log\frac{n}{\delta})$, with probability at least $1-\delta$.

Proof. We discuss both parts separately. Regarding computing the inner products, equivalently, the goal is to compute approximations to all $\left\|\exp(\frac{1}{2}\mathbf{M})a_i\right\|_2^2$ for $i \in [n]$. First, by an application of Fact 12 with $\delta = \frac{\epsilon}{8}\exp(-2R)$, and then multiplying all sides of the inequality by $\exp(R)$, there is a degree-$O(R)$ polynomial such that

$$\left(1-\frac{\epsilon}{8}\right)\exp\left(\frac{1}{2}\mathbf{M}\right) \preceq \exp\left(\frac{1}{2}\mathbf{M}\right) - \frac{\epsilon}{8}\mathbf{I} \preceq p\left(\frac{1}{2}\mathbf{M}\right) \preceq \exp\left(\frac{1}{2}\mathbf{M}\right) + \frac{\epsilon}{8}\mathbf{I} \preceq \left(1+\frac{\epsilon}{8}\right)\exp\left(\frac{1}{2}\mathbf{M}\right)$$
$$\implies \left(1-\frac{\epsilon}{3}\right)\exp(\mathbf{M}) \preceq p\left(\frac{1}{2}\mathbf{M}\right)^2 \preceq \left(1+\frac{\epsilon}{3}\right)\exp(\mathbf{M}).$$

This implies that $\left\|p(\frac{1}{2}\mathbf{M})a_i\right\|_2^2$ approximates $\left\|\exp(\frac{1}{2}\mathbf{M})a_i\right\|_2^2$ to a multiplicative $\frac{\epsilon}{3}$ by the definition of Loewner order. Moreover, applying Fact 11 with a sufficiently large $k = O(\log\frac{n}{\delta})$ implies by a union bound that for all $i \in [n]$, $\left\|\mathbf{Q}p(\frac{1}{2}\mathbf{M})a_i\right\|_2^2$ is a ϵ-multiplicative approximation to $\left\|\exp(\frac{1}{2}\mathbf{M})a_i\right\|_2^2$. To compute all the vectors $\mathbf{Q}p(\frac{1}{2}\mathbf{M})a_i$, it suffices to first apply $p(\frac{1}{2}\mathbf{M})$ to all rows of $\mathbf{Q}$, which takes time $O(\mathcal{T}_{\mathrm{mv}}(\mathbf{M})\cdot kR)$ since p is a degree-$O(R)$ polynomial. Next, once we have the explicit $k \times d$ matrix $\mathbf{Q}p(\frac{1}{2}\mathbf{M})$, we can apply it to all $\{a_i\}_{i\in[n]}$ in time $O(\mathrm{nnz}(\mathbf{A})\cdot k)$.

Next, consider computing $\operatorname{Tr}\exp(\mathbf{M})$, which by definition has

$$\operatorname{Tr}\exp(\mathbf{M}) = \sum_{j\in[d]}\left\|\left[\exp\left(\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2.$$

Applying the same $\mathbf{Q}$ and p as before, we have by the following sequence of equalities

$$\sum_{j\in[d]}\left\|\mathbf{Q}\left[\exp\left(\frac{1}{2}\mathbf{M}\right)\right]_{j:}\right\|_2^2 = \operatorname{Tr}\left(\exp\left(\frac{1}{2}\mathbf{M}\right)\mathbf{Q}^\top\mathbf{Q}\exp\left(\frac{1}{2}\mathbf{M}\right)\right)$$
$$= \operatorname{Tr}\left(\mathbf{Q}\exp\left(\mathbf{M}\right)\mathbf{Q}^\top\right) = \sum_{\ell\in[k]}\left\|\exp\left(\frac{1}{2}\mathbf{M}\right)\mathbf{Q}_{\ell:}\right\|_2^2,$$

that for the desired approximation, it instead suffices to compute

$$\sum_{\ell\in[k]}\left\|p\left(\frac{1}{2}\mathbf{M}\right)\mathbf{Q}_{\ell:}\right\|_2^2.$$

This can be performed in time $O(\mathcal{T}_{\mathrm{mv}}(\mathbf{M})\cdot kR)$ as previously argued. □

A straightforward modification of this proof alongside Lemma 154 also implies that we can compute these same quantities to $p(\mathbf{AWA})$, when we are only given $\mathbf{K}=\mathbf{A}^2$, assuming that $\mathbf{K}$ is reasonably well-conditioned. We omit the proof, as it follows almost identically to the proofs of Lemmas 323, 157, and 158, the latter two demonstrating how to appropriately apply Lemma 154.

Corollary 72. *Let $\mathbf{K}\in\mathbb{S}^d_{\succ\mathbf{0}}$ such that $\mathbf{K}=\mathbf{A}^2$ and $\kappa(\mathbf{K})\le\kappa_{\mathrm{scale}}$. Let $\mathbf{W}$ be a diagonal matrix such that $\lambda_{\max}(\mathbf{AWA})\le R$. For $\delta,\epsilon\in(0,1)$, we can compute ϵ-multiplicative approximations to*

$$\left\langle a_i a_i^\top, \exp(\mathbf{AWA})\right\rangle \text{ for all } i\in[n], \text{ and } \operatorname{Tr}\exp(\mathbf{AWA})$$

with probability $\ge 1-\delta$ in time $O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K})\cdot R\cdot\left(R+\sqrt{\kappa_{\mathrm{scale}}}\log\frac{d\kappa_{\mathrm{scale}}}{\delta}\right)\log\frac{n}{\delta}\right)$.

Approximating $\kappa^\star$ under Assumption 18

We show that, given Assumption 18, we obtain improved runtimes for all three types of diagonal scaling problems, roughly improving Theorems 62 and 63 by a $\kappa^\star$ factor.

Inner scalings. We first demonstrate this improvement for inner scalings.

Theorem 92. *Under Assumption 18, there is an algorithm which, given full-rank $\mathbf{A}\in\mathbb{R}^{n\times d}$ for $n\ge d$ computes $w\in\mathbb{R}^n_{\ge 0}$ such that $\kappa(\mathbf{A}^\top\mathbf{W}\mathbf{A})\le(1+\epsilon)\kappa_i^\star(\mathbf{A})$ for arbitrarily small $\epsilon=\Theta(1)$, with probability $\ge 1-\delta$ in time*

$$O\left(\operatorname{nnz}(\mathbf{A})\cdot\sqrt{\kappa_i^\star(\mathbf{A})}\cdot\operatorname{poly}\log\frac{n\kappa_i^\star(\mathbf{A})}{\delta}\right).$$

Proof. For now, assume we know $\kappa_i^\star(\mathbf{A})$ exactly, which we denote as $\kappa_i^\star$ for brevity. Let $\{a_i\}_{i\in[n]}$ denote the rows of $\mathbf{A}$, and assume that $\|a_i\|_2=1$ for all $i\in[n]$. By scale invariance, this assumption is without loss of generality. We instantiate Assumption 18 with $\mathbf{P}_i=a_ia_i^\top$ and $\mathbf{C}_i=\kappa_i^\star a_ia_i^\top$, for

$i \in [n]$. It is immediate that a solution yields an inner scaling with the same quality up to a $1+\epsilon$ factor, because by assumption (H.4) is feasible so MPC cannot return "infeasible."

We now instantiate the primitives in (H.5) needed by Assumption 18. Throughout, note that $\rho = \kappa_i^\star$ in this setting. Since we run MPC for $\text{poly}(\log n\kappa_i^\star)$ iterations, we will set $\delta' \leftarrow \delta \cdot (\text{poly}(n\kappa_i^\star))^{-1}$ for the failure probability of each of our computations in (H.5), such that by a union bound all of these computations are correct.

By Lemma 323, we can instantiate the packing gradients to the desired approximation quality in time $O(\text{nnz}(\mathbf{A}) \cdot \text{poly}\log \frac{n\kappa_i^\star}{\delta})$ with probability $1-\delta'$. By Lemmas 97 and 98, we can instantiate the covering gradients in time $O(\text{nnz}(\mathbf{A})\sqrt{\kappa_i^\star} \cdot \text{poly}\log \frac{n\kappa_i^\star}{\delta})$ with probability $1-\delta'$. In applying these lemmas, we use the assumption that $\lambda_{\min}(\sum_{i\in[n]} w_i \mathbf{C}_i) = O(\log n\kappa_i^\star)$ as in Assumption 18, and that the covering matrices are a $\kappa_i^\star$ multiple of the packing matrices so $\lambda_{\max}(\sum_{i\in[n]} w_i \mathbf{C}_i) = O(\kappa_i^\star \log n\kappa_i^\star)$. Thus, the overall runtime of all iterations is

$$O\left(\text{nnz}(\mathbf{A}) \cdot \sqrt{\kappa_i^\star} \cdot \text{poly}\log \frac{n\kappa_i^\star}{\delta}\right)$$

for $\epsilon = \Theta(1)$. To remove the assumption that we know $\kappa_i^\star(\mathbf{A})$, we can use an incremental search on the scaling multiple between $\{\mathbf{C}_i\}_{i\in[n]}$ and $\{\mathbf{P}_i\}_{i\in[n]}$, starting from 1 and increasing by factors of $1+\epsilon$, adding a constant overhead to the runtime. Our width will never be larger than $O(\kappa_i^\star(\mathbf{A}))$ in any run, since MPC must conclude feasible when the width is sufficiently large. □

Outer scalings. We next discuss the case where we wish to symmetrically outer scale a matrix $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ near-optimally (i.e. demonstrating an improvement to Theorem 62 under Assumption 18).

Theorem 93. *Under Assumption 18, there is an algorithm which, given $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ computes $w \in \mathbb{R}^d_{\geq 0}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) \leq (1+\epsilon)\kappa_o^\star(\mathbf{K})$ for arbitrarily small $\epsilon \in \Theta(1)$, with probability $\geq 1-\delta$ in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_o^\star(\mathbf{K})} \cdot \text{poly}\log \frac{d\kappa_o^\star(\mathbf{K})}{\delta}\right).$$

Proof. Throughout we denote $\kappa_o^\star := \kappa_o^\star(\mathbf{K})$ for brevity. Our proof follows that of Theorem 62, which demonstrates that it suffices to reduce to the case where we have a $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$ with $\kappa(\mathbf{K}) \leq \kappa_{\text{scale}} := 3\kappa_o^\star$, and we wish to find an outer diagonal scaling $\mathbf{W} \in \mathbb{S}^d_{\succ \mathbf{0}}$ such that $\kappa(\mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}) \leq (1+\epsilon)\kappa_o^\star$. We incur a polylogarithmic overhead on the runtime of this subproblem, by using it to solve all phases of the homotopy method in Theorem 62, and the cost of an incremental search on $\kappa_o^\star$.

To solve this problem, we again instantiate Assumption 18 with $\mathbf{P}_i = a_i a_i^\top$ and $\mathbf{C}_i = \kappa_o^\star a_i a_i^\top$, where $\{a_i\}_{i\in[d]}$ are rows of $\mathbf{A} := \mathbf{K}^{\frac{1}{2}}$. As in the proof of Theorem 62, the main difficulty is to implement the gradients in (H.5) with only implicit access to $\mathbf{A}$, which we again will perform to probability $1-\delta'$ for some $\delta' = \delta \cdot (\text{poly}(n\kappa_o^\star))^{-1}$ which suffices by a union bound. Applying Lemmas 157 and 158 with the same parameters as in the proof of Theorem 62 (up to constants)

implies that we can approximate the covering gradients in (H.5) to the desired quality within time

$$O\left(\mathcal{T}_{\mathrm{mv}}(\mathbf{K}) \cdot \sqrt{\kappa_o^\star} \cdot \mathrm{poly}\log \frac{n\kappa_o^\star}{\delta}\right).$$

Similarly, Corollary 72 implies we can compute the necessary approximate packing gradients in the same time. Multiplying by the overhead of the homotopy method in Theorem 62 gives the result. □

Average-case conditioning under Assumption 18

A number of recent linear system solvers depend on average notions of conditioning, namely the ratio between the average eigenvalue and smallest [497, 347, 300, 168, 553, 17, 10]. Normalized by dimension, we define this average conditioning as follows: for $\mathbf{M} \in \mathbb{S}^d_{\succ \mathbf{0}}$,

$$\tau\left(\mathbf{M}\right) := \frac{\mathrm{Tr}(\mathbf{M})}{\lambda_{\min}(\mathbf{M})}.$$

Observe that since $\mathrm{Tr}(\mathbf{M})$ is the sum of eigenvalues, the following inequalities always hold:

$$d \leq \tau\left(\mathbf{M}\right) \leq d\kappa\left(\mathbf{M}\right). \tag{H.6}$$

In analogy with $\kappa_i^\star$ and $\kappa_o^\star$, we define for full-rank $\mathbf{A} \in \mathbb{R}^{n \times d}$ for $n \geq d$, and $\mathbf{K} \in \mathbb{S}^d_{\succ \mathbf{0}}$,

$$\tau_i^\star(\mathbf{A}) := \min_{\text{diagonal } \mathbf{W} \succeq \mathbf{0}} \tau\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right), \ \tau_o^\star(\mathbf{K}) := \min_{\text{diagonal } \mathbf{W} \succeq \mathbf{0}} \tau\left(\mathbf{W}^{\frac{1}{2}} \mathbf{K} \mathbf{W}^{\frac{1}{2}}\right). \tag{H.7}$$

We give an informal discussion on how to use Assumption 18 to develop a solver for approximating $\tau_i^\star$ to a constant factor, which has a runtime nearly-matching the fastest linear system solvers depending on $\tau_i^\star$ after applying the appropriate rescalings.[3] Qualitatively, this may be thought of as the average-case variant of Theorem 92. We defer an analogous result on approximating $\tau_o^\star$ (with or without a factorization) to future work for brevity.

To develop our algorithm for approximating $\tau_i^\star$, we require several tools. The first is the rational approximation analog of the polynomial approximation in Fact 12.

Fact 45 (Rational approximation of exp [466], Theorem 7.1)**.** *Let $\mathbf{M} \in \mathbb{S}^d_{\geq 0}$ and $\delta > 0$. There is an explicit polynomial p of degree $\Delta = \Theta(\log(\delta^{-1}))$ with absolute coefficients at most $\Delta^{O(\Delta)}$ with*

$$\exp(-\mathbf{M}) - \delta\mathbf{I} \preceq p\left(\left(\mathbf{I} + \frac{\mathbf{M}}{\Delta}\right)^{-1}\right) \preceq \exp(-\mathbf{M}) + \delta\mathbf{I}.$$

We also use the runtime of the fastest-known solver for linear systems based on row subsampling,

[3] We remark that these problems may be solved to high precision by casting them as an appropriate SDP and applying general SDP solvers, but in this section we focus on fast runtimes.

with a runtime dependent on the average conditioning τ. Our goal is to compute reweightings $\mathbf{W}$ which approximately attain the minimums in (H.7), with runtimes comparable to that of Fact 46.

Fact 46 ([10]). *There is an algorithm which given $\mathbf{M} \in \mathbb{S}^d_{\succ \mathbf{0}}$, $b \in \mathbb{R}^d$, and $\delta, \epsilon \in (0,1)$ returns $v \in \mathbb{R}^d$ such that $\left\|v - \mathbf{M}^{-1}b\right\|_2 \leq \epsilon \left\|\mathbf{M}^{-1}b\right\|_2$ with probability $\geq 1-\delta$ in time*

$$O\left(\left(n + \sqrt{d\tau(\mathbf{M})}\right) \cdot d \cdot \operatorname{poly}\log \frac{n\tau(\mathbf{K})}{\delta\epsilon}\right).$$

Remark. The runtime of Fact 46 applies more broadly to quadratic optimization problems in $\mathbf{M}$, e.g. regression problems of the form $\|\mathbf{A}x - b\|_2^2$ where $\mathbf{A}^\top \mathbf{A} = \mathbf{M}$. Moreover, Fact 46 enjoys runtime improvements when the rows of $\mathbf{M}$ (or the factorization component $\mathbf{A}$) are sparse; our methods in the following discussion do as well as they are directly based on Fact 46, and we omit this discussion for simplicity. Finally, [10] demonstrates how to improve the dependence on $\sqrt{d\tau(\mathbf{M})}$ to a more fine-grained quantity in the case of non-uniform eigenvalue distributions. We defer obtaining similar improvements for approximating optimal rescalings to interesting future work.

We now give a sketch of how to use Facts 45 and 46 to obtain near-optimal runtimes for computing a rescaling approximating $\tau_i^\star$ under Assumption 18. Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ for $n \geq d$ be full rank, and assume that we known $\tau_i^\star := \tau_i^\star(\mathbf{A})$ for simplicity, which we can approximate using an incremental search with a logarithmic overhead. Denote the rows of $\mathbf{A}$ by $\{a_i\}_{i \in [n]}$. We instantiate Assumption 18 with

$$\mathbf{P}_i = \|a_i\|_2^2, \ \mathbf{C}_i = \tau_i^\star a_i a_i^\top, \ \text{for all } i \in [n], \tag{H.8}$$

from which it follows that (H.4) is feasible using the reweighting $\mathbf{W} = \mathbf{diag}\,(w)$ attaining $\tau_i^\star$:

$$\begin{aligned} \lambda_{\max}\left(\sum_{i\in[n]} w_i \mathbf{P}_i\right) &= \lambda_{\max}\left(\sum_{i\in[n]} w_i \|a_i\|_2^2\right) = \operatorname{Tr}\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right), \\ \lambda_{\min}\left(\sum_{i\in[n]} w_i \mathbf{C}_i\right) &= \tau_i^\star \lambda_{\min}\left(\sum_{i\in[n]} w_i a_i a_i^\top\right) = \tau_i^\star \lambda_{\min}\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right). \end{aligned} \tag{H.9}$$

Hence, if we can efficiently implement each step of MPC with these matrices, it will return a reweighting satisfying (H.3), which yields a trace-to-bottom eigenvalue ratio approximating $\tau_i^\star$ to a $1+\epsilon$ factor. We remark that in the algorithm parameterization, we have $\rho = \tau_i^\star$. Moreover, all of the packing gradient computations in (H.5) are one-dimensional and hence amount to vector operations, so we will only discuss the computation of covering gradients.

Next, observe that Assumption 18 guarantees that for all intermediate reweightings $\mathbf{W}$ computed by the algorithm and $R = O(\log n\tau_i^\star)$, $\lambda_{\max}(\sum_{i\in[n]} w_i \mathbf{P}_i) = \operatorname{Tr}(\mathbf{A}^\top \mathbf{W} \mathbf{A}) \leq R$. This implies that

the trace of the matrix involved in covering gradient computations is always bounded:

$$\mathrm{Tr}\left(\sum_{i\in[n]} w_i \mathbf{C}_i\right) = \tau_i^\star \mathrm{Tr}\left(\mathbf{A}^\top \mathbf{W}\mathbf{A}\right) \leq \tau_i^\star R. \tag{H.10}$$

To implement the covering gradient computations, we appropriately modify Lemmas 97 and 98 to use the rational approximation in Fact 45 instead of the polynomial approximation in Fact 12. It is straightforward to check that the degree of the rational approximation required is $\Delta = O(\log n\tau_i^\star)$.

Moreover, each of the Δ linear systems which Fact 45 requires us to solve is in the matrix

$$\mathbf{M} := \mathbf{I} + \frac{\sum_{i\in[n]} w_i \mathbf{C}_i}{\Delta},$$

which by (H.10) and the fact that $\mathbf{I}$ has all eigenvalues 1, has $\tau(\mathbf{M}) = O(\tau_i^\star)$. Thus, we can apply Fact 46 to solve these linear systems in time

$$O\left(\left(n + \sqrt{d\tau_i^\star}\right) \cdot d \cdot \mathrm{poly}\log \frac{n\tau_i^\star}{\delta}\right).$$

Here, we noted that the main fact that e.g. Lemmas 97 and 98 use is that the rational approximation approximates the exponential up to a $\mathrm{poly}(n^{-1}, (\tau_i^\star)^{-1})$ multiple of the identity. Since all coefficients of the polynomial in Fact 45 are bounded by $\Delta^{O(\Delta)}$, the precision to which we need to apply Fact 46 to satisfy the requisite approximations is $\epsilon = \Delta^{-O(\Delta)}$, which only affects the runtime by polylogarithmic factors. Combining the cost of computing (H.5) with the iteration bound of Assumption 18, the overall runtime of our method for approximating $\tau_i^\star$ is

$$O\left(\left(n + \sqrt{d\tau_i^\star(\mathbf{A})}\right) \cdot d \cdot \mathrm{poly}\log \frac{n\tau_i^\star(\mathbf{A})}{\delta}\right),$$

which matches Fact 46's runtime after rescaling in all parameters up to logarithmic factors.

H.2 Greedy and non-convex methods fail in the semi-random setting

In this section, we show how a few standard, commonly-used non-convex or greedy methods can fail (potentially quite drastically) in the semi-random adversary setting. The two algorithms that we examine are Iterative Hard Thresholding and Orthogonal Matching Pursuit [86, 510]. We believe it is likely that similar counterexamples can be constructed for other, more complex algorithms such as CoSaMP [410]. For simplicity in this section, we will only discuss the specific semi-random model introduced in Definition 37, where $\mathbf{A}$ is pRIP, i.e. it contains an unknown RIP matrix $\mathbf{G}$ as a subset

of its rows.

H.2.1 Iterative hard thresholding

The iterative hard thresholding algorithm [86] involves initializing $x_0 = 0$ and taking

$$x_{t+1} = H_s\left(x_t - \frac{1}{n}\mathbf{A}^\top(b - \mathbf{A}x_t)\right)$$

where H_s zeroes out all but the s largest entries in magnitude (ties broken lexicographically). We can break this algorithm in the semi-random setting by simply duplicating one row many times.

Hard semi-random adversary. Let $n = Cm$ for some sufficiently large constant C. The first m rows of $\mathbf{A}$ are drawn independently from $\mathcal{N}(0, \mathbf{I})$. Now draw $v \sim \mathcal{N}(0, \mathbf{I})$, except set the first entry of v to 1. We set the last $(C-1)m$ rows of $\mathbf{A}$ all equal to v. We will set the sparsity parameter $s = 1$ and let $x^\star = (1, 0, \ldots, 0)$. We let $b = \mathbf{A}x^\star$.

Proposition 77. *With $\mathbf{A}, b$ generated as above, with high probability, iterative hard thresholding does not converge.*

Proof. With high probability, some coordinate of v is $\Omega(\sqrt{\log d})$. We then have that some entry of $\mathbf{A}^\top b$ has magnitude at least $\Omega(n\sqrt{\log d})$ with high probability. Thus, the next iterate x_1 must have exactly one nonzero entry that has magnitude at least $\Omega(\sqrt{\log d})$ and furthermore, this entry must correspond to some coordinate of v that has magnitude at least $\Omega(\sqrt{\log d})$. However, this means that the residuals in all of the rows that are copies of v are at least $\Omega(\log d)$. In the next step, by the same argument, we get that the residuals blow up even more and clearly this algorithm will never converge. In fact, x_t will never have the right support because its support will always be on one of the entries where v is large. □

H.2.2 Orthogonal matching pursuit

The orthogonal matching pursuit algorithm [510] involves initializing $x_0 = 0$ and keeping track of a set S (that corresponds to our guess of the support of $x^\star$). Each iteration, we choose a column c_j of $\mathbf{A}$ that maximizes $\frac{|\langle c_j, r_t\rangle|}{\|c_j\|_2^2}$ and then add j to S (where $r_t = \mathbf{A}x_t - b$ is the residual). We then add j to S and project the residual onto the orthogonal complement of all coordinates in S. We show that we can again very easily break this algorithm in the semi-random setting.

Hard semi-random adversary. Let $n = 3m$. First, we draw all rows of $\mathbf{A}$ independently from $\mathcal{N}(0, \mathbf{I})$. Next, we modify some of the entries in the last $2m$ rows. Let s be the sparsity parameter. Let $x^\star = (s^{-\frac{1}{2}}, \ldots, s^{-\frac{1}{2}}, 0, \ldots, 0)$ be supported on the first s coordinates and set $b = \mathbf{A}x^\star$. Now we modify the columns of $\mathbf{A}$ (aside from the first s so $\mathbf{A}x^\star$ is not affected). We set the last $2m$ entries of one of these columns c_j to match those of b.

Proposition 78. *With* $\mathbf{A}, b$ *generated as above, with high probability, orthogonal matching pursuit does not recover* $x^\star$*.*

Proof. With high probability (as long as $s \geq 10$), the column c_j is the one that maximizes $\frac{|\langle c_j, b\rangle|}{\|c_j\|_2^2}$ because its last $2m$ entries exactly match those of b. However, j is not in the support of $x^\star$ so the algorithm has already failed. □

We further make the following observation.

Remark 15. *By modifying other columns of* $\mathbf{A}$ *as well, the semi-random adversary can actually make the algorithm pick all of the wrong columns in the support.*

H.2.3 Convex methods

Now we briefly comment on how convex methods are robust, in the sense that they can still be used in the semi-random setting (but may have substantially slower rates than their fast counterparts). In the noiseless observations case, this is clear because the additional rows of $\mathbf{A}$ are simply additional constraints that are added to the standard ℓ_1 minimization convex program.

In the noisy case, let the target error be $\theta = \left\|\xi_{(m)}\right\|_2$. We then solve the modified problem

$$\begin{aligned} &\min \|x\|_1 \\ &\text{subject to } \left\|[\mathbf{A}x - b]_{(m)}\right\|_2 \leq \theta. \end{aligned}$$

Note that the above is a convex program and thus can be solved in polynomial time by e.g. cutting plane methods [255]. Also, note that $x^\star$ is indeed feasible for the second constraint. Now for the solution $\widehat{x}$ that we obtain, we must have $\|\widehat{x}\|_1 \leq \|x^\star\|_1$ and

$$\left\|[\mathbf{A}(x^\star - \widehat{x})]_{(m)}\right\|_2 \leq 2\theta.$$

Let $\mathbf{G}$ be the set of m randomly generated rows of $\mathbf{A}$ under our semi-random adversarial model. The previous two conditions imply

- $\|\widehat{x} - x^\star\|_1 \leq 2\sqrt{s}\,\|x^\star - \widehat{x}\|_2$
- $\|\mathbf{G}(x^\star - \widehat{x})\|_2 \leq 2\theta$

which now by restricted strong convexity of $\mathbf{G}$ (see [9]) implies that $\|x^\star - \widehat{x}\|_2 = O(\frac{\theta}{\sqrt{m}})$. We can furthermore round $\widehat{x}$ to s-sparse to obtain the sparse vector x', and the above bound only worsens by a factor of 2 for x' (see Lemma 170 for this argument).

H.3 Proof of Lemma 171

Lemma 171. *Let $\delta > 0$ and $\theta \geq 0$. For any vector $\gamma \in \mathbb{R}^d$, we can solve the optimization problem*

$$\min_{\|p\|_2 \leq \theta} \operatorname{sqmax}_\mu(\gamma - p)$$

to additive accuracy δ in time

$$O\left(d \log^2\left(\frac{\|\gamma\|_2^2}{\mu\sqrt{\delta}}\right)\right).$$

Proof. Let $\mathcal{P} \subset \mathbb{R}^d$ be the set of p such that p has the same sign as γ entrywise and $|p_j| \leq |\gamma_j|$ for all $j \in [d]$. By symmetry of the sqmax and the ℓ_2 norm under negation, the optimal p lies in $\mathcal{P}$.

Next we claim that the function $\operatorname{smax}_\mu(\gamma - p)$ is $2\|\gamma\|_2$-Lipschitz in the ℓ_2 norm as a function of p, over $\mathcal{P}$. To see this, the gradient is directly computable as

$$2(p - \gamma) \circ x \text{ where } x \in \Delta^d \text{ with } x_i = \frac{\exp([\gamma_i - p_i]^2/\mu^2)}{\sum_{j \in [n]} \exp([\gamma_j - p_j]^2/\mu^2)} \text{ for all } i \in [n]$$

where $\circ$ denotes entrywise multiplication. Thus, the ℓ_2 norm of the derivative is bounded by $2\|\gamma\|_2$ over $\mathcal{P}$. In the remainder of the proof, we show how to find $p \in \mathcal{P}$ which has ℓ_2 error $\frac{\delta}{2\|\gamma\|_2}$ to the optimal, which implies by Lipschitzness that the function value is within additive δ of optimal.

Next, since $0 \in \mathcal{P}$, we may assume without loss of generality that

$$\theta > \frac{\delta}{2\|\gamma\|_2}. \tag{H.11}$$

else we may just output 0, which achieves optimality gap at most $2\|\gamma\|_2 \theta$.

Now, by monotonicity of ln it suffices to approximately minimize

$$\sum_{j \in [d]} \exp\left(\frac{[\gamma - p]_j^2}{\mu^2}\right).$$

The sum above is always at least d. First we check if $\|\gamma\|_2 \leq \theta + \sqrt{\delta}$. If this is true then clearly we can set p so that all entries of $\gamma - p$ have magnitude at most $\sqrt{\delta}$. This gives a solution such that

$$\operatorname{sqmax}_\mu(\gamma - p) \leq \mu^2 \log\left(d \exp\left(\frac{\delta}{\mu^2}\right)\right) = \mu^2 \log d + \delta$$

and since the value of sqmax is always at least $\mu^2 \log d$, this solution is optimal up to additive error δ. Thus, we can assume $\|\gamma\|_2 \geq \theta + \sqrt{\delta}$ in the remainder of the proof. We also assume all entries of γ are nonzero since if an entry of γ is 0 then the corresponding entry of p should also be 0. Finally by symmetry of the problem under negation we will assume all entries of γ are positive in the remainder

of the proof, such that each entry of p is also positive.

By monotonicity of sqmax in each coordinate (as long as signs are preserved) and the assumption that $\|\gamma\|_2 \geq \theta + \sqrt{\delta}$, the optimal solution must have $\|p\|_2 = \theta$. By using Lagrange multipliers, for some scalar ζ and all j,

$$p_j = \exp(\zeta) \cdot [\gamma - p]_j \exp\left(\frac{[\gamma - p]_j^2}{\mu^2}\right). \tag{H.12}$$

For the optimal ζ by taking ℓ_2 norms of the quantity above, we have

$$\theta = \|p\|_2 = \zeta \|\gamma - p\|_2 \cdot C \text{ for some } C \in \left[0, \exp\left(\frac{\|\gamma\|_2^2}{\mu^2}\right)\right].$$

Hence taking logarithms of both sides and using both the bounds (H.11) and $\|\gamma - p\|_2 \geq \sqrt{\delta}$ at the optimum, which follows from the previous discussion, we obtain

$$\log \frac{\theta}{\|\gamma - p\|_2} - \zeta \in \left[0, \frac{\|\gamma\|_2^2}{\mu^2}\right] \implies \zeta \in \left[-\frac{\|\gamma\|_2^2}{\mu^2} - \log\left(\frac{2\|\gamma\|_2^2}{\delta}\right), \log\left(\frac{\|\gamma\|_2}{\sqrt{\delta}}\right)\right].$$

We next show how to compute p to high accuracy given a guess on ζ. Observe that if $\gamma_j > 0$, then the right-hand side of (H.12) is decreasing in p_j and hence by the intermediate value theorem, there is a unique solution strictly between 0 and γ_j for any ζ. Also, note that the location of this solution increases with ζ. Let $p(\zeta)$ be the solution obtained by exactly solving (H.12) for some given ζ. We have shown for all ζ that $0 \leq [p(\zeta)]_j \leq \gamma_j$ entrywise and hence $\|p(\zeta)\|_2 \leq \|\gamma\|_2$ for all ζ.

For a fixed ζ, we claim we can estimate $p(\zeta)$ to ℓ_2 error β in time $O(d \log \frac{\|\gamma\|_2}{\beta})$. To see this, fix some ζ, μ, and γ_j, and consider solving (H.12) for the fixed point p_j. We can discretize $[0, \gamma_j]$ into intervals of length $\frac{\gamma_j \beta}{\|\gamma\|_2}$ and perform a binary search. The right-hand side is decreasing in p_j and the left-hand side is increasing so the binary search yields some interval of length $\frac{\gamma_j \beta}{\|\gamma\|_2}$ containing the fixed point p_j via the intermediate value theorem. The resulting ℓ_2 error along all coordinates is then β. We also round this approximate $p(\zeta)$ entrywise down in the above search to form a vector $\tilde{p}(\zeta, \beta)$ such that $\tilde{p}(\zeta, \beta) \leq p(\zeta)$ entrywise and $\|\tilde{p}(\zeta, \beta) - p(\zeta)\|_2 \leq \beta$. We use this notation and it is well-defined as the search is deterministic.

In the remainder of the proof we choose the constants

$$\alpha := \frac{\delta^2}{192\|\gamma\|_2^4}, \ \beta := \min\left(\frac{\delta^2}{192\|\gamma\|_2^3}, \frac{\delta}{4\|\gamma\|_2}\right).$$

We define $\tilde{p}(\zeta) := \tilde{p}(\zeta, \beta)$ for short as β will be fixed. Discretize the range $[-\frac{\|\gamma\|_2^2}{\mu^2} - \log \frac{2\|\gamma\|_2^2}{\delta}, \log \frac{\|\gamma\|_2}{\sqrt{\delta}}]$ into a grid of uniform intervals of length α. Consider the ζ such that $\zeta \leq \zeta^\star < \zeta + \alpha$. Because $p(\zeta^\star)$ is entrywise larger than $p(\zeta)$ and hence the logarithmic term on the right-hand side of (H.12)

is smaller for $p(\zeta^\star)$ than $p(\zeta)$, we have

$$[p(\zeta)]_j \le [p(\zeta^\star)]_j \le \exp(\alpha)\,[p(\zeta)]_j\,.$$

Moreover the optimal $p(\zeta^\star)$ has ℓ_2 norm θ, so $|\zeta - \zeta^\star| \le \alpha$ and $\exp(\alpha) - 1 \le 2\alpha$ imply

$$\|p(\zeta) - p(\zeta^\star)\|_2 \le 2\alpha\,\|p(\zeta^\star)\|_2 \le 2\alpha\,\|\gamma\|_2 \le \Delta := \frac{\delta^2}{96\,\|\gamma\|_2^3}.$$

Consider the algorithm which returns the ζ_{alg} on the search grid which minimizes $|\|\tilde{p}(\zeta_{\text{alg}})\|_2 - \theta|$ (we will discuss computational issues at the end of the proof). As we have argued above, there is a choice which yields $\|p(\zeta)\|_2 \in [\theta - \Delta, \theta + \Delta]$ and hence

$$\|\tilde{p}(\zeta_{\text{alg}})\|_2 \in [\theta - \Delta - \beta, \theta + \Delta + \beta]\,. \tag{H.13}$$

We next claim that

$$\|p(\zeta_{\text{alg}}) - p(\zeta^\star)\|_2 \le \frac{\delta}{4\,\|\gamma\|_2}. \tag{H.14}$$

Suppose (H.14) is false and $\zeta_{\text{alg}} > \zeta^\star$. Then letting $u := p(\zeta_{\text{alg}})$ and $v := p(\zeta^\star)$, note that u, v, and $u - v$ are all entrywise nonnegative and hence

$$\|u\|_2^2 \ge \|v\|_2^2 + \sum_{i\in[n]} 2u_i(u_i - v_i) + (u_i - v_i)^2 > \|v\|_2^2 + \left(\frac{\delta}{4\,\|\gamma\|_2}\right)^2.$$

Hence, we have by $\sqrt{x^2 + y^2} \ge x + \frac{y^2}{3x}$ for $0 \le y \le x$, (H.11), and $\theta \le \|\gamma\|_2$,

$$\|p(\zeta_{\text{alg}})\|_2 = \|u\|_2 > \|v\|_2 + \frac{\left(\frac{\delta}{4\|\gamma\|_2}\right)^2}{3\,\|v\|_2} \ge \theta + \frac{\delta^2}{48\,\|\gamma\|_2^3} \ge \theta + \Delta + 2\beta.$$

So, by triangle inequality $\|\tilde{p}(\zeta_{\text{alg}})\|_2 > \theta + \Delta + \beta$ and hence we reach a contradiction with (H.13).

Similarly, suppose (H.14) is false and $\zeta_{\text{alg}} < \zeta^\star$. Then for the same definitions of u, v, and using the inequality $\sqrt{x^2 - y^2} \le x - \frac{y^2}{3x}$ for $0 \le y \le x$, we conclude

$$\|v\|_2^2 > \|u\|_2^2 + \left(\frac{\delta}{4\,\|\gamma\|_2}\right)^2 \implies \|u\|_2 \le \sqrt{\|v\|_2^2 - \left(\frac{\delta}{4\,\|\gamma\|_2}\right)^2} < \theta - \Delta - 2\beta.$$

So we reach a contradiction with (H.13) in this case as well.

In conclusion, (H.14) is true and we obtain by triangle inequality the desired

$$\|\tilde{p}(\zeta_{\text{alg}}) - p(\zeta^\star)\|_2 \le \frac{\delta}{4\,\|\gamma\|_2} + \beta\sqrt{d} \le \frac{\delta}{2\,\|\gamma\|_2}.$$

The complexity of the algorithm is bottlenecked by the cost of finding $\tilde{p}(\zeta_{\text{alg}})$. For each ζ on the grid the cost of evaluating $\tilde{p}(\zeta)$ induces a multiplicative $d \log(\frac{\|\gamma\|_2^2}{\delta})$ overhead. The cost of performing the binary search on the ζ grid is a multiplicative $\log(\frac{\|\gamma\|_2^2}{\mu\sqrt{\delta}})$ overhead; note that a binary search suffices because $\|\tilde{p}(\zeta_{\text{alg}})\|_2$ is monotonic by our consistent choice of rounding down, and hence $|\,\|\tilde{p}(\zeta_{\text{alg}})\|_2 - \theta|$ is unimodal. □

Appendix I

Deferred proofs from Chapter 10

I.1 Discussion of inexactness tolerance

We briefly discuss the tolerance of our algorithm to approximation error in two places: computation of minimizers, and implementation of RGOs in the methods of Sections 10.3 and 10.5.

Inexact minimization. For all function classes considered in this work, there exist efficient optimization methods converging to a minimizer with logarithmic dependence on the target accuracy.

Specifically, for negative log-densities with condition number κ, accelerated gradient descent [417] converges at a rate $O(\sqrt{\kappa})$ with logarithmic dependence on initial error and target accuracy (we implicitly assumed in stating our runtimes that one can attain initial error polynomial in problem parameters for negative log-densities; otherwise, there is additional logarithmic overhead in the quality of the initial point to optimization procedures). For composite functions $f_{\text{wc}} + f_{\text{oracle}}$ where f_{wc} has condition number κ, the FISTA method of [66] converges at the same rate with each iteration querying ∇f_{wc} and a proximal oracle for f_{oracle} once; typically, access to a proximal oracle is a weaker assumption than access to a restricted Gaussian oracle, so this is not restrictive. Finally, for minimizing finite sums with condition number κ, the algorithm of [17] obtains a convergence rate linearly dependent on $n + \sqrt{n\kappa} \leq n + \kappa$; alternatively, [300] has a dependence on $n + \kappa$. In all our final runtimes, these optimization rates do not constitute the bottleneck for oracle complexities.

The only additional difficulty our algorithms may present is if the function requiring minimization, say of the form $f_{\text{oracle}}(x) + \frac{1}{2\eta} \|x - y\|_2^2$ for some $y \in \mathbb{R}^d$ where we have computed the minimizer x^* to f_{oracle}, has $\|y - x^*\|_2^2$ very large (so the initial function error is bad). However, in all our settings y is drawn from a distribution with sub-Gaussian tails, so $\|y - x^*\|_2^2$ decays exponentially (whereas the complexity of first-order methods increases only logarithmically), negligibly affecting the expected oracle query complexity for our methods.

Finally, by solving the relevant optimization problems to high accuracy as a subroutine in each of our methods, and adjusting various distance bounds to the minimizer by constants (e.g. by expanding

the radius in the definition of the sets Ω in Algorithm 62 and Section 10.6.2), this accomodates tolerance to inexact minimization and only affects all bounds throughout the paper by constants. The only other place that x^* is used in our algorithms is in initializing warm starts; tolerance to inexactness in our warmness calculations follows essentially identically to Section 3.2.1 of [210].

Inexact oracle implementation. Our algorithms based on restricted Gaussian oracle access are tolerant to total variation error inverse polynomial in problem parameters for the restricted Gaussian oracle for g. We discussed this at the end of Section 10.3, in the case of RGO use for our reduction framework. To see this in the case of the composite sampler in Section 10.5, we pessimistically handled the case where the sampler `Y-Oracle` for a quadratic restriction of f resulted in total variation error in the proof of Proposition 46, assuming that the error was incurred in every iteration. By accounting for similar amounts of error in calls to $\mathcal{O}$ (on the order of $\frac{\epsilon}{T}$, where T is the number of times an RGO was used), the bounds in our algorithm are only affected by constants.

I.2 Deferred proofs from Section 10.5

I.2.1 Deferred proofs from Section 10.5.2

Approximate rejection sampling

We first define the rejection sampling framework we will use, and prove various properties.

Definition 60 (Approximate rejection sampling)**.** *Let π be a distribution, with $\frac{d\pi}{dx}(x) \propto p(x)$. Suppose set Ω has $\pi(\Omega) = 1 - \epsilon'$, and distribution $\hat{\pi}$ with $\frac{d\hat{\pi}}{dx}(x) \propto \hat{p}(x)$ has for some $C \geq 1$,*

$$\frac{p(x)}{\hat{p}(x)} \leq C \text{ for all } x \in \Omega, \text{ and } \frac{\int \hat{p}(x)dx}{\int p(x)dx} \leq 1.$$

Suppose there is an algorithm $\mathcal{A}$ which draws samples from a distribution $\hat{\pi}'$, such that $\|\hat{\pi}' - \hat{\pi}\|_{\text{TV}} \leq 1 - \delta$. We call the following scheme approximate rejection sampling: repeat independent runs of the following procedure until a point is outputted.

1. *Draw x via $\mathcal{A}$ until $x \in \Omega$.*
2. *With probability $\frac{p(x)}{C\hat{p}(x)}$, output x.*

Lemma 324. *Consider an approximate rejection sampling scheme with relevant parameters defined as in Definition 60, with $2\delta \leq \frac{1-\epsilon'}{C}$. The algorithm terminates in at most*

$$\frac{1}{\frac{1-\epsilon'}{C} - 2\delta} \tag{I.1}$$

calls to $\mathcal{A}$ in expectation, and outputs a point from a distribution π' with $\|\pi' - \pi\|_{\text{TV}} \leq \epsilon' + \frac{2\delta C}{1-\epsilon'}$.

Proof. Define for notational simplicity normalization constants $Z := \int p(x)dx$ and $\hat{Z} := \int \hat{p}(x)dx$. First, we bound the probability any particular call to $\mathcal{A}$ returns in the scheme:

$$\begin{aligned}\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x) &\geq \int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x) - \left| \int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} (d\hat{\pi}'(x) - d\hat{\pi}(x)) \right| \\ &= \int_{x\in\Omega} \frac{Z}{C\hat{Z}} d\pi(x) - \left| \int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} (d\hat{\pi}'(x) - d\hat{\pi}(x)) \right| \\ &\geq \frac{1-\epsilon'}{C} - \int_{x\in\Omega} |d\hat{\pi}'(x) - d\hat{\pi}(x)| \geq \frac{1-\epsilon'}{C} - 2\delta.\end{aligned} \tag{I.2}$$

The second line followed by the definitions of Z and $\hat{Z}$, and the third followed by triangle inequality, the assumed lower bound on $Z/\hat{Z}$, and the total variation distance between $\hat{\pi}'$ and $\hat{\pi}$. By linearity of expectation and independence, this proves the first claim.

Next, we claim the output distribution is close in total variation distance to the conditional distribution of π restricted to Ω. The derivation of (I.2) implies

$$\begin{aligned}&\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x) \geq \frac{1-\epsilon'}{C},\ \left| \int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} (d\hat{\pi}'(x) - d\hat{\pi}(x)) \right| \leq 2\delta, \\ &\qquad \implies 1 - \frac{2\delta C}{1-\epsilon'} \leq \frac{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x)} \leq 1 + \frac{2\delta C}{1-\epsilon'}.\end{aligned} \tag{I.3}$$

Thus, the total variation of the true output distribution from π restricted to Ω is

$$\begin{aligned}&\frac{1}{2} \int_{x\in\Omega} \left| \frac{d\pi(x)}{1-\epsilon'} - \frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)} \right| \\ \leq\ &\frac{1}{2} \int_{x\in\Omega} \left| \frac{d\pi(x)}{1-\epsilon'} - \frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x)} \right| + \frac{1}{2} \int_{x\in\Omega} \left| \frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x)} - \frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)} \right| \\ \leq\ &\frac{1}{2} \int_{x\in\Omega} \left| \frac{d\pi(x)}{1-\epsilon'} - \frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x)} \right| + \frac{\delta C}{1-\epsilon'} = \frac{1}{2} \int_{x\in\Omega} \frac{d\pi(x)}{1-\epsilon'} \left| 1 - \frac{d\hat{\pi}'}{d\hat{\pi}}(x) \right| + \frac{\delta C}{1-\epsilon'}.\end{aligned}$$

The first inequality was triangle inequality, and we bounded the second term by (I.3). To obtain the final equality, we used

$$\begin{aligned}&\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x) = \int_{x\in\Omega} \frac{Z}{C\hat{Z}} d\pi(x) = \frac{(1-\epsilon')Z}{C\hat{Z}} \\ \implies\ &\frac{\frac{p(x)}{C\hat{p}(x)} d\hat{\pi}'(x)}{\int_{x\in\Omega} \frac{p(x)}{C\hat{p}(x)} d\hat{\pi}(x)} = \frac{p(x)}{Z} \cdot \frac{\hat{Z}}{\hat{p}(x)} \cdot \frac{1}{1-\epsilon'} \cdot d\hat{\pi}'(x) = \frac{d\pi(x)}{1-\epsilon'} \cdot \frac{d\hat{\pi}'}{d\hat{\pi}}(x).\end{aligned}$$

We now bound this final term. Observe that the given conditions imply that $\frac{d\pi}{d\hat{\pi}}(x)$ is bounded by

C everywhere in Ω. Thus, expanding we have

$$\frac{1}{2}\int_{x\in\Omega}\frac{d\pi(x)}{1-\epsilon'}\left|1-\frac{d\hat{\pi}'}{d\hat{\pi}}(x)\right| \le \frac{C}{2(1-\epsilon')}\int_{x\in\Omega}|d\hat{\pi}(x)-d\hat{\pi}'(x)| \le \frac{\delta C}{1-\epsilon'}.$$

Finally, combining these guarantees, and the fact that restricting π to Ω loses ϵ' in total variation distance, yields the desired conclusion by triangle inequality. □

Corollary 73. *Let $\hat{\theta}(x)$ be an unbiased estimator for $\frac{p(x)}{\tilde{p}(x)}$, and suppose $\hat{\theta}(x) \le C$ with probability 1 for all $x \in \Omega$. Then, implementing the procedure of Definition 60 with acceptance probability $\frac{\hat{\theta}(x)}{C}$ has the same runtime bound and total variation guarantee as given by Lemma 324.*

Proof. It suffices to take expectations over the randomness of $\hat{\theta}$ everywhere in the proof of Lemma 324. □

Distribution ratio bounds

We next show two bounds relating the densities of distributions π and $\hat{\pi}$. We first define the normalization constants of (10.20), (10.22) for shorthand, and then tightly bound their ratio.

Definition 61 (Normalization constants)**.** *We denote normalization constants of π and $\hat{\pi}$ by*

$$Z_\pi := \int_x \exp\left(-f(x)-g(x)\right)dx,$$
$$Z_{\hat{\pi}} := \int_{x,y}\exp\left(-f(y)-g(x)-\frac{1}{2\eta}\left\|y-x\right\|_2^2-\frac{\eta L^2}{2}\left\|x-x^*\right\|_2^2\right)dxdy.$$

Lemma 325 (Normalization constant bounds)**.** *Let Z_π and $Z_{\hat{\pi}}$ be as in Definition 61. Then,*

$$\left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}}\left(1+\frac{\eta L^2}{\mu}\right)^{-\frac{d}{2}} \le \frac{Z_{\hat{\pi}}}{Z_\pi} \le (2\pi\eta)^{\frac{d}{2}}.$$

Proof. For each x, by convexity we have

$$\begin{aligned}
&\int_y \exp\left(-f(y)-g(x)-\frac{1}{2\eta}\left\|y-x\right\|_2^2-\frac{\eta L^2}{2}\left\|x-x^*\right\|_2^2\right)dy \\
&\le \exp\left(-g(x)-\frac{\eta L^2}{2}\left\|x-x^*\right\|_2^2\right)\int_y \exp\left(-f(x)-\langle\nabla f(x),y-x\rangle-\frac{1}{2\eta}\left\|y-x\right\|_2^2\right)dy \\
&= \exp\left(-f(x)-g(x)-\frac{\eta L^2}{2}\left\|x-x^*\right\|_2^2\right)\int_y \exp\left(\frac{\eta}{2}\left\|\nabla f(x)\right\|_2^2-\frac{1}{2\eta}\left\|y-x+\eta\nabla f(x)\right\|_2^2\right)dy \\
&= (2\pi\eta)^{\frac{d}{2}}\exp\left(-f(x)-g(x)\right)\exp\left(\frac{\eta}{2}\left\|\nabla f(x)\right\|_2^2-\frac{\eta L^2}{2}\left\|x-x^*\right\|_2^2\right) \\
&\le (2\pi\eta)^{\frac{d}{2}}\exp\left(-f(x)-g(x)\right).
\end{aligned} \tag{I.4}$$

Integrating both sides over x yields the upper bound on $\frac{Z_{\hat{\pi}}}{Z_\pi}$. Next, for the lower bound we have a similar derivation. For each x, by smoothness

$$\begin{aligned}
&\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dy \\
&\geq \exp\left(-f(x) - g(x) - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) \int_y \exp\left(\langle \nabla f(x), x-y\rangle - \frac{1+\eta L}{2\eta}\|y-x\|_2^2\right) dy \\
&= \exp\left(-f(x) - g(x) - \frac{\eta L^2}{2}\|x-x^*\|^2 + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|^2\right) \left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}} \\
&\geq \exp\left(-f(x) - g(x) - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) \left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}}.
\end{aligned}$$

Integrating both sides over x yields

$$\frac{Z_{\hat{\pi}}}{Z_\pi} \geq \left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}} \frac{\int_x \exp\left(-f(x) - g(x) - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dx}{\int_x \exp\left(-f(x) - g(x)\right) dx} \geq \left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}} \left(1 + \frac{\eta L^2}{\mu}\right)^{-\frac{d}{2}}.$$

The last inequality followed from Proposition 82, where we used $f+g$ is μ-strongly convex. □

Lemma 326 (Relative density bounds)**.** *Let $\eta = \frac{1}{32 L \kappa d \log(288\kappa/\epsilon)}$. For all $x \in \Omega$, as defined in* (10.21)*, $\frac{d\pi}{d\hat{\pi}}(x) \leq 2$. Here, $\frac{d\hat{\pi}}{dx}(x)$ denotes the marginal density of $\hat{\pi}$. Moreover, for all $x \in \mathbb{R}^d$, $\frac{d\pi}{d\hat{\pi}}(x) \geq \frac{1}{2}$.*

Proof. We first show the upper bound. By Lemma 325,

$$\begin{aligned}
\frac{d\pi}{d\hat{\pi}}(x) &= \frac{\exp\left(-f(x) - g(x)\right)}{\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dy} \cdot \frac{Z_{\hat{\pi}}}{Z_\pi} \\
&\leq \frac{\exp\left(-f(x) - g(x)\right)}{\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dy} \cdot (2\pi\eta)^{\frac{d}{2}}.
\end{aligned} \tag{I.5}$$

We now bound the first term, for $x \in \Omega$. By smoothness, we have

$$\frac{\exp\left(-f(y) - g(x)\right)}{\exp\left(-f(x) - g(x)\right)} \geq \exp\left(\langle \nabla f(x), x-y\rangle - \frac{L}{2}\|y-x\|_2^2\right),$$

so applying this for each y,

$$\begin{aligned}
&\frac{\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dy}{\exp\left(-f(x)-g(x)\right)} \\
&\geq \exp\left(-\frac{\eta L^2}{2}\|x-x^*\|_2^2\right) \int_y \exp\left(\langle \nabla f(x), x-y\rangle - \frac{1+\eta L}{2\eta}\|y-x\|_2^2\right) dy \\
&= \exp\left(-\frac{\eta L^2}{2}\|x-x^*\|_2^2 + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right) \int_y \exp\left(-\frac{1+\eta L}{2\eta}\left\|x-y-\frac{\eta}{1+\eta L}\nabla f(x)\right\|_2^2\right) dy \\
&\geq \exp\left(-\frac{\eta L^2}{2}\cdot\frac{16d\log(288\kappa/\epsilon)}{\mu}\right)\left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}} \geq \frac{3}{4}\left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}}.
\end{aligned}$$

In the last line, we used that $x \in \Omega$ implies $\|x-x^*\|_2^2 \leq \frac{16d\log(288\kappa/\epsilon)}{\mu}$, and the definition of η. Combining this bound with (I.5), we have the desired

$$\frac{d\pi}{d\hat{\pi}}(x) \leq \frac{4}{3}(1+\eta L)^{\frac{d}{2}} \leq 2.$$

Next, we consider the lower bound. By combining (I.4) with Lemma 325, we have the desired

$$\begin{aligned}
\frac{d\pi}{d\hat{\pi}}(x) &= \frac{\exp\left(-f(x)-g(x)\right)}{\int_y \exp\left(-f(y)-g(x)-\frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right) dy} \cdot \frac{Z_{\hat{\pi}}}{Z_\pi} \\
&\geq (2\pi\eta)^{-\frac{d}{2}} \cdot \left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}} \left(1+\frac{\eta L^2}{\mu}\right)^{-\frac{d}{2}} = \left(\frac{1}{1+\eta L}\right)^{\frac{d}{2}} (1+\eta L\kappa)^{-\frac{d}{2}} \geq \frac{1}{2}.
\end{aligned}$$

□

Correctness of `Composite-Sample-Shared-Min`

Proposition 45. *Let $\eta = \frac{1}{32L\kappa d\log(288\kappa/\epsilon)}$, and assume `Sample-Joint-Dist`$(f, g, x^*, \mathcal{O}, \delta)$ samples within δ total variation of the x-marginal on* (10.22). *`Composite-Sample-Shared-Min` outputs a sample within total variation ϵ of* (10.20) *in an expected $O(1)$ calls to `Sample-Joint-Dist`.*

Proof. We remark that $\eta = \frac{1}{32L\kappa d\log(288\kappa/\epsilon)}$ is precisely the choice of η in `Sample-Joint-Dist` where $\delta = \epsilon/18$, as in `Composite-Sample-Shared-Min`. First, we may apply Fact 28 to conclude that the measure of set Ω with respect to the μ-strongly logconcave density π is at least $1-\epsilon/3$. The conclusion of correctness will follow from an appeal to Corollary 73, with parameters

$$C = 4,\ \epsilon' = \frac{\epsilon}{3},\ \delta = \frac{\epsilon}{18}.$$

Note that indeed we have $\epsilon' + \frac{2\delta C}{1-\epsilon'}$ is bounded by ϵ, as $1-\epsilon' \geq \frac{2}{3}$. Moreover, the expected number of calls (I.1) is clearly bounded by a constant as well.

We now show that these parameters satisfy the requirements of Corollary 73. Define the functions

$$p(x) := \exp(-f(x) - g(x)),$$
$$\hat{p}(x) := (2\pi\eta)^{-\frac{d}{2}} \int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y - x\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) dy,$$

and observe that clearly the densities of π and $\hat{\pi}$ are respectively proportional to p and $\hat{p}$. Moreover, define $Z = \int p(x)dx$ and $\hat{Z} = \int \hat{p}(x)dx$. By comparing these definitions with Lemma 325, we have $Z = Z_\pi$ and $\hat{Z} = (2\pi\eta)^{-\frac{d}{2}} Z_{\hat{\pi}}$, so by the upper bound in Lemma 325, $\hat{Z}/Z \leq 1$. Next, we claim that the following procedure produces an unbiased estimator for $\frac{p(x)}{\hat{p}(x)}$.

1. Sample $y \sim \pi_x$, where $\frac{d\pi_x(y)}{dy} \propto \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right)$
2. $\alpha \leftarrow \exp\left(f(y) - \langle \nabla f(x), y - x\rangle - \frac{L}{2}\|y - x\|_2^2 + g(x) + \frac{\eta L^2}{2}\|x - x^*\|_2^2\right)$
3. Output $\hat{\theta}(x) \leftarrow \exp\left(-f(x) - g(x) + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)(1 + \eta L)^{\frac{d}{2}}\alpha$

To prove correctness of this estimator $\hat{\theta}$, define for simplicity

$$Z_x := \int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y - x\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) dy.$$

We compute, using $\frac{d\pi_x(y)}{dy} = \frac{\exp(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2)}{Z_x}$, that

$$\begin{aligned}
\mathbb{E}_{\pi_x}[\alpha] &= \int_y \exp\left(f(y) - \langle \nabla f(x), y - x\rangle - \frac{L}{2}\|y - x\|_2^2 + g(x) + \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) d\pi_x(y) \\
&= \frac{1}{Z_x}\int_y \exp\left(-\langle \nabla f(x), y - x\rangle - \frac{L}{2}\|y - x\|_2^2 - \frac{1}{2\eta}\|y - x\|_2^2\right) dy \\
&= \frac{1}{Z_x}\exp\left(-\frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)\left(\frac{2\pi\eta}{1+\eta L}\right)^{\frac{d}{2}}.
\end{aligned}$$

This implies that the output quantity

$$\hat{\theta}(x) = \exp\left(-f(x) - g(x) + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)(1 + \eta L)^{\frac{d}{2}}\alpha$$

is unbiased for $\frac{p(x)}{\hat{p}(x)} = \exp(-f(x) - g(x))Z_x^{-1}(2\pi\eta)^{\frac{d}{2}}$. Finally, note that for any y used in the

definition of $\hat{\theta}(x)$, by using $f(y) - f(x) - \langle \nabla f(x), y - x\rangle - \frac{L}{2}\|y-x\|_2^2 \le 0$ via smoothness, we have

$$\begin{aligned}\hat{\theta}(x) &= \exp\left(-f(x) - g(x) + \frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)(1+\eta L)^{\frac{d}{2}}\alpha \\ &\le (1+\eta L)^{\frac{d}{2}}\exp\left(\frac{\eta}{2(1+\eta L)}\|\nabla f(x)\|_2^2 + \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) \\ &\le (1+\eta L)^{\frac{d}{2}}\exp\left(\eta L^2\|x-x^*\|_2^2\right) \le 4.\end{aligned}$$

Here, we used the definition of η and $L^2\|x-x^*\|_2^2 \le 16L\kappa d\log(288\kappa/\epsilon)$ by the definition of Ω. □

I.2.2 Deferred proofs from Section 10.5.3

Throughout this section, for error tolerance $\delta \in [0,1]$ which parameterizes `Sample-Joint-Dist`, we denote for shorthand a high-probability region Ω_δ and its radius R_δ by

$$\Omega_\delta := \{x \mid \|x - x^*\|_2 \le R_\delta\}, \text{ for } R_\delta := 4\sqrt{\frac{d\log(16\kappa/\delta)}{\mu}}. \tag{I.6}$$

The following density ratio bounds hold within this region, by simply modifying Lemma 326.

Corollary 74. *Let $\eta = \frac{1}{32L\kappa d\log(16\kappa/\delta)}$, and let $\hat{\pi}$ be parameterized by this choice of η in* (10.22)*. For all $x \in \Omega_\delta$, as defined in* (I.6)*, $\frac{d\pi}{d\hat{\pi}}(x) \le 2$. Moreover, for all $x \in \mathbb{R}^d$, $\frac{d\pi}{d\hat{\pi}}(x) \ge \frac{1}{2}$.*

The following claim follows immediately from applying Fact 28.

Lemma 327. *With probability at least $1 - \frac{\delta^2}{8(1+\kappa)^d}$, $x \sim \hat{\pi}$ lies in Ω_δ.*

Finally, when clear from context, we overload $\hat{\pi}$ as a distribution on $x \in \mathbb{R}^d$ to be the x component marginal of the distribution (10.22), i.e. with density

$$\frac{d\hat{\pi}}{dx}(x) \propto \int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y-x\|_2^2 - \frac{\eta L^2}{2}\|x-x^*\|_2^2\right)dy.$$

We first note that $\hat{\pi}$ is stationary for `Sample-Joint-Dist`; this follows immediately from Lemma 181. In Section I.2.2, we bound the *conductance* of the walk. We then use this bound in Section I.2.2 to bound the mixing time and overall complexity of `Sample-Joint-Dist`.

Conductance of `Sample-Joint-Dist`

We bound the conductance of this random walk, as a process on the iterates $\{x_k\}$, to show the final point has distribution close to the marginal of $\hat{\pi}$ on x. To do so, we break Proposition 48 into two pieces, which we will use in a more white-box manner to prove our conductance bound.

Definition 62 (Restricted conductance)**.** *Let a random walk with stationary distribution $\hat{\pi}$ on $x \in \mathbb{R}^d$ have transition densities $\mathcal{T}_x$, and let $\Omega \subseteq \mathbb{R}^d$. The Ω-restricted conductance, for $v \in (0, \frac{1}{2}\hat{\pi}(\Omega))$, is*

$$\Phi_\Omega(v) = \inf_{\hat{\pi}(S \cap \Omega) \in (0, v]} \frac{\mathcal{T}_S(S^c)}{\hat{\pi}(S \cap \Omega)}, \text{ where } \mathcal{T}_S(S^c) := \int_{x \in S} \int_{x' \in S^c} \mathcal{T}_x(x') d\hat{\pi}(x) dx'.$$

Proposition 79 (Lemma 1, [128])**.** *Let π_{start} be a β-warm start for $\hat{\pi}$, and let $x_0 \sim \pi_{\text{start}}$. For some $\delta > 0$, let $\Omega \subseteq \mathbb{R}^d$ have $\hat{\pi}(\Omega) \geq 1 - \frac{\delta^2}{2\beta^2}$. Suppose that a random walk with stationary distribution $\hat{\pi}$ satisfies the Ω-restricted conductance bound*

$$\Phi_\Omega(v) \geq \sqrt{B \log\left(\frac{1}{v}\right)}, \text{ for all } v \in \left[\frac{4}{\beta}, \frac{1}{2}\right].$$

Let x_K be the result of K steps of this random walk, starting from x_0. Then, for

$$K \geq \frac{64}{B} \log\left(\frac{\log \beta}{2\delta}\right),$$

the resulting distribution of x_K has total variation at most $\frac{\delta}{2}$ from $\hat{\pi}$.

We state a well-known strategy for lower bounding conductance, via showing the stationary distribution has good *isoperimetry* and that transition distributions of nearby points have large overlap.

Proposition 80 (Lemma 2, [128])**.** *Let a random walk with stationary distribution $\hat{\pi}$ on $x \in \mathbb{R}^d$ have transition distribution densities $\mathcal{T}_x$, and let $\Omega \subseteq \mathbb{R}^d$, and let $\hat{\pi}_\Omega$ be the conditional distribution of $\hat{\pi}$ on Ω. Suppose for any $x, x' \in \Omega$ with $\|x - x'\|_2 \leq \Delta$,*

$$\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\text{TV}} \leq \frac{1}{2}.$$

Also, suppose $\hat{\pi}_\Omega$ satisfies, for any partition S_1, S_2, S_3 of Ω, where $d(S_1, S_2)$ is the minimum Euclidean distance between points in S_1, S_2, the log-isoperimetric inequality

$$\hat{\pi}_\Omega(S_3) \geq \frac{1}{2\psi} d(S_1, S_2) \cdot \min\left(\hat{\pi}_\Omega(S_1), \hat{\pi}_\Omega(S_2)\right) \cdot \sqrt{\log\left(1 + \frac{1}{\min\left(\hat{\pi}_\Omega(S_1), \hat{\pi}_\Omega(S_2)\right)}\right)}. \tag{I.7}$$

Then, we have the bound for all $v \in (0, \frac{1}{2}]$

$$\Phi_\Omega(v) \geq \min\left(1, \frac{\Delta}{128\psi} \sqrt{\log\left(\frac{1}{v}\right)}\right).$$

To utilize Propositions 79 and 80, we prove the following bounds in Appendices I.3.1, I.3.2, and I.3.3.

Lemma 328 (Warm start). *For $\eta \le \frac{1}{L\kappa d}$, π_{start} defined in* (10.23) *is a $2(1+\kappa)^{\frac{d}{2}}$-warm start for $\hat{\pi}$.*

Lemma 329 (Transitions of nearby points). *Suppose $\eta L \le 1$, $\eta L^2 R_\delta^2 \le \frac{1}{2}$, and $400d^2\eta \le R_\delta^2$. For a point x, let $\mathcal{T}_x$ be the density of x_k after sampling according to Lines 6 and 7 of Algorithm 63 from $x_{k-1} = x$. For $x, x' \in \Omega_\delta$ with $\|x - x'\|_2 \le \frac{\sqrt{\eta}}{10}$, for Ω_δ defined in* (I.6), *we have $\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\text{TV}} \le \frac{1}{2}$.*

Lemma 330 (Isoperimetry). *Density $\hat{\pi}$ and set Ω_δ defined in* (10.22), (I.6) *satisfy* (I.7) *with $\psi = 8\mu^{-\frac{1}{2}}$.*

We note that the parameters of Algorithm 63 and the set Ω_δ in (I.6) satisfy all assumptions of Lemmas 328, 329, and 330. By combining these results in the context of Proposition 80, we see that the random walk satisfies the bound for all $v \in (0, \frac{1}{2}]$:

$$\Phi_{\Omega_\delta}(v) \ge \sqrt{\frac{\eta\mu}{2^{20} \cdot 100} \cdot \log\left(\frac{1}{v}\right)}.$$

Plugging this conductance lower bound, the high-probability guarantee of Ω_δ by Lemma 327, and the warm start bound of Lemma 328 into Proposition 79, we have the following conclusion.

Corollary 75 (Mixing time of ideal `Sample-Joint-Dist`). *Assume that calls to `Y-Oracle` are exact in the implementation of `Sample-Joint-Dist`. Then, for any error parameter δ, and*

$$K := \frac{2^{26} \cdot 100}{\eta\mu} \log\left(\frac{d \log(16\kappa)}{4\delta}\right),$$

the distribution of x_K has total variation at most $\frac{\delta}{2}$ from $\hat{\pi}$.

Complexity of `Sample-Joint-Dist`

We first state a guarantee on the subroutine `Y-Oracle`, which we prove in Appendix I.3.4.

Lemma 331 (`Y-Oracle` guarantee). *For $\delta \in [0, 1]$, define R_δ as in* (I.6), *and let $\eta = \frac{1}{32L\kappa d \log(16\kappa/\delta)}$. For any x with $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$, Algorithm 3 (`Y-Oracle`) draws an exact sample y from the density proportional to $\exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right)$ in an expected 2 iterations.*

We also state a result due to [128], which bounds the mixing time of 1-step Metropolized HMC for well-conditioned distributions; this handles the case when $\|x - x^*\|_2$ is large in Algorithm 3.

Proposition 81 (Theorem 1, [128]). *Let π be a distribution on $\mathbb{R}^d$ whose negative log-density is convex and has condition number bounded by a constant. Then, Metropolized HMC from an explicit starting distribution mixes to total variation δ to the distribution π in $O(d \log(\frac{d}{\delta}))$ iterations.*

Proposition 46. *`Sample-Joint-Dist` outputs a point with distribution within δ total variation distance from the x-marginal of $\hat{\pi}$. The expected number of gradient queries per iteration is constant.*

Proof. Under an exact `Y-Oracle`, Corollary 75 shows the output distribution of `Sample-Joint-Dist` has total variation at most $\frac{\delta}{2}$ from $\hat{\pi}$. Next, the resulting distribution of the subroutine `Y-Oracle` is never larger than $\delta/(2Kd\log(\frac{d\kappa}{\delta}))$ in total variation distance away from an exact sampler. By running for K steps, and using the coupling characterization of total variation, it follows that this can only incur additional error $\delta/(2d\log(\frac{d\kappa}{\delta}))$, proving correctness (in fact, the distribution is always at most $O((d\log(d\kappa/\delta))^{-1})$ away in total variation from an exact `Y-Oracle`).

Next, we prove the guarantee on the expected gradient evaluations per iteration. Lemma 331 shows whenever the current iterate x_k has $\|x - x^*\|_2 \leq \sqrt{\kappa d\log(16\kappa/\delta)} \cdot R_\delta$, the expected number of gradient evaluations is constant, and moreover Proposition 81 shows that the number of gradient evaluations is never larger than $O(d\log(\frac{d\kappa}{\delta}))$, where we use that the condition number of the log-density in (10.24) is bounded by a constant. Therefore, it suffices to show in every iteration $0 \leq k \leq K$, the probability $\|x_k - x^*\|_2 > \sqrt{\kappa d\log(16\kappa/\delta)} \cdot R_\delta$ is $O((d\log(d\kappa/\delta))^{-1})$. By the warmness assumption in Lemma 328, and the concentration bound in Fact 28, the probability x_0 does not satisfy this bound is negligible (inverse exponential in $\kappa d^2\log(\kappa/\delta)$). Since warmness is monotonically decreasing with an exact sampler,[1] and the accumulated error due to inexactness of `Y-Oracle` is at most $O((d\log(d\kappa/\delta))^{-1})$ through the whole algorithm, this holds for all iterations. □

I.3 Mixing time ingredients

We now prove facts which are used in the mixing time analysis of `Sample-Joint-Dist`. Throughout this section, as in the specification of `Sample-Joint-Dist`, f and g are functions with properties as in (10.20), and share a minimizer x^*.

I.3.1 Warm start

We show that we obtain a warm start for the distribution $\hat{\pi}$ in algorithm `Sample-Joint-Dist` via one call to the restricted Gaussian oracle for g, by proving Lemma 328.

Lemma 328 (Warm start)**.** *For $\eta \leq \frac{1}{L\kappa d}$, π_{start} defined in (10.23) is a $2(1+\kappa)^{\frac{d}{2}}$-warm start for $\hat{\pi}$.*

Proof. By the definitions of $\hat{\pi}$ and π_{start} in (10.22), (10.23), we wish to bound everywhere the quantity

$$\frac{d\pi_{\text{start}}}{d\hat{\pi}}(x) = \frac{Z_{\hat{\pi}}}{Z_{\text{start}}} \cdot \frac{\exp\left(-\frac{L}{2}\|x - x^*\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2 - g(x)\right)}{\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\|y - x\|_2^2 - \frac{\eta L^2}{2}\|x - x^*\|_2^2\right) dy}. \tag{I.8}$$

[1] This fact is well-known in the literature, and a simple proof is that if a distribution is warm, then taking one step of the Markov chain induces a convex combination of warm point masses, and is thus also warm.

Here, $Z_{\hat{\pi}}$ is as in Definition 61, and we let Z_{start} denote the normalization constant of π_{start}, i.e.

$$Z_{\text{start}} := \int_x \exp\left(-\frac{L}{2}\left\|x - x^*\right\|_2^2 - \frac{\eta L^2}{2}\left\|x - x^*\right\|_2^2 - g(x)\right) dx.$$

Regarding the first term of (I.8), the earlier derivation (I.4) showed

$$\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\left\|y - x\right\|_2^2 - \frac{\eta L^2}{2}\left\|x - x^*\right\|_2^2\right) dy \leq (2\pi\eta)^{\frac{d}{2}} \exp\left(-f(x) - g(x)\right).$$

Then, integrating, we can bound the ratio of the normalization constants

$$\begin{aligned}
\frac{Z_{\hat{\pi}}}{Z_{\pi_{\text{start}}}} &\leq \frac{\int_x (2\pi\eta)^{\frac{d}{2}} \exp\left(-f(x) - g(x)\right) dx}{\int_x \exp\left(-\frac{L}{2}\left\|x - x^*\right\|_2^2 - \frac{\eta L^2}{2}\left\|x - x^*\right\|_2^2 - g(x)\right) dx} \\
&\leq \frac{\int_x (2\pi\eta)^{\frac{d}{2}} \exp\left(-f(x^*) - \frac{\mu}{2}\left\|x - x^*\right\|_2^2 - g(x)\right) dx}{\int_x \exp\left(-\frac{L}{2}\left\|x - x^*\right\|_2^2 - \frac{\mu}{2}\left\|x - x^*\right\|_2^2 - g(x)\right) dx} \\
&\leq (2\pi\eta)^{\frac{d}{2}} \exp\left(-f(x^*)\right) \left(1 + \frac{L}{\mu}\right)^{\frac{d}{2}}.
\end{aligned} \tag{I.9}$$

The second inequality followed from f is μ-strongly convex and $\eta L^2 \leq \mu$ by assumption. The last inequality followed from Proposition 82, where we used $\frac{\mu}{2}\left\|x - x^*\right\|_2^2 + g(x)$ is μ-strongly convex. Next, to bound the second term of (I.8), notice first that

$$\frac{\exp\left(-\frac{L}{2}\left\|x - x^*\right\|_2^2 - \frac{\eta L^2}{2}\left\|x - x^*\right\|_2^2 - g(x)\right)}{\int_y \exp\left(-f(y) - g(x) - \frac{1}{2\eta}\left\|y - x\right\|_2^2 - \frac{\eta L^2}{2}\left\|x - x^*\right\|_2^2\right) dy} = \frac{\exp\left(-\frac{L}{2}\left\|x - x^*\right\|_2^2\right)}{\int_y \exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x\right\|_2^2\right) dy}.$$

It thus suffices to lower bound $\exp\left(\frac{L}{2}\left\|x - x^*\right\|_2^2\right) \int_y \exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x\right\|_2^2\right) dy$. We have

$$\begin{aligned}
&\exp\left(\frac{L}{2}\left\|x - x^*\right\|_2^2\right) \int_y \exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x\right\|_2^2\right) dy \\
&\geq \exp\left(-f(x) + \frac{L}{2}\left\|x - x^*\right\|_2^2\right) \int_y \exp\left(-\langle \nabla f(x), y - x\rangle - \left(\frac{1}{2\eta} + \frac{L}{2}\right)\left\|y - x\right\|_2^2\right) dy \\
&= \exp\left(-f(x) + \frac{L}{2}\left\|x - x^*\right\|_2^2\right) \left(\frac{2\pi\eta}{1 + L\eta}\right)^{\frac{d}{2}} \exp\left(\frac{\eta}{2(1 + L\eta)}\left\|\nabla f(x)\right\|_2^2\right) \\
&\geq \exp(-f(x^*)) \left(\frac{2\pi\eta}{1 + L\eta}\right)^{\frac{d}{2}}
\end{aligned} \tag{I.10}$$

The first and third steps followed from L-smoothness of f, and the second applied the Gaussian

integral (Fact 27). Combining the bounds in (I.9) and (I.10), (I.8) becomes

$$\frac{d\pi_{\text{start}}}{d\hat{\pi}}(x) \le \left(1 + \frac{L}{\mu}\right)^{\frac{d}{2}} (1 + L\eta)^{\frac{d}{2}} \le 2(1+\kappa)^{\frac{d}{2}},$$

where $x \in \mathbb{R}^d$ was arbitrary, which completes the proof. □

I.3.2 Transitions of nearby points

Here, we prove Lemma 329. Throughout this section, $\mathcal{T}_x$ is the density of x_k, according to the steps in Lines 6 and 7 of `Sample-Joint-Dist` (Algorithm 63) starting at $x_{k-1} = x$. We also define $\mathcal{P}_x$ to be the density of y_k, by just the step in Line 6. We first make a simplifying observation: by Observation 1, for any two points x, x', we have

$$\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\text{TV}} \le \|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\text{TV}}.$$

Thus, it suffices to understand $\|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\text{TV}}$ for nearby $x, x' \in \Omega_\delta$. Our proof of Lemma 329 combines two pieces: (1) bounding the ratio of normalization constants Z_x, $Z_{x'}$ of $\mathcal{P}_x$ and $\mathcal{P}_{x'}$ for nearby x, x' in Lemma 334 and (2) the structural result Proposition 83. To bound the normalization constant ratio, we state two helper lemmas. Lemma 332 characterizes facts about the minimizer of

$$f(y) + \frac{1}{2\eta}\|y - x\|_2^2. \tag{I.11}$$

Lemma 332. *Let f be convex with minimizer x^*, and y_x minimize* (I.11) *for a given x. Then,*

1. $\|y_x - y_{x'}\|_2 \le \|x - x'\|_2$.
2. *For any x,* $\|y_x - x^*\|_2 \le \|x - x^*\|_2$.
3. *For any x with $\|x - x^*\|_2 \le R$,* $\|x - y_x\|_2 \le \eta L R$.

Proof. By optimality conditions in the definition of y_x,

$$\eta \nabla f(y_x) = x - y_x.$$

Fix two points x, x', and let $x_t := (1-t)x + tx'$. Letting $\mathbf{J}_x(y_x)$ be the Jacobian matrix of y_x,

$$\begin{aligned}\frac{d}{dt}\eta\nabla f(y_{x_t}) = \frac{d}{dt}(x_t - y_{x_t}) &\implies \eta\nabla^2 f(y_{x_t})\mathbf{J}_x(y_{x_t})(x' - x) = (\mathbf{I} - \mathbf{J}_x(y_{x_t}))(x' - x) \\ &\implies \mathbf{J}_x(y_{x_t})(x' - x) = (\mathbf{I} + \eta\nabla^2 f(y_{x_t}))^{-1}(x' - x).\end{aligned}$$

We can then compute

$$y_{x'} - y_x = \int_0^1 \frac{d}{dt} y_{x_t} dt = \int_0^1 \mathbf{J}_x(y_{x_t})(x' - x)dt = \int_0^1 (\mathbf{I} + \eta \nabla^2 f(y_{x_t}))^{-1}(x' - x)dt.$$

By triangle inequality and convexity of f, the first claim follows:

$$\|y_{x'} - y_x\|_2 \le \int_0^1 \left\|(\mathbf{I} + \eta \nabla^2 f(y_{x_t}))^{-1}\right\|_2 \|x' - x\|_2 \, dt \le \|x' - x\|_2 .$$

The second claim follows from the first by $y_{x^*} = x^*$. The third claim follows from the second via

$$\|x - y_x\|_2 = \eta \|\nabla f(y_x)\|_2 \le \eta L \|y_x - x^*\|_2 \le \eta L R.$$

□

Next, Lemma 333 states well-known bounds on the integral of a well-conditioned function h.

Lemma 333. *Let h be a L_h-smooth, μ_h-strongly convex function and let y_h^* be its minimizer. Then*

$$\left(2\pi L_h^{-1}\right)^{\frac{d}{2}} \exp\left(-h(y_h^*)\right) \le \int_y \exp\left(-h(y)\right) \le \left(2\pi \mu_h^{-1}\right)^{\frac{d}{2}} \exp\left(-h(y_h^*)\right).$$

Proof. By smoothness and strong convexity,

$$\exp\left(-h(y_h^*) - \frac{L_h}{2}\|y - y_h^*\|_2^2\right) \le \exp(-h(y)) \le \exp\left(-h(y_h^*) - \frac{\mu_h}{2}\|y - y_h^*\|_2^2\right).$$

The result follows by Gaussian integrals, i.e. Fact 27. □

We now define the normalization constants of $\mathcal{P}_x$ and $\mathcal{P}_{x'}$:

$$\begin{aligned} Z_x &= \int_y \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right) dy, \\ Z_{x'} &= \int_y \exp\left(-f(y) - \frac{1}{2\eta}\|y - x'\|_2^2\right) dy. \end{aligned} \tag{I.12}$$

We apply Lemma 332 and Lemma 333 to bound the ratio of Z_x and $Z_{x'}$.

Lemma 334. *Let f be μ-strongly convex and L-smooth. Let $x, x' \in \Omega_\delta$, for Ω_δ defined in* (I.6)*, and let $\|x - x'\|_2 \le \Delta$. Then, the normalization constants Z_x and $Z_{x'}$ in* (I.12) *satisfy*

$$\frac{Z_x}{Z_{x'}} \le 1.05 \exp\left(3LR\Delta + \frac{L\Delta^2}{2}\right).$$

Proof. First, applying Lemma 333 to Z_x and $Z_{x'}$ yields that the ratio is bounded by

$$\begin{aligned}\frac{Z_x}{Z_{x'}} &\le \frac{\exp\left(-f(y_x) - \frac{1}{2\eta}\|y_x - x\|_2^2\right)\left(2\pi\left(\mu + \frac{1}{\eta}\right)^{-1}\right)^{\frac{d}{2}}}{\exp\left(-f(y_{x'}) - \frac{1}{2\eta}\|y_{x'} - x\|_2^2\right)\left(2\pi\left(L + \frac{1}{\eta}\right)^{-1}\right)^{\frac{d}{2}}} \\ &\le 1.05\exp\left(f(y_{x'}) - f(y_x) + \frac{1}{2\eta}\left(\|y_{x'} - x'\|_2^2 - \|y_x - x\|_2^2\right)\right).\end{aligned}$$

Here, we used the bound for $\eta^{-1} \ge 32Ld$ that

$$\left(\frac{L + \frac{1}{\eta}}{\mu + \frac{1}{\eta}}\right)^{d/2} \le 1.05.$$

Regarding the remaining term, recall x, x' both belong to Ω_δ, and $\|x - x'\|_2 \le \Delta$. We have

$$\begin{aligned}&f(y_{x'}) - f(y_x) + \frac{1}{2\eta}\left(\|y_{x'} - x'\|_2^2 - \|y_x - x\|_2^2\right) \\ &\le \langle \nabla f(y_x), y_{x'} - y_x\rangle + \frac{L}{2}\|y_{x'} - y_x\|_2^2 + \frac{1}{2\eta}\langle y_{x'} - x' + y_x - x, y_{x'} - y_x + x - x'\rangle \\ &\le LR\Delta + \frac{L\Delta^2}{2} + \frac{1}{2\eta}\left(\|y_x - x\|_2 + \|y_{x'} - x'\|_2\right)\left(\|y_{x'} - y_x\|_2 + \|x' - x\|_2\right) \\ &\le LR\Delta + \frac{L\Delta^2}{2} + \frac{2\eta LR}{2\eta}\left(\|y_{x'} - y_x\|_2 + \|x' - x\|_2\right) \le 3LR\Delta + \frac{L\Delta^2}{2}.\end{aligned}$$

The first inequality was smoothness and expanding the difference of quadratics. The second was by $\|\nabla f(y_x)\|_2 \le L\|y_x - x^*\|_2 \le LR$ and $\|y_{x'} - y_x\|_2 \le \Delta$, where we used the first and second parts of Lemma 332; we also applied Cauchy-Schwarz and triangle inequality. The third used the third part of Lemma 332. Finally, the last inequality was by the first part of Lemma 332 and $\|x' - x\|_2 \le \Delta$. □

We now are ready to prove Lemma 329.

Lemma 329 (Transitions of nearby points)**.** *Suppose $\eta L \le 1$, $\eta L^2 R_\delta^2 \le \frac{1}{2}$, and $400d^2\eta \le R_\delta^2$. For a point x, let $\mathcal{T}_x$ be the density of x_k after sampling according to Lines 6 and 7 of Algorithm 63 from $x_{k-1} = x$. For $x, x' \in \Omega_\delta$ with $\|x - x'\|_2 \le \frac{\sqrt{\eta}}{10}$, for Ω_δ defined in* (I.6)*, we have $\|\mathcal{T}_x - \mathcal{T}_{x'}\|_{\mathrm{TV}} \le \frac{1}{2}$.*

Proof. First, by Observation 1, it suffices to show $\|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}} \le \frac{1}{2}$. Pinsker's inequality states

$$\|\mathcal{P}_x - \mathcal{P}_{x'}\|_{\mathrm{TV}} \le \sqrt{\frac{1}{2}d_{\mathrm{KL}}\left(\mathcal{P}_x, \mathcal{P}_{x'}\right)},$$

where d_{KL} is KL-divergence, so it is enough to show $d_{\mathrm{KL}}\left(\mathcal{P}_x, \mathcal{P}_{x'}\right) \leq \frac{1}{2}$. Notice that

$$d_{\mathrm{KL}}\left(\mathcal{P}_x, \mathcal{P}_{x'}\right) = \log\left(\frac{Z_{x'}}{Z_x}\right) + \int_y \mathcal{P}_x(y) \log\left(\frac{\exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x\right\|_2^2\right)}{\exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x'\right\|_2^2\right)}\right) dy.$$

By Lemma 334, the first term satisfies, for $\Delta := \frac{\sqrt{\eta}}{10}$,

$$\log\left(\frac{Z_{x'}}{Z_x}\right) \leq 3LR\Delta + \frac{L\Delta^2}{2} + \log(1.05).$$

To bound the second term, we have

$$\begin{aligned}
\int_y \mathcal{P}_x(y) \log\left(\frac{\exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x\right\|_2^2\right)}{\exp\left(-f(y) - \frac{1}{2\eta}\left\|y - x'\right\|_2^2\right)}\right) dy &= \frac{1}{2\eta}\int_y \mathcal{P}_x(y)\left(\left\|y - x'\right\|_2^2 - \left\|y - x\right\|_2^2\right) dy \\
&= \frac{1}{2\eta}\int_y \mathcal{P}_x(y)\left\langle x - x', 2\left(y - x\right) + \left(x - x'\right)\right\rangle dy \\
&\leq \frac{\Delta^2}{2\eta} + \frac{\Delta}{\eta}\left\|\int_y y\mathcal{P}_x(y)dy - x\right\|_2.
\end{aligned}$$

Here, the second line was by expanding and the third line was by $\left\|x - x'\right\|_2 \leq \Delta$ and Cauchy-Schwarz. By Proposition 83, $\left\|\int_y y\mathcal{P}_x(y)dy - x\right\|_2 \leq 2\eta LR$, where by assumption the parameters satisfy the conditions of Proposition 83. Then, combining the two bounds, we have

$$d_{\mathrm{KL}}\left(\mathcal{P}_x, \mathcal{P}_{x'}\right) \leq 3LR\Delta + \frac{L\Delta^2}{2} + \frac{\Delta^2}{2\eta} + 2LR\Delta + \log(1.05) = 5LR\Delta + \frac{L\Delta^2}{2} + \frac{\Delta^2}{2\eta} + \log(1.05).$$

When $\Delta = \frac{\sqrt{\eta}}{10}$, $\eta L \leq 1$, and $\eta L^2 R^2 \leq \frac{1}{2}$, we have the desired

$$d_{\mathrm{KL}}\left(\mathcal{P}_x, \mathcal{P}_{x'}\right) \leq \frac{\sqrt{\eta}LR}{2} + \frac{L\eta}{200} + \frac{1}{200} + \log(1.05) \leq \frac{1}{2}.$$

□

I.3.3 Isoperimetry

In this section, we prove Lemma 330, which asks to show that $\hat{\pi}_{\Omega_\delta}$ satisfies a log-isoperimetric inequality (I.7). Here, we define $\hat{\pi}_{\Omega_\delta}$ to be the conditional distribution of the $\hat{\pi}$ x-marginal on set Ω_δ. We recall this means that for any partition S_1, S_2, S_3 of Ω_δ,

$$\hat{\pi}_{\Omega_\delta}(S_3) \geq \frac{1}{2\psi} d(S_1, S_2) \cdot \min\left(\hat{\pi}_{\Omega_\delta}(S_1), \hat{\pi}_{\Omega_\delta}(S_2)\right) \cdot \sqrt{\log\left(1 + \frac{1}{\min\left(\hat{\pi}_{\Omega_\delta}(S_1), \hat{\pi}_{\Omega_\delta}(S_2)\right)}\right)}.$$

The following fact was shown in [128].

Lemma 335 ([128], Lemma 11)**.** *Any μ-strongly logconcave distribution π satisfies the log-isoperimetric inequality* (I.7) *with $\psi = \mu^{-\frac{1}{2}}$.*

Observe that π_{Ω_δ}, the restriction of π to the convex set Ω_δ, is μ-strongly logconcave by the definition of π (10.20), so it satisfies a log-isoperimetric inequality. We now combine this fact with the relative density bounds Lemma 326 to prove Lemma 330.

Lemma 330 (Isoperimetry)**.** *Density $\hat{\pi}$ and set Ω_δ defined in* (10.22), (I.6) *satisfy* (I.7) *with $\psi = 8\mu^{-\frac{1}{2}}$.*

Proof. Fix some partition S_1, S_2, S_3 of Ω_δ, and without loss of generality let $\hat{\pi}_{\Omega_\delta}(S_1) \le \hat{\pi}_{\Omega_\delta}(S_2)$. First, by applying Corollary 74, which shows $\frac{d\pi}{d\hat{\pi}}(x) \in [\frac{1}{2}, 2]$ everywhere in Ω_δ, we have the bounds

$$\frac{1}{2}\pi_{\Omega_\delta}(S_1) \le \hat{\pi}_{\Omega_\delta}(S_1) \le 2\pi_{\Omega_\delta}(S_1),\ \frac{1}{2}\pi_{\Omega_\delta}(S_2) \le \hat{\pi}_{\Omega_\delta}(S_2) \le 2\pi_{\Omega_\delta}(S_2),\ \text{and } \hat{\pi}_{\Omega_\delta}(S_3) \ge \frac{1}{2}\pi_{\Omega_\delta}(S_3).$$

Therefore, we have the sequence of conclusions

$$\begin{aligned}
\hat{\pi}_{\Omega_\delta}(S_3) &\ge \frac{1}{2}\pi_{\Omega_\delta}(S_3) \\
&\ge \frac{d(S_1,S_2)\sqrt{\mu}}{4} \cdot \min\left(\pi_{\Omega_\delta}(S_1), \pi_{\Omega_\delta}(S_2)\right) \cdot \sqrt{\log\left(1 + \frac{1}{\min\left(\pi_{\Omega_\delta}(S_1), \pi_{\Omega_\delta}(S_2)\right)}\right)} \\
&\ge \frac{d(S_1,S_2)\sqrt{\mu}}{8} \cdot \hat{\pi}_{\Omega_\delta}(S_1) \cdot \sqrt{\log\left(1 + \frac{1}{2\hat{\pi}_{\Omega_\delta}(S_1)}\right)} \\
&\ge \frac{d(S_1,S_2)\sqrt{\mu}}{16} \cdot \hat{\pi}_{\Omega_\delta}(S_1) \cdot \sqrt{\log\left(1 + \frac{1}{\hat{\pi}_{\Omega_\delta}(S_1)}\right)}.
\end{aligned}$$

Here, the second line was by applying Lemma 335 to the μ-strongly logconcave distribution π_{Ω_δ}, and the final line used $\sqrt{\log(1+\alpha)} \le 2\sqrt{\log(1+\frac{\alpha}{2})}$ for all $\alpha > 0$. □

I.3.4 Correctness of `Y-Oracle`

In this section, we show how we can sample y efficiently in the alternating scheme of the algorithm `Sample-Joint-Dist`, within an extremely high probability region. Specifically, for any x with $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$, where R_δ is defined in (I.6), we give a method for implementing

$$\text{draw } y \propto \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right) dy.$$

The algorithm is Algorithm 3, which is a simple rejection sampling scheme.

Algorithm 3 `Y-Oracle`(f, x, η, δ)

Input: L-smooth, μ-strongly convex $f : \mathbb{R}^d \to \mathbb{R}$ with minimizer x^*, $\eta > 0$, $\delta \in [0, 1]$, $x \in \mathbb{R}^d$
Output: If $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$, return exact sample from distribution with density $\propto \exp(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2)$ (see (I.6) for definition of R_δ). Otherwise, return sample within δ TV from distribution with density $\propto \exp(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2)$ **if** $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$ **then**

- **while** ***true*** **do**
  - Draw $y \sim \mathcal{N}(x - \eta\nabla f(x), \eta\mathbf{I})$ $\tau \sim \mathrm{Unif}[0, 1]$ **if** $\tau \le \exp(f(x) + \langle \nabla f(x), y - x\rangle - f(y))$ **then**
    - **Return:** y

Return: Sample x within TV δ from density $\propto \exp(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2)$ using [128]

We recall that we gave guarantees on rejection sampling procedures in Lemma 185 (an "exact" version of Lemma 324 and Corollary 73). We now prove Lemma 331 via a direct application of Lemma 185.

Lemma 331 (`Y-Oracle` guarantee)**.** *For $\delta \in [0, 1]$, define R_δ as in* (I.6)*, and let $\eta = \frac{1}{32L\kappa d \log(16\kappa/\delta)}$. For any x with $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$, Algorithm 3 (`Y-Oracle`) draws an exact sample y from the density proportional to $\exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right)$ in an expected 2 iterations.*

Proof. For $\|x - x^*\|_2 \le \sqrt{\kappa d \log(16\kappa/\delta)} \cdot R_\delta$, `Y-Oracle` is a rejection sampling scheme with

$$p(y) = \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right), \ \hat{p}(y) = \exp\left(-f(x) - \langle \nabla f(x), y - x\rangle - \frac{1}{2\eta}\|y - x\|_2^2\right).$$

It is clear that $p(y) \le \hat{p}(y)$ everywhere by convexity of f, so we may choose $C = 1$. To bound the expected number of iterations and obtain the desired conclusion, Lemma 185 requires a bound on

$$\frac{\int_y \exp\left(-f(x) - \langle \nabla f(x), y - x\rangle - \frac{1}{2\eta}\|y - x\|_2^2\right) dy}{\int_y \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right) dy}, \tag{I.13}$$

the ratio of the normalization constants of $\hat{p}$ and p. First, by Fact 27,

$$\int_y \exp\left(-f(x) - \langle \nabla f(x), y - x\rangle - \frac{1}{2\eta}\|y - x\|_2^2\right) dy = \exp\left(-f(x) + \frac{\eta}{2}\|\nabla f(x)\|_2^2\right)(2\pi\eta)^{\frac{d}{2}}.$$

Next, by smoothness and Fact 27 once more,

$$\begin{aligned}\int_y \exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right) dy &\ge \int_y \exp\left(-f(x) - \langle \nabla f(x), y - x\rangle - \frac{1 + \eta L}{2\eta}\|y - x\|_2^2\right) dy \\ &= \exp\left(-f(x) + \frac{\eta}{2(1 + \eta L)}\|\nabla f(x)\|_2^2\right)\left(\frac{2\pi\eta}{1 + \eta L}\right)^{\frac{d}{2}}.\end{aligned}$$

Taking a ratio, the quantity in (I.13) is bounded above by

$$\exp\left(\left(\frac{\eta}{2} - \frac{\eta}{2(1+\eta L)}\right)\|\nabla f(x)\|_2^2\right)(1+\eta L)^{\frac{d}{2}} \le 1.5\exp\left(\frac{\eta^2 L}{2(1+\eta L)}\|\nabla f(x)\|_2^2\right)$$
$$\le 1.5\exp\left(\frac{\eta^2 L^3}{2}\cdot\left(\frac{16\kappa d^2\log^2(16\kappa/\delta)}{\mu}\right)\right) \le 2.$$

The first inequality was $(1+\eta L)^{\frac{d}{2}} \le 1.5$, the second used smoothness and the assumed bound on $\|x - x^*\|_2$, and the third again used our choice of η. □

I.4 Structural results

Here, we prove two structural results about distributions whose negative log-densities are small perturbations of a quadratic, which obtain tighter concentration guarantees compared to naive bounds on strongly logconcave distributions. They are used in obtaining our bounds in Section I.3 (and for the warm start bounds in Section 10.4), but we hope both the statements and proof techniques are of independent interest to the community. Our first structural result is a bound on normalization constant ratios, used throughout the paper.

Proposition 82. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be μ-strongly convex with minimizer x^*, and let $\lambda > 0$. Then,*

$$\frac{\int \exp(-f(x))dx}{\int \exp\left(-f(x) - \frac{1}{2\lambda}\|x - x^*\|_2^2\right)dx} \le \left(1 + \frac{1}{\mu\lambda}\right)^{\frac{d}{2}}.$$

Proof. Define the function

$$R(\alpha) := \frac{\int \exp\left(-f(x) - \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2\right)dx}{\int \exp\left(-f(x) - \frac{1}{2\lambda}\|x - x^*\|_2^2\right)dx}.$$

Let $d\pi_\alpha(x)$ be the density proportional to $\exp\left(-f(x) - \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2\right)dx$. We compute

$$\frac{d}{d\alpha}R(\alpha) = \int \frac{\exp\left(-f(x) - \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2\right)}{\int \exp\left(-f(x) - \frac{1}{2\lambda}\|x - x^*\|_2^2\right)dx}\frac{1}{2\lambda\alpha^2}\|x - x^*\|_2^2\,dx$$
$$= \frac{R(\alpha)}{2\lambda\alpha^2}\int \frac{\exp\left(-f(x) - \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2\right)\|x - x^*\|_2^2}{\int \exp\left(-f(x) - \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2\right)dx}dx$$
$$= \frac{R(\alpha)}{2\lambda\alpha^2}\int \|x - x^*\|_2^2\,d\pi_\alpha(x) \le \frac{R(\alpha)}{2\alpha}\cdot\frac{d}{\mu\lambda\alpha + 1}.$$

Here, the last inequality was by Fact 30, using the fact that the function $f(x) + \frac{1}{2\lambda\alpha}\|x - x^*\|_2^2$ is

$\mu + \frac{1}{\lambda\alpha}$-strongly convex. Moreover, note that $R(1) = 1$, and

$$\frac{d}{d\alpha}\log\left(\frac{\alpha}{\mu\lambda\alpha+1}\right) = \frac{1}{\alpha} - \frac{\mu\lambda}{\mu\lambda\alpha+1} = \frac{1}{\mu\lambda\alpha^2+\alpha}.$$

Solving the differential inequality

$$\frac{d}{d\alpha}\log(R(\alpha)) = \frac{dR(\alpha)}{d\alpha}\cdot\frac{1}{R(\alpha)} \le \frac{d}{2}\cdot\frac{1}{\mu\lambda\alpha^2+\alpha},$$

we obtain the bound for any $\alpha \ge 1$ (since $\log(R(1)) = 0$)

$$\log(R(\alpha)) \le \frac{d}{2}\log\left(\frac{\mu\lambda\alpha+\alpha}{\mu\lambda\alpha+1}\right) \implies R(\alpha) \le \left(\frac{\mu\lambda\alpha+\alpha}{\mu\lambda\alpha+1}\right)^{\frac{d}{2}} \le \left(1+\frac{1}{\mu\lambda}\right)^{\frac{d}{2}}.$$

Taking a limit $\alpha \to \infty$ yields the conclusion. □

Our second structural result uses a similar proof technique to show that the mean of a bounded perturbation f of a Gaussian is not far from its mode, as long as the gradient of the mode is small. We remark that one may directly apply strong logconcavity, i.e. a variant of Fact 30, to obtain a weaker bound by roughly a $\sqrt{d}$ factor, which would result in a loss of $\Omega(d)$ in the guarantees of Theorem 69. This tighter analysis is crucial in our improved mixing time result.

Before stating the bound, we apply Fact 29 to the convex functions $h(x) = (\theta^\top x)^2$ and $h(x) = \|x\|_2^4$ to obtain the following conclusions which will be used in the proof of Proposition 83.

Corollary 76. *Let π be a μ-strongly logconcave density. Then,*

1. $\mathbb{E}_\pi[(\theta^\top(x - \mathbb{E}_\pi[x]))^2] \le \mu^{-1}$, *for all unit vectors θ.*
2. $\mathbb{E}_\pi[\|x - \mathbb{E}_\pi[x]\|_2^4] \le 3d^2\mu^{-2}$.

Proposition 83. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and convex with minimizer x^*, let $x \in \mathbb{R}^d$ with $\|x - x^*\|_2 \le R$, and let $d\pi_\eta(y)$ be the density proportional to $\exp\left(-f(y) - \frac{1}{2\eta}\|y - x\|_2^2\right)dy$. Suppose that $\eta \le \min\left(\frac{1}{2L^2R^2}, \frac{R^2}{400d^2}\right)$. Then,*

$$\left\|\mathbb{E}_{\pi_\eta}[y] - x\right\|_2 \le 2\eta LR.$$

Proof. Define a family of distributions π^α for $\alpha \in [0, 1]$, with

$$d\pi^\alpha(y) \propto \exp\left(-\alpha\left(f(y) - f(x) - \langle\nabla f(x), y - x\rangle\right) - f(x) - \langle\nabla f(x), y - x\rangle - \frac{1}{2\eta}\|y - x\|_2^2\right)dy.$$

In particular, $\pi^1 = \pi_\eta$, and π^0 is a Gaussian with mean $x - \eta\nabla f(x)$. We define $\bar{y}_\alpha := \mathbb{E}_{\pi_\alpha}[y]$, and

$$y^*_\alpha := \mathrm{argmin}_y \left\{ \alpha \left(f(y) - f(x) - \langle \nabla f(x), y - x \rangle \right) + f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2\eta} \|y - x\|_2^2 \right\}.$$

Define the function $D(\alpha) := \|\bar{y}_\alpha - x\|_2$, such that we wish to bound $D(1)$. First, by smoothness

$$D(0) = \|\mathbb{E}_{\pi_0}[y] - x\|_2 = \|\eta\nabla f(x)\|_2 \leq \eta LR.$$

Next, we observe

$$\frac{d}{d\alpha} D(\alpha) = \left\langle \frac{\bar{y}_\alpha - x}{\|\bar{y}_\alpha - x\|_2}, \frac{d\bar{y}_\alpha}{d\alpha} \right\rangle \leq \left\| \frac{d\bar{y}_\alpha}{d\alpha} \right\|_2.$$

In order to bound $\left\| \frac{d\bar{y}_\alpha}{d\alpha} \right\|_2$, fix a unit vector θ. We have

$$\begin{aligned}
\left\langle \frac{d\bar{y}_\alpha}{d\alpha}, \theta \right\rangle &= \frac{d}{d\alpha} \left\langle \int (y - x) d\pi^\alpha(y), \theta \right\rangle \\
&= \int \langle y - x, \theta \rangle \left(f(x) + \langle \nabla f(x), y - x \rangle - f(y) \right) d\pi^\alpha(y) \\
&\leq \sqrt{\int (\langle y - x, \theta \rangle)^2 d\pi^\alpha(y)} \sqrt{\int (f(x) + \langle \nabla f(x), y - x \rangle - f(y))^2 d\pi^\alpha(y)} \\
&\leq \sqrt{\int (\langle y - x, \theta \rangle)^2 d\pi^\alpha(y)} \sqrt{\int \frac{L^2}{4} \|y - x\|_2^4 d\pi^\alpha(y)}.
\end{aligned} \tag{I.14}$$

The third line was Cauchy-Schwarz and the last line used smoothness and convexity, i.e.

$$-\frac{L}{2} \|y - x\|_2^2 \leq f(x) + \langle \nabla f(x), y - x \rangle - f(y) \leq 0.$$

We now bound these terms. First,

$$\begin{aligned}
\int (\langle y - x, \theta \rangle)^2 d\pi^\alpha(y) &\leq 2 \int (\langle y - \bar{y}_\alpha, \theta \rangle)^2 d\pi^\alpha(y) + 2 \int (\langle \bar{y}_\alpha - x, \theta \rangle)^2 d\pi^\alpha(y) \\
&\leq 2\eta + 2 \|\bar{y}_\alpha - x\|_2^2 = 2\eta + 2D(\alpha)^2.
\end{aligned} \tag{I.15}$$

Here, we applied the first part of Corollary 76, as π^α is η^{-1}-strongly logconcave, and the definition of $D(\alpha)$. Next, using for any $a, b \in \mathbb{R}^d$, $\|a + b\|_2^4 \leq (\|a\|_2 + \|b\|_2)^4 \leq 16 \|a\|_2^4 + 16 \|b\|_2^4$, we have

$$\begin{aligned}
\int \frac{L^2}{4} \|y - x\|_2^4 d\pi^\alpha(y) &\leq \int 4L^2 \|y - \bar{y}_\alpha\|_2^4 d\pi^\alpha(y) + \int 4L^2 \|x - \bar{y}_\alpha\|_2^4 d\pi^\alpha(y) \\
&\leq 12L^2 d^2 \eta^2 + 4L^2 D(\alpha)^4.
\end{aligned} \tag{I.16}$$

Here, we used the second part of Corollary 76. Maximizing (I.14) over θ, and applying (I.15), (I.16),

$$\frac{d}{d\alpha}D(\alpha) \le \left\| \frac{d\bar{y}_\alpha}{d\alpha} \right\|_2 \le \sqrt{8L^2(\eta + D(\alpha)^2)(3d^2\eta^2 + D(\alpha)^4)}$$
$$\le 4L(\sqrt{\eta} + D(\alpha)) \cdot \max(2\eta d, D(\alpha)^2). \tag{I.17}$$

Assume for contradiction that $D(1) > 2\eta LR$, violating the conclusion of the proposition. By continuity of D, there must have been some $\bar{\alpha} \in (0,1)$ where $D(\bar{\alpha}) = 2\eta LR$, and for all $0 \le \alpha < \bar{\alpha}$, $D(\alpha) < 2\eta LR$. By the mean value theorem, there then exists $0 \le \hat{\alpha} \le \bar{\alpha}$ such that

$$\frac{dD(\hat{\alpha})}{d\alpha} = \frac{D(\bar{\alpha}) - D(0)}{\bar{\alpha}} > \eta LR.$$

On the other hand, by our assumption that $2\eta L^2 R^2 \le 1$, for any $d \ge 1$ it follows that

$$2\eta d \ge 4\eta^2 L^2 R^2 > D(\hat{\alpha})^2, \ \sqrt{2\eta} \ge 2\eta LR > D(\hat{\alpha}).$$

Then, plugging these bounds into (I.17) and using $\sqrt{\eta} + D(\hat{\alpha}) \le \frac{5}{2}\sqrt{\eta}$ as $\sqrt{2} \le \frac{3}{2}$,

$$\frac{d}{d\alpha}D(\hat{\alpha}) \le 4L \cdot \frac{5}{2}\sqrt{\eta} \cdot 2\eta d = 20\sqrt{\eta}\frac{d}{R} \cdot \eta LR \le \eta LR.$$

We used $\eta \le \frac{R^2}{400d^2}$ in the last inequality. This is a contradiction, implying $D(1) \le 2\eta LR$. □

I.5 Equivalence of HMC and Metropolis-adjusted Langevin dynamics

We briefly remark on the equivalence of Metropolized HMC and the Metropolis-adjusted Langevin dynamics algorithm (MALA), a well-studied algorithm since its introduction in [76]. This equivalence was also commented on in [128]. The algorithm can be seen as a filtered discretization of the continuous-time Langevin dynamics,

$$dx_t = -\nabla f(x_t)dt + \sqrt{2}dW_t,$$

where W_t is Brownian motion. In short, the Metropolized HMC update is

$$v \sim \mathcal{N}(0, I), \ \tilde{x} \leftarrow x + \eta v - \frac{\eta^2}{2}\nabla f(x), \text{ accept with probability } \min\left\{1, \frac{\exp(-\mathcal{H}(\tilde{x}, \tilde{v}))}{\exp(-\mathcal{H}(x, v))}\right\}.$$

Similarly, the MALA update with step size h is

$$\tilde{x} \sim \mathcal{N}(x - h\nabla f(x), 2hI), \text{ accept with probability } \min\left\{1, \frac{\exp(-f(\tilde{x}) - \|x - \tilde{x} + h\nabla f(\tilde{x})\|_2^2/4h)}{\exp(-f(x) - \|\tilde{x} - x + h\nabla f(x)\|_2^2/4h)}\right\}.$$

It is clear that in HMC the distribution of $\tilde{x}$ is

$$\tilde{x} \sim \mathcal{N}\left(x - \frac{\eta^2}{2}\nabla f(x), \eta^2 I\right),$$

so it suffices to show for $h = \eta^2/2$,

$$\frac{\|\tilde{x} - x + h\nabla f(x)\|_2^2 - \|x - \tilde{x} + h\nabla f(\tilde{x})\|_2^2}{4h} = \frac{1}{2}\left(\|v\|_2^2 - \|\tilde{v}\|_2^2\right).$$

Indeed, the right hand side simplifies to

$$\frac{\eta}{2}\langle \nabla f(\tilde{x}) + \nabla f(x), v\rangle - \frac{\eta^2}{8}\|\nabla f(\tilde{x}) + \nabla f(x)\|_2^2,$$

and the left hand side is

$$\begin{aligned}&\frac{1}{2}\langle \nabla f(\tilde{x}) + \nabla f(x), \tilde{x} - x\rangle + \frac{h}{4}\left(\|\nabla f(x)\|_2^2 - \|\nabla f(\tilde{x})\|_2^2\right)\\ =\ &\frac{1}{2}\left\langle \nabla f(\tilde{x}) + \nabla f(x), \eta v - \frac{\eta^2}{2}\nabla f(x)\right\rangle + \frac{\eta^2}{8}\left(\|\nabla f(x)\|_2^2 - \|\nabla f(\tilde{x})\|_2^2\right).\end{aligned}$$

Comparing coefficients shows the equivalence.

I.6 Gradient concentration

In this section, we give a bound on how well the norm of the gradient $\|\nabla f(x)\|$ concentrates when f is smooth and $x \sim d\pi^*(x)/dx \propto \exp(-f(x))$. First, we recall the following "Hessian-weighted" variant of the Poincaré inequality, which first appeared in [95].

Theorem 94 (Hessian Poincaré)**.** *For probability density $d\pi^*(x)/dx \propto \exp(-f(x))$, and continuously differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ with bounded variance with respect to π^*,*

$$\mathrm{Var}_{\pi^*}[g] \le \int_{\mathbb{R}^d} \left\langle \left(\nabla^2 f(x)\right)^{-1} \nabla g(x), \nabla g(x)\right\rangle d\pi^*(x).$$

An immediate corollary of Theorem 94 is that the Poincaré constant of a μ-strongly logconcave distribution is at most μ^{-1}. While it does not appear to have been previously stated in the literature, our concentration bound can be viewed as a simple application of an argument of Herbst which reduces concentration to an isoperimetric inequality such as Theorem 94; an exposition of this

technique can be found in [341]. We now state the concentration result.

Theorem 95 (Gradient norm concentration)**.** *If twice-differentiable $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and convex, then for $d\pi^*(x)/dx \propto \exp(-f(x))$, and all $c > 0$,*

$$\Pr_{\pi^*}\left[\|\nabla f(x)\| \geq \mathbb{E}_{\pi^*}\left[\|\nabla f\|\right] + c\sqrt{L}\log d\right] \leq 3d^{-c}.$$

Proof. Let $G(x) := \|\nabla f(x)\|$, and let $g(x) := \exp(\frac{1}{2}\lambda G(x))$. Clearly g is continuously differentiable. Moreover, suppose first for simplicity that f is strongly convex; then the existence of the variance of g follows from the well-known fact that f has sub-Gaussian tails (e.g. [210], Lemma 1) and Lipschitzness of its gradient, from which the sublevel sets of the gradient norm grow more slowly than the decay of $\|x - x^*\|_2$. The final conclusion has no dependence on the strong concavity of f, and we can extend this to arbitrary convex functions by regularizing by a small amount of quadratic regularizer (which only affects smoothness) and taking a limit as the regularizer amount vanishes. We now apply Theorem 94, which implies (noting that the gradient of $\|\nabla f\|$ is $\nabla^2 f \frac{\nabla f}{\|\nabla f\|}$)

$$\begin{aligned}\mathbb{E}_{\pi^*}\left[\exp(\lambda G)\right] - \mathbb{E}_{\pi^*}\left[\exp\left(\frac{\lambda G}{2}\right)\right]^2 &\leq \frac{\lambda^2}{4}\mathbb{E}_{\pi^*}\left[\left\langle(\nabla^2 f)\frac{\nabla f}{\|\nabla f\|}, \frac{\nabla f}{\|\nabla f\|}\right\rangle \exp(\lambda G)\right] \\ &\leq \frac{L\lambda^2}{4}\mathbb{E}_{\pi^*}\left[\exp(\lambda G)\right].\end{aligned}$$

In the last inequality we used smoothness. Letting $H(\lambda) := \mathbb{E}_{\pi^*}\left[\exp(\lambda G)\right]$, for $\lambda < \frac{2}{\sqrt{L}}$,

$$H(\lambda) \leq \frac{1}{1 - \frac{L\lambda^2}{4}} H\left(\frac{\lambda}{2}\right)^2.$$

Using this recursively, we have

$$H(\lambda) \leq \prod_{k=0}^{\infty}\left(\frac{1}{1 - \frac{L\lambda^2}{4^{k+1}}}\right)^{2^k} \lim_{\ell\to\infty} H\left(\frac{\lambda}{\ell}\right)^{\ell}.$$

There are two things to estimate on the right hand side. First, for sufficiently large ℓ,

$$\mathbb{E}_{\pi^*}\left[\exp\left(\frac{\lambda G}{\ell}\right)\right]^{\ell} \approx \left(1 + \mathbb{E}_{\pi^*}\left[\frac{\lambda G}{\ell}\right]\right)^{\ell} \approx \exp\left(\lambda \mathbb{E}_{\pi^*}\left[G\right]\right).$$

Second, letting $C = \frac{L\lambda^2}{4} < 1$, [88] showed that

$$\prod_{k=0}^{\infty}\left(\frac{1}{1 - \frac{C}{4^k}}\right)^{2^k} \leq \frac{1 + \sqrt{C}}{1 - \sqrt{C}}.$$

For completeness, we show this in Appendix I.2. Altogether, we have that for all $\lambda < \frac{2}{\sqrt{L}}$,

$$\mathbb{E}_{\pi^*}\left[\exp(\lambda G)\right] \leq \frac{1+\frac{1}{2}\sqrt{L}\lambda}{1-\frac{1}{2}\sqrt{L}\lambda}\exp\left(\lambda \mathbb{E}_{\pi^*}\left[G\right]\right).$$

By Markov's inequality on the exponential, we thus conclude that

$$\Pr_{\pi^*}[G \geq \mathbb{E}_{\pi^*}\left[G\right] + r] \leq \exp(-\lambda r)\frac{1+\frac{1}{2}\sqrt{L}\lambda}{1-\frac{1}{2}\sqrt{L}\lambda}.$$

Finally, letting $\lambda = \frac{1}{\sqrt{L}}$ and $r = c\sqrt{L}\log d$,

$$\Pr_{\pi^*}\left[\|\nabla f\| \geq \sqrt{Ld} + c\sqrt{L}\log d\right] \leq 3d^{-c}.$$

□

As an immediate corollary, we obtain the following.

Corollary 77. *If twice-differentiable $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and strongly convex, then $\forall c > 0$,*

$$\Pr_{\pi^*}\left[\|\nabla f\| \geq \sqrt{Ld} + c\sqrt{L}\log d\right] \leq 3d^{-c}.$$

Proof. It suffices to show that

$$\mathbb{E}_{\pi^*}\left[\|\nabla f\|\right] \leq \sqrt{Ld}. \tag{I.18}$$

This was observed in [152, 521]; we adapt a proof here. Observe that because

$$\nabla \cdot (\nabla f(x)\pi^*(x)) = \Delta f(x)\pi^*(x) - \langle \nabla f(x), \nabla f(x)\rangle \pi(x),$$

where $\nabla\cdot$ is divergence and Δ is the Laplacian operator, integrating both sides and noting that the boundary term vanishes,

$$\mathbb{E}_{\pi^*}\left[\|\nabla f\|^2\right] = \mathbb{E}_{\pi^*}\left[\Delta f\right] \leq Ld.$$

The last equality used smoothness of f. (I.18) then follows from concavity of the square root. □

We remark that for densities $d\pi^*$ where a log-Sobolev variant of the inequality in Theorem 94 holds, we can sharpen the bound in Corollary 77 to $O(d^{-c^2})$.

Definition 63 (Hessian log-Sobolev)**.** *We say density $d\pi^*/dx \propto \exp(-f(x))$ satisfies a* Hessian log-Sobolev inequality *if for all continuously differentiable $g : \mathbb{R}^d \to \mathbb{R}$, and for*

$$\mathrm{Ent}_{\pi^*}\left[g\right] := \left(\int g(x)\log\left(g(x)\right)d\pi^*(x)\right) - \left(\int g(x)d\pi^*(x)\right)\log\left(\int g(x)d\pi^*(x)\right),$$

we have

$$\mathrm{Ent}_{\pi^*}\left[g^2\right] \le 2\int_{\mathbb{R}^d} \left\langle \left(\nabla^2 f(x)\right)^{-1} \nabla g(x), \nabla g(x) \right\rangle d\pi^*(x).$$

In general, this is a much more restrictive condition than Theorem 94; some sufficient conditions are given in [89]. We now show an improved concentration result under a Hessian LSI; the proof follows Herbst's argument, a framework developed in [341].

Theorem 96 (Gradient norm concentration under LSI)**.** *Suppose f is L-smooth and strongly convex, and $d\pi^*(x)/dx \propto \exp(-f(x))$ satisfies a Hessian log-Sobolev inequality. Then for all $c > 0$,*

$$\Pr_{\pi^*}\left[\|\nabla f(x)\| \ge \mathbb{E}_{\pi^*}\left[\|\nabla f\|\right] + c\sqrt{2L\log d}\right] \le d^{-c^2}.$$

Proof. Denote $G := \|\nabla f\|$, where we note $\nabla G = \frac{(\nabla^2 f)\nabla f}{\|\nabla f\|}$. Let $H(\lambda) := \mathbb{E}_{\pi^*}\left[\exp(\lambda G)\right]$, such that $H'(\lambda) = \mathbb{E}_{\pi^*}\left[G\exp(\lambda G)\right]$. Then, for $g^2 = \exp(\lambda G)$,

$$H(\lambda) = \mathbb{E}_{\pi^*}\left[g^2\right],\ \lambda H'(\lambda) = \mathbb{E}_{\pi^*}\left[g^2\log g^2\right].$$

This in turn implies via the LSI that

$$\lambda H'(\lambda) - H(\lambda)\log H(\lambda) = \mathbb{E}_{\pi^*}\left[g^2\log g^2\right] - \mathbb{E}_{\pi^*}\left[g^2\right]\log \mathbb{E}_{\pi^*}\left[g^2\right] \le 2\mathbb{E}_{\pi^*}\left[\|\nabla g\|^2_{(\nabla^2 f)^{-1}}\right]. \quad \text{(I.19)}$$

By smoothness and the definition of $g = \exp(\frac{1}{2}\lambda G)$, we may bound the right hand side:

$$\mathbb{E}_{\pi^*}\left[\|\nabla g\|^2_{(\nabla^2 f)^{-1}}\right] = \frac{\lambda^2}{4}\mathbb{E}_{\pi^*}\left[\|\nabla G\|^2_{(\nabla^2 f)^{-1}}\exp(\lambda G)\right] \le \frac{\lambda^2 L}{4}H(\lambda). \quad \text{(I.20)}$$

In the last inequality we used our calculation of ∇G, and $\nabla^2 f \preceq LI_d$. Now, consider the function $K(\lambda) = \frac{1}{\lambda}\log H(\lambda)$. We handle the definition of $K(0)$ by a limiting argument (and $\log(1+x) \approx x$):

$$K(0) = \lim_{\lambda\to 0}\frac{1}{\lambda}\log \mathbb{E}_{\pi^*}\left[e^{\lambda G}\right] = \frac{H(\lambda) - H(0)}{\lambda} = H'(0) = \mathbb{E}_{\pi^*}\left[G\right].$$

We compute

$$K'(\lambda) = -\frac{1}{\lambda^2}\log H(\lambda) + \frac{H'(\lambda)}{\lambda H(\lambda)} = \frac{\lambda H'(\lambda) - H(\lambda)\log H(\lambda)}{\lambda^2 H(\lambda)}.$$

This, combined with (I.19) and (I.20) imply $K'(\lambda) \le \frac{L}{2}$. Therefore, by integrating, we have

$$K(\lambda) \le \mathbb{E}_{\pi^*}\left[G\right] + \frac{L\lambda}{2} \Rightarrow H(\lambda) = \exp(\lambda K(\lambda)) \le \exp\left(\lambda\mathbb{E}_{\pi^*}\left[G\right] + \frac{L\lambda^2}{2}\right).$$

Finally, we have concentration:

$$\Pr_{\pi^*}[G \geq \mathbb{E}_{\pi^*}[G] + r] = \Pr_{\pi^*}[\exp(\lambda G) \geq \exp(\lambda \mathbb{E}_{\pi^*}[G] + \lambda r)] \leq \exp\left(-\lambda r + \frac{L\lambda^2}{2}\right),$$

where the last statement is by Markov. Choosing $r = c\sqrt{2L \log d}$, $\lambda = r/L$ yields the conclusion. □

This sharpening is desirable for reasons related to the warmness of starting distributions for sampling from π^*. However, the "Hessian log-Sobolev" inequality is strictly stronger than Theorem 94, and does not hold for general strongly logconcave distributions [89]. Correspondingly, the concentration arguments derivable from Poincaré inequalities appear to be weaker [341]: we find exploring the tightness of Corollary 77 to be an interesting open question.

I.7 Necessity of fixing a scale

We give a simple argument showing if the step size η of the HMC algorithm does not depend on the "scale" of the problem, namely the eigenvalues of the function Hessian (as opposed to scale-invariant quantities, e.g. the condition number κ and the dimension), then the task of proving lower bounds becomes much more trivial. In particular, we can adaptively pick a scale of the problem in response to the fixed η. This justifies the additional requirement in Theorems 71, 72, 73 and 74 of the fixed scale $[1, \kappa]$, which we remark is a *strengthening* of an analogous scale-free lower bound.

Concretely, suppose we wished to prove the statement of Theorem 74 but only on functions with condition number κ (without specifying a range of eigenvalues). Then, for fixed η, K, consider

$$f(x) = \frac{\lambda}{2}x^2, \text{ where } \lambda := \frac{2\left(1 - \cos\left(\frac{\pi}{K}\right)\right)}{\eta^2}.$$

Clearly, $f : \mathbb{R} \to \mathbb{R}$ has condition number $1 \leq \kappa$ for any κ. Then, the proof of Proposition 52 applies to show that the HMC Markov chain cannot leave any symmetric set, because the coefficients encounter extremal points or zeroes of the Chebyshev polynomials.

I.8 HMC lower bounds beyond $\kappa\sqrt{d}$

Here, we analyze the behavior of HMC on the hard function (10.44). We will use this construction to demonstrate that when the number of steps K is small, we cannot improve either the relaxation time (Section I.8.1) or the mixing time (Section I.8.2) of MALA by more than roughly a $O(K)$ factor.

I.8.1 Relaxation time lower bound for small K

We first give a bound on the acceptance probability (10.14) for general HMC Markov chain. We expand the term $-\mathcal{H}(x_K, v_K) + \mathcal{H}(x_0, v_0)$ and extend the result given by Lemma 203.

Lemma 336. *For the iterates given by Fact 34, write $\tilde{x}_j := x_0 + \eta j v_0$ for $0 \le j \le K-1$. Then, for a κ-smooth function f,*

$$-\mathcal{H}(x_K, v_K) + \mathcal{H}(x_0, v_0) \le \sum_{j=0}^{K-1} \left(-f(\tilde{x}_{j+1}) + f(\tilde{x}_j) + \frac{1}{2} \left\langle \eta v_0, \nabla f(\tilde{x}_{j+1}) + \nabla f(\tilde{x}_j) \right\rangle \right)$$
$$+\eta K \left\| v_0 \right\|_2 \max_{0 \le j \le K} \left\| \nabla f(x_j) - \nabla f(\tilde{x}_j) \right\|_2 + \frac{1}{2} \eta^2 K^2 \max_{0 \le j_1, j_2 \le K} \left\| \nabla f(\tilde{x}_K) - \nabla f(x_{j_2}) \right\|_2 \left\| \nabla f(x_{j_1}) \right\|_2$$
$$+\frac{1}{2} \eta^2 K^2 \max_{0 \le j_1, j_2, j_3 \le K} \left\| \nabla f(x_{j_3}) \right\|_2 \left\| \nabla f(x_{j_1}) - \nabla f(x_{j_2}) \right\|_2 .$$

Proof. Expanding $\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K)$ according to the definition of $\mathcal{H}$, x_K and v_K,

$$
\begin{aligned}
&\mathcal{H}(x_0, v_0) - \mathcal{H}(x_K, v_K) \\
=& -f(x_K) + f(x_0) - \frac{\left\| v_0 - \frac{\eta}{2}\nabla f(x_0) - \eta \sum_{j=1}^{K-1} \nabla f(x_j) - \frac{\eta}{2}\nabla f(x_K) \right\|_2^2}{2} + \frac{\|v_0\|_2^2}{2} \\
=& -f(x_K) + f(\tilde{x}_K) - f(\tilde{x}_K) + f(x_0) + \left\langle v_0, \frac{\eta}{2}\nabla f(x_0) + \eta \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{\eta}{2}\nabla f(x_K) \right\rangle \\
&- \frac{1}{2}\left\| \frac{\eta}{2}\nabla f(x_0) + \eta \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{\eta}{2}\nabla f(x_K) \right\|_2^2 \\
=& -f(x_K) + f(\tilde{x}_K) + \sum_{j=0}^{K-1} \left(-f(\tilde{x}_{j+1}) + f(\tilde{x}_j)\right) + \left\langle \eta v_0, \frac{1}{2}\nabla f(\tilde{x}_0) + \sum_{j=1}^{K-1} \nabla f(\tilde{x}_j) + \frac{1}{2}\nabla f(\tilde{x}_K) \right\rangle \\
&- \frac{1}{2}\left\| \frac{\eta}{2}\nabla f(x_0) + \eta \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{\eta}{2}\nabla f(x_K) \right\|_2^2 \\
&+ \left\langle \eta v_0, \left(\frac{1}{2}\nabla f(x_0) + \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{1}{2}\nabla f(x_K) \right) - \left(\frac{1}{2}\nabla f(\tilde{x}_0) + \sum_{j=1}^{K-1} \nabla f(\tilde{x}_j) + \frac{1}{2}\nabla f(\tilde{x}_K) \right) \right\rangle \\
=& \sum_{j=0}^{K-1} \left(-f(\tilde{x}_{j+1}) + f(\tilde{x}_j) + \frac{1}{2}\left\langle \eta v_0, \nabla f(\tilde{x}_{j+1}) + \nabla f(\tilde{x}_j) \right\rangle \right) \\
&- f(x_K) + f(\tilde{x}_K) - \frac{1}{2}\left\| \frac{\eta}{2}\nabla f(x_0) + \eta \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{\eta}{2}\nabla f(x_K) \right\|_2^2 \\
&+ \left\langle \eta v_0, \left(\frac{1}{2}\nabla f(x_0) + \sum_{j=1}^{K-1} \nabla f(x_j) + \frac{1}{2}\nabla f(x_K) \right) - \left(\frac{1}{2}\nabla f(\tilde{x}_0) + \sum_{j=1}^{K-1} \nabla f(\tilde{x}_j) + \frac{1}{2}\nabla f(\tilde{x}_K) \right) \right\rangle .
\end{aligned}
\tag{I.21}
$$

Now we bound the last two lines in the decomposition (I.21). For the second-to-last line of (I.21),

by convexity of f and the Cauchy-Schwarz inequality,

$$\begin{aligned}
&- f(x_K) + f(\tilde{x}_K) - \frac{1}{2}\left\|\frac{\eta}{2}\nabla f(x_0) + \eta\sum_{j=1}^{K-1}\nabla f(x_j) - \frac{\eta}{2}\nabla f(x_K)\right\|_2^2 \\
\leq &\left\langle \nabla f(\tilde{x}_K), \frac{1}{2}K\eta^2\nabla f(x_0) + \eta^2\sum_{j=1}^{K-1}(K-j)\nabla f(x_j)\right\rangle - \frac{1}{2}\left\|\frac{\eta}{2}\nabla f(x_0) + \eta\sum_{j=1}^{K-1}\nabla f(x_j) + \frac{\eta}{2}\nabla f(x_K)\right\|_2^2 \\
\leq &\frac{1}{2}\eta^2K^2\max_{0\leq j_1,j_2,j_3\leq K}\left(\nabla f(\tilde{x}_K)^\top\nabla f(x_{j_1}) - \nabla f(x_{j_2})^\top\nabla f(x_{j_3})\right) \\
\leq &\frac{1}{2}\eta^2K^2\left(\max_{0\leq j_1,j_2\leq K}\|\nabla f(\tilde{x}_K) - \nabla f(x_{j_2})\|_2\,\|\nabla f(x_{j_1})\|_2 + \max_{0\leq j_1,j_2,j_3\leq K}\|\nabla f(x_{j_3})\|_2\,\|\nabla f(x_{j_1}) - \nabla f(x_{j_2})\|_2\right).
\end{aligned} \tag{I.22}$$

In the third line above, we used that the total "number of gradient inner products" for both terms is $\frac{1}{2}\eta^2K^2$, and took the largest such inner product difference.

Finally, for the last line of (I.21), by the Cauchy-Schwarz inequality,

$$\begin{aligned}
&\left\langle \eta v_0, \left(\frac{1}{2}\nabla f(x_0) + \sum_{j=1}^{K-1}\nabla f(x_j) + \frac{1}{2}\nabla f(x_K)\right) - \left(\frac{1}{2}\nabla f(\tilde{x}_0) + \sum_{j=1}^{K-1}\nabla f(\tilde{x}_j) + \frac{1}{2}\nabla f(\tilde{x}_K)\right)\right\rangle \\
\leq &\eta K\|v_0\|_2\max_{0\leq j\leq K}\|\nabla f(x_j) - \nabla f(\tilde{x}_j)\|_2.
\end{aligned} \tag{I.23}$$

Combining (I.21), (I.22) and (I.23) proves the desired claim. □

We define a hard function $f_{\text{hard}} : \mathbb{R}^d \to \mathbb{R}$ that is κ-smooth and 1-strongly convex (note it is the same hard function as in Section 10.10, under the change of variable $h = \frac{\eta^2}{2}$). We will show it is hard to sample from the density proportional to $\exp(-f_{\text{hard}})$ when K is small.

$$f_{\text{hard}}(x) := \sum_{i\in[d]} f_i(x_i), \text{ where } f_i(c) = \begin{cases} \frac{1}{2}c^2 & i = 1 \\ \frac{\kappa}{3}c^2 - \frac{\kappa\eta^2}{6}\cos\left(\frac{\sqrt{2}c}{\eta}\right) & 2 \leq i \leq d \end{cases}. \tag{I.24}$$

Lemma 337. *For $\eta^2 \leq 1$, let $\tilde{x}_j := x_0 + \eta j v_0$ for $0 \leq j \leq K-1$ and $v_0 \sim \mathcal{N}(0, \mathbf{I})$. Let $R^{(j)}$ be the random variable with given by $R^{(j)} = \sum_{i=1}^d R_i^{(j)}$ where*

$$R_i^{(j)} = -f_i([\tilde{x}_{j+1}]_i) + f_i([\tilde{x}_j]_i) + \frac{1}{2}\eta[v_0]_i\cdot(\nabla f_i([\tilde{x}_{j+1}]_i) + \nabla f_i([\tilde{x}_j]_i)).$$

Then,

$$\mathbb{E}_{v_0\sim\mathcal{N}(0,1)}\left[\sum_{j=0}^{K-1} R^{(j)}\right] \leq -0.02\kappa\eta^2\sum_{i=2}^{d}\cos\frac{\sqrt{2}[x_0]_i}{\eta}. \tag{I.25}$$

and

$$\Pr\left[\sum_{j=0}^{K-1} R^{(j)} - \mathbb{E}\left[\sum_{j=0}^{K-1} R^{(j)}\right] \geq 10\eta^2 K\kappa\sqrt{d\log d}\right] \leq \frac{1}{d^5}. \tag{I.26}$$

Proof. In this proof, all expectations $\mathbb{E}$ are taken over $v_0 \sim \mathcal{N}(0, \mathbf{I})$, so we omit them. For $i = 1$,

$$\begin{aligned}\mathbb{E}\left[\sum_{j=0}^{K-1} R_i^{(j)}\right] &= \mathbb{E}\left[-\frac{1}{2}([x_0]_1 + \eta K[v_0]_1)^2 + \frac{1}{2}[x_0]_1^2 + \frac{1}{2}\sum_{j=0}^{K-1}\eta[v_0]_1(2[x_0]_1 + \eta(2j+1)[v_0]_1)\right] \\ &= \mathbb{E}\left[-\frac{1}{2}[x_0]_1^2 - \frac{1}{2}\eta^2K^2[v_0]_1^2 - \eta K[x_0]_1[v_0]_1 + \frac{1}{2}[x_0]_1^2 + \frac{1}{2}\eta^2K^2[v_0]_1^2 + \eta K[x_0]_1[v_0]_1\right] = 0.\end{aligned}$$

We bound each coordinate $2 \leq i \leq d$ separately.

$$\begin{aligned}&\mathbb{E}\left[\sum_{j=0}^{K-1} R_i^{(j)}\right] \\ =&\mathbb{E}\left[\sum_{j=0}^{K-1} -f_i([\tilde{x}_{j+1}]_i) + f_i([\tilde{x}_j]_i) + \frac{1}{2}\eta[v_0]_i \cdot (\nabla f_i([\tilde{x}_{j+1}]_i) + \nabla f_i([\tilde{x}_j]_i))\right] \\ =& -\frac{\kappa}{3}\mathbb{E}\left[([x_0]_i + \eta K[v_0]_i)^2 - [x_0]_i^2\right] + \frac{1}{3}\eta\kappa\mathbb{E}\left[[v_0]_i \cdot \left(2[x_0]_i + \eta\sum_{j=0}^{K-1}(2j+1)[v_0]_i\right)\right] \\ &+\frac{\kappa\eta^2}{6}\mathbb{E}\left[\sum_{j=0}^{K-1}\cos\frac{\sqrt{2}\left([x_0]_i + \eta(j+1)[v_0]_i\right)}{\eta} - \cos\frac{\sqrt{2}\left([x_0]_i + \eta j[v_0]_i\right)}{\eta}\right] \\ &+\frac{\sqrt{2}\eta^2\kappa}{12}\mathbb{E}\left[[v_0]_i\sum_{j=0}^{K-1}\left(\sin\frac{\sqrt{2}\left([x_0]_i + \eta j[v_0]_i\right)}{\eta} + \sin\frac{\sqrt{2}\left([x_0]_i + \eta(j+1)[v_0]_i\right)}{\eta}\right)\right] \\ =& -\frac{\kappa\eta^2}{6}\sum_{j=0}^{K-1}\exp(-j^2) - \exp(-(j+1)^2) - j\exp(-j^2) - (j+1)\exp(-(j+1)^2)\cos\frac{\sqrt{2}[x_0]_i}{\eta}\end{aligned}$$

The last line used the computation

$$\mathbb{E}\left[[v_0]_i \sin\frac{\sqrt{2}\left([x_0]_i + \eta j[v_0]_i\right)}{\eta}\right] = \sqrt{2}j\exp(-j^2)\cos\frac{\sqrt{2}[x_0]_i}{\eta},$$
$$\mathbb{E}\left[\cos\frac{\sqrt{2}\left([x_0]_i + \eta j[v_0]_i\right)}{\eta}\right] = \exp(-j^2)\cos\frac{\sqrt{2}[x_0]_i}{\eta}.$$

Next, we bound $\sum_{j=0}^{K-1}\left(\exp(-j^2) - \exp(-(j+1)^2) - j\exp(-j^2) - (j+1)\exp(-(j+1)^2)\right)$. For $j = 0$, $1 - \frac{2}{\exp(1)} \geq 0.264$. For $j = 1$, the negative terms have $-3\exp(-4) \geq -0.06$, and the positive

terms can only help this inequality. For the remaining terms,

$$\sum_{j=2}^{K-1} \left(\exp(-j^2) - \exp(-(j+1)^2) - j\exp(-j^2) - (j+1)\exp(-(j+1)^2)\right)$$
$$\geq \sum_{j=2}^{K-1} \left(-j\exp(-j^2) - (j+1)\exp(-(j+1)^2)\right)$$
$$\geq -2\sum_{j=2}^{K} \left(j\exp(-j^2)\right) \geq -2\frac{2}{\exp(4)}\frac{1}{1-2\exp(-5)} \geq -0.075.$$

The last inequality used the ratio between two consecutive terms is bounded by $\frac{j+1}{j}\exp(j^2-(j+1)^2) \leq 2\exp(-5)$. Summing over d coordinates proves (I.25).

Next, we prove the concentration property of $\sum_{j=0}^{K-1} R^{(j)}$. Let $\tilde{x}_{j,s} = \tilde{x}_j + s\eta v_0$, for $s \in [0,1]$ and $j = 0, ..., K-1$. By Lemma 197, we have

$$\sum_{j=0}^{K-1} R^{(j)} = \sum_{j=0}^{K-1} -\eta^2 \int_0^1 \left(\frac{1}{2} - s\right) v_0^\top \nabla^2 f(\tilde{x}_{j,s}) v_0 ds.$$

For coordinate $1 \leq i \leq d$, $\left|\eta^2 \int_0^1 \left(\frac{1}{2}-s\right) f_i''([x_{j,s}]_i)ds\right| \leq \frac{\eta^2\kappa}{2}$ by smoothness. Then, the random variables $\sum_{j=0}^{K-1} R_i^{(j)} - \mathbb{E}\left[\sum_{j=0}^{K-1} R_i^{(j)}\right]$ for $1 \leq i \leq d$ are sub-exponential with parameter $\frac{\eta^2\kappa K}{2}$(for coordinates where the coefficient is negative, note the negation of a sub-exponential random variable is still sub-exponential). Hence, by Fact 33,

$$\Pr\left[\sum_{i\in[d]} \left(\sum_{k=0}^{K-1} R_i^{(j)} - \mathbb{E}\left[\sum_{k=0}^{K-1} R_i^{(j)}\right]\right) \geq 10\eta^2 K\kappa\sqrt{d\log d}\right] \leq \frac{1}{d^5}.$$

□

Now, we build a bad set Ω_{hard} with lower bounded measure that starting from a point $x_0 \in \Omega_{\text{hard}}$, such that with high probability, $-\mathbb{E}\left[\sum_{j=0}^{K-1} R^{(j)}\right]$ is very negative. Let $h = \frac{1}{2}\eta^2$ so that we may use the results from Section 10.8. We use the bad set Ω_{hard} defined in (10.45).

$$\Omega_{\text{hard}} = \Bigg\{x \;\Bigg|\; |x_1| \leq 2, \forall 2 \leq i \leq d, \exists k_i \in \mathbb{Z}, |k_i| \leq \lfloor\frac{5}{\pi\sqrt{h\kappa}}\rfloor, \text{ such that}$$
$$-\frac{9}{20}\pi\sqrt{h} + 2\pi k_i\sqrt{h} \leq x_i \leq \frac{9}{20}\pi\sqrt{h} + 2\pi k_i\sqrt{h}\Bigg\}.$$

We restate Lemma 200 here, which lower bounds $\pi^*(\Omega_{\text{hard}})$ and bounds $\|\nabla f(x)\|_2$ for $x \in \Omega_{\text{hard}}$.

Lemma 200. *Let $h \leq \frac{1}{10000\pi^2\kappa}$. Let π^* have log-density $-f_{\text{hard}}$ (10.44). Then, $\pi^*(\Omega_{\text{hard}}) \geq$*

$\exp(-d)$. *Moreover, for all* $x \in \Omega_{\text{hard}}$, $\|\nabla f(x)\|_2 \le 10\sqrt{\kappa d}$.

We can further show the following, which is used to bound the remaining terms in Lemma 336.

Lemma 338. *Let* $x_0 \in \Omega_{\text{hard}}$, $\eta K \le \frac{1}{100\sqrt{\kappa}\log d}$ *and* $d \ge 8$. *Let let* x_j *for* $1 \le j \le K-1$ *be given by the iterates in Fact 34 and* $\tilde{x}_K = x_0 + \eta K v_0$. *Then, with probability at least* $1 - \frac{1}{d^5}$ *over random* $v_0 \sim \mathcal{N}(0, \mathbf{I})$, $\|v_0\|_2 \le 4\sqrt{d}\log d$ *and for all* $0 \le j \le K$, $\|\nabla f(x_j)\|_2 \le 11\sqrt{\kappa d}$ *and* $\|\nabla f(\tilde{x}_K)\|_2 \le 11\sqrt{\kappa d}$.

Proof. We first derive a bound on $v_0 \sim \mathcal{N}(0, \mathbf{I})$. By a standard Gaussian tail bound, for $d \ge 8$, with probability at least $1 - \frac{1}{d^5}$, $|[v_0]_i| \le 4\log d$ for all $1 \le i \le d$. Then, $\|v_0\|_2 \le \sqrt{16d(\log d)^2} = 4\sqrt{d}\log d$. Now, we prove the bound on $\|x_j - x_0\|_2$ and $\|\nabla f(x_j)\|_2$ using induction. First, $\|\nabla f(x_0)\| \le 11\sqrt{d\kappa}$ holds by Lemma 200. Assume for induction $\|\nabla f(x_k)\|_2 \le 11\sqrt{d\kappa}$ for $1 \le k < j$. Then,

$$\begin{aligned}\|x_j - x_0\|_2 &\le \left\|\eta j v_0 - \frac{\eta^2 j}{2}\nabla f(x_0) - \eta^2 \sum_{k=1}^{j-1}(j-k)\nabla f(x_k)\right\|_2 \\ &\le 4\eta j\sqrt{d}\log d + \eta^2 j^2 \cdot 11\sqrt{\kappa d} \le \sqrt{\frac{d}{\kappa}}.\end{aligned}$$

The last inequality used the assumption $\eta K \le \frac{1}{100\sqrt{\kappa}\log d}$. Since f is κ-smooth, we have

$$\|\nabla f(x_j)\|_2 \le \|\nabla f(x_0)\|_2 + \kappa\|x_j - x_0\|_2 \le 10\sqrt{\kappa d} + \kappa\sqrt{\frac{d}{\kappa}} \le 11\sqrt{\kappa d}.$$

This completes the induction step. Finally, we have

$$\|\nabla f(\tilde{x}_K)\|_2 \le \|\nabla f(x_0)\|_2 + \kappa\|\eta K v_0\|_2 \le 10\sqrt{\kappa d} + 4\eta K\kappa\sqrt{d}\log d \le 11\sqrt{\kappa d},$$

where we used $\eta K \le \frac{1}{100\sqrt{\kappa}\log d}$. □

Lemma 339. *Let* η *and* K *satisfy* $K \le \frac{\sqrt{d}}{10000\sqrt{\log d}}$, *and* $\eta K^3 \le \frac{1}{100000\sqrt{\kappa}\log d}$. *For any* $x_0 \in \Omega_{\text{hard}}$, *let* (x_K, v_K) *be given by the iterates in Fact 34 and* $v_0 \sim \mathcal{N}(0, \mathbf{I})$. *With probability at least* $1 - \frac{2}{d^5}$,

$$-\mathcal{H}(x_K, v_K) + \mathcal{H}(x_0, v_0) \le -\Omega\left(\eta^2\kappa d\right).$$

Proof. We first remark that the bound on ηK^3 implies we may apply Lemma 200 and Lemma 338. Next, for $x_0 \in \Omega_{\text{hard}}$, $\cos\frac{\sqrt{2}[x_0]_i}{\eta}$ is bounded away from 0 for all $2 \le i \le d$. By Lemma 337, when $K \le \frac{\sqrt{d}}{10000\sqrt{\log d}}$, with probability at least $1 - \frac{1}{d^5}$, $\sum_{j=0}^{K-1} R^{(j)} \le -0.002\eta^2\kappa d$ (the expectation term

dominates). By Lemma 338, with probability at least $1 - \frac{1}{d^5}$, the other terms in Lemma 336 have

$$\begin{aligned}\eta K \left\|v_0\right\|_2 \max_{0\le j\le K} \left\|\nabla f(x_j) - \nabla f(\tilde{x}_j)\right\|_2 + \frac{1}{2}\eta^2 K^2 \max_{0\le j_1,j_2\le K} \left\|\nabla f(\tilde{x}_K) - \nabla f(x_{j_2})\right\|_2 \left\|\nabla f(x_{j_1})\right\|_2 \\ + \frac{1}{2}\eta^2 K^2 \max_{0\le j_1,j_2,j_3\le K} \left\|\nabla f(x_{j_3})\right\|_2 \left\|\nabla f(x_{j_1}) - \nabla f(x_{j_2})\right\|_2 \\ \le 4\eta K\sqrt{d}\log d\cdot \kappa\eta^2 \left(K\left\|\nabla f(x_0)\right\|_2 + \sum_{j\in[K-1]} (K-j)\left\|\nabla f(x_j)\right\|_2\right) \\ + \eta^2K^2\cdot 11\sqrt{\kappa d}\cdot\kappa\left(\eta K\left\|v_0\right\|_2 + \eta^2 K\left\|\nabla f(x_0)\right\|_2 + \eta^2\sum_{j\in[K-1]}(K-j)\left\|\nabla f(x_j)\right\|_2\right) \\ \le 44\eta^3K^3\kappa^{1.5}d\log d + 44\eta^3K^3\kappa^{1.5}d\log d + 121\eta^4K^4\kappa^2 d \le 0.001\eta^2\kappa d.\end{aligned}$$

The last inequality used the assumption $\eta \le \frac{1}{100000K^3\sqrt{\kappa}\log d}$. Combining the above bounds with Lemma 336 yields the claim. □

Proposition 84. *For $\eta^2K = O\left(\frac{\sqrt{\log d}}{\kappa\sqrt{d}}\right)$ and $K = O\left(d^{0.099}\right)$, there is a target density on $\mathbb{R}^d$ whose negative log-density is κ smooth, such that relaxation time of HMC is $\Omega\left(\frac{\kappa d}{K^2}\right)$.*

Proof. It is straightforward to check that such a range of η and K satisfies the assumptions of Lemma 339. Applying Lemma 339 with the hard function f_{hard}, the remainder of the proof follows analogously to that of Theorem 74. □

We give a brief discussion of the implications of Proposition 84. For $\eta^2K = \omega(\frac{\sqrt{\log d}}{\kappa\sqrt{d}})$, the proof of Theorem 74 rules out a polynomial relaxation time. In the remaining range, Proposition 84 implies that for small $K = O\left(d^{0.099}\right)$, the most we can improve the relaxation time of MALA (Theorem 72) by taking multiple steps in HMC is by a K^2 factor. Since each iteration takes K gradients, this is roughly an improvement of K in the query complexity, and strengthens Theorem 74 for small K.

I.8.2 Mixing time lower bound for small K

In this section, we first use prior results to narrow down the range of η we consider (assuming K is small). We then generalize the ideas of Section 10.9, our MALA mixing lower bound, to this setting.

Mixing time lower bound for large η. Suppose $K = O\left(d^{0.099}\right)$ throughout this section. The arguments of Section 10.10, specifically Proposition 52 and Lemma 209, imply mixing time lower bounds for all $\eta K = \Omega(\frac{1}{\sqrt{\kappa}})$ (using the "boosting constants" argument of Section 10.10.2 for sufficiently large κ as necessary). For $\eta K = O(\frac{1}{\sqrt{\kappa}})$, the proof of Theorem 74 further implies mixing time lower bounds for all $\eta^2K = \omega(\frac{\sqrt{\log d}}{\kappa\sqrt{d}})$. Hence, we can assume $\eta K = O(\frac{1}{\sqrt{\kappa}})$ and $\eta^2K = O(\frac{\sqrt{\log d}}{\kappa\sqrt{d}})$.

Next, under the further assumption that $K = O\left(d^{0.099}\right)$, it is easy to check under the specified assumptions on η and K, the preconditions of Lemma 339 are met. This implies that we can rule

out $\eta^2 = \omega(\frac{\log d}{\kappa d})$ for polynomial-time mixing. Thus, in the following discussion we assume

$$K = O\left(d^{0.099}\right),\ \eta^2 = O\left(\frac{\log d}{\kappa d}\right). \tag{I.27}$$

Mixing time lower bound for small η. Let $\pi^* = \mathcal{N}(0, \mathbf{I})$ be the standard d-dimensional multivariate Gaussian. We will let π_0 be the marginal distribution of π^* on the set

$$\Omega := \left\{x \mid \|x\|_2^2 \leq \frac{1}{2}d\right\}.$$

Recall from Lemma 195 that π_0 is a $\exp(d)$-warm start. Our main proof strategy will be to show that for small η and K as in (I.27), after $T = O(\frac{\kappa d}{K^2 \log^3 d})$ iterations, with constant probability both of the following events happen: no rejections occur throughout the Markov chain, and $\|x_{t,K}\|_2^2 \leq \frac{9}{10}d$ holds for all $t \in [T]$. Combining these two facts will demonstrate our total variation lower bound.

Lemma 340. *Let $\{x_{t,k}, v_{t,k}\}_{0\leq t<T, 0\leq k\leq K}$ be the sub-iterates generated by the HMC Markov chain with step size $\eta^2 = O\left(\frac{\log d}{\kappa d}\right)$ and $\eta^2 K^2 \leq 1$, for $T = O(\frac{\kappa d}{K^2 \log^3 d})$ and $x_0 \sim \pi_0$; we denote the actual HMC iterates by $\{x_t\}_{0\leq t<T}$. With probability at least $\frac{99}{100}$, both of the following events occur:*

1. *Throughout the Markov chain, $\|x_t\|_2 \leq 0.9\sqrt{d}$.*
2. *Throughout the Markov chain, the Metropolis filter never rejected.*

Proof. Let $h = \frac{1}{2}\eta^2$. We inductively bound the failure probability of the above events in every iteration by $\frac{0.01}{T}$, which will yield the claim via a union bound. Take some iteration $t+1$, and note that by triangle inequality, and assuming all prior iterations did not reject,

$$\|x_{t+1,K}\|_2 \leq \|x_{0,0}\|_2 + \eta K \left\|\sum_{s=0}^{t} v_{s,0}\right\| + \eta^2 K \sum_{s=0}^{t}\sum_{k=1}^{K} \|x_{s,k}\|_2 \leq \|x_{0,0}\|_2 + 0.9\eta^2 K^2 T\sqrt{d} + \eta K \|G_t\|_2$$
$$\leq 0.8\sqrt{d} + \eta K \|G_t\|_2.$$

Here, we applied the inductive hypothesis on all $\|x_{s,k}\|_2$, the initial bound $\|x_{0,0}\|_2 \leq \sqrt{\frac{1}{2}d}$, and that $\eta^2 K^2 T = o(1)$ by assumption. We also defined $G_t = \sum_{s=0}^{t} v_{t,0}$, where $v_{t,0}$ is the random Gaussian used by HMC in iteration k; note that by independence, $G_t \sim \mathcal{N}(0, t+1)$. By Fact 32, with probability at least $\frac{1}{200T}$, $\|G_t\|_2 \leq 2\sqrt{Td}$, and hence $0.8\sqrt{d} + \eta K \|G_t\|_2 \leq 0.9\sqrt{d}$, as desired.

Next, we prove that with probability $\geq 1 - \frac{1}{200T}$, step t does not reject. This concludes the proof by union bounding over both events in iteration t, and then union bounding over all iterations. By Corollary 53 and the calculation in Lemma 206, when $\eta^2 K^2 \leq 1$, the accept probability is

$$\min\left(1, \exp\left(\frac{h}{4}\left(\left(2\alpha - \alpha^2\right)\|x_{t,0}\|_2^2 - \beta^2 \|v_{t,0}\|_2^2 - 2(1-\alpha)\beta\left\langle x_{t,0}, v_{t,0}\right\rangle\right)\right)\right),$$

for some $\alpha \in \left[0.8hK^2, hK^2\right]$ and $\beta \in \left[0.8\sqrt{2h}K, \sqrt{2h}K\right]$. We lower bound the argument of the exponential as follows. With probability at least $1 - d^{-5} \geq 1 - \frac{1}{400T}$, Facts 31 and 32 imply both of the events $\|v_{t,0}\|_2^2 \leq 2d$ and $\langle x_{t,0}, v_{t,0}\rangle \leq 10\sqrt{\log d}\,\|x_{t,0}\|_2$ occur. Conditional on these bounds, we compute (using $2\alpha \geq \alpha^2$ and the assumption $\|x_t\|_2 \leq 0.9\sqrt{d}$)

$$\left(2\alpha - \alpha^2\right)\|x_{t,0}\|_2^2 - \beta^2\|g\|_2^2 - 2(1-\alpha)\beta\langle x_{t,0}, g\rangle \geq -4hK^2d - 40\sqrt{h}K\sqrt{d\log d} \geq -O(K^2\log d).$$

Hence, the acceptance probability is at least

$$\exp\left(-O\left(\eta^2K^2\log d\right)\right) \geq 1 - \frac{1}{400T},$$

by our choice of T with $T\eta^2K^2\log d = o(1)$, concluding the proof. □

Proposition 85. *The HMC Markov chain with step size $\eta^2 = O\left(\frac{\log d}{\kappa d}\right)$ and $\eta^2K^2 \leq 1$ requires $\Omega(\frac{\kappa d}{K^2\log^3 d})$ iterations to reach total variation distance $\frac{1}{e}$ to π^*, starting from π_0.*

Proof. The proof is identical to Proposition 51, where we use Lemma 340 instead of Lemma 202. □

Bibliography

[1] Martín Abadi. Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*, page 1, 2016.

[2] Jacob D. Abernethy and Elad Hazan. Faster convex optimization: Simulated annealing with an efficient universal barrier. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2520–2528, 2016.

[3] Jacob D. Abernethy, Kevin A. Lai, Kfir Y. Levy, and Jun-Kun Wang. Faster rates for convex-concave games. In *31st Annual Conference on Computational Learning Theory (COLT)*, pages 1595–1625, 2018.

[4] Jacob D. Abernethy, Kevin A. Lai, and Andre Wibisono. Last-iterate convergence rates for min-max optimization. *CoRR*, abs/1906.02027, 2019.

[5] Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 335–344. IEEE Computer Society, 2016.

[6] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[7] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *International Conference on Computational Learning Theory*, pages 458–469. Springer, 2005.

[8] Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for p-norm regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1405–1424, 2019.

[9] Alekh Agarwal, Sahand Negahban, and Martin J Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, pages 2452–2482, 2012.

[10] Naman Agarwal, Sham M. Kakade, Rahul Kidambi, Yin Tat Lee, Praneeth Netrapalli, and Aaron Sidford. Leverage score sampling for faster accelerated regression and ERM. In *Algorithmic Learning Theory, ALT 2020*, pages 22–47, 2020.

[11] Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 555–564, 2014.

[12] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.

[13] Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.*, 4(4):17:1–17:40, 2018.

[14] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467, 2012.

[15] Abhishek Aich and P Palanisamy. On application of omp and cosamp algorithms for doa estimation problem. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, pages 1983–1987. IEEE, 2017.

[16] Ahmet Alacaoglu and Yura Malitsky. Stochastic variance reduction for variational inequality methods. *arXiv e-prints*, abs/2102.08352, 2021.

[17] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *J. Mach. Learn. Res.*, 18:221:1–221:51, 2017.

[18] Zeyuan Allen Zhu. Katyusha: the first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1200–1205, 2017.

[19] Zeyuan Allen-Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive sdp solver. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1824–1831. Society for Industrial and Applied Mathematics, 2016.

[20] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the compressed leader: Faster online learning of eigenvectors and faster MMWU. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 116–125, 2017.

[21] Zeyuan Allen Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 237–245, 2015.

[22] Zeyuan Allen-Zhu, Zhenyu Liao, and Yang Yuan. Optimization algorithms for faster computational geometry. In *43rd International Colloquium on Automata, Languages, and Programming*, pages 53:1–53:6, 2016.

[23] Zeyuan Allen Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 229–236, 2015.

[24] Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively scalable sinkhorn distances via the nyström method. In *Advances in Neural Information Processing Systems*, pages 4427–4437, 2019.

[25] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1964–1974, 2017.

[26] Nima Anari, Callum Burgess, Kevin Tian, and Thuy-Duong Vuong. Improved sampling-to-counting reductions in high-dimensional expanders and faster parallel determinantal sampling. *CoRR*, abs/2203.11190, 2022.

[27] ED Andersen, C Roos, T Terlaky, T Trafalis, and JP Warners. The use of low-rank updates in interior-point methods. *Numerical Linear Algebra and Optimization*, pages 1–12, 1996.

[28] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014.

[29] Alexandr Andoni, Clifford Stein, and Peilin Zhong. Parallel approximate undirected shortest paths via low hop emulators. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 322–335. ACM, 2020.

[30] Richard André-Jeannin. A generalization of morgan-voyce polynomials. *Fibonacci Quarterly*, 32(3), 1994.

[31] Frank J Anscombe. Rejection of outliers. *Technometrics*, 2(2):123–146, 1960.

[32] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 214–223, 2017.

[33] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8(1):121–164, 2012.

[34] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *J. ACM*, 63(2):12:1–12:35, 2016.

[35] Sanjeev Arora and Ravi Kannan. Learning mixtures of separated nonspherical gaussians. *The Annals of Applied Probability*, 15(1A):69–92, 2005.

[36] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5:1–5:37, 2009.

[37] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635, 2019.

[38] Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[39] Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 627–669. SIAM, 2022.

[40] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017.

[41] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016.

[42] Sepehr Assadi, Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *4th Symposium on Simplicity in Algorithms, SOSA@SODA 2021, January 2021*, 2021.

[43] Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 342–353. IEEE, 2020.

[44] Pranjal Awasthi. Communication. Personal, September 2021.

[45] Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 37–49. Springer, 2012.

[46] Pranjal Awasthi and Aravindan Vijayaraghavan. Towards learning sparsely used dictionaries with arbitrary supports. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 283–296. IEEE Computer Society, 2018.

[47] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

[48] Dmitry Babichev, Dmitrii Ostrovskii, and Francis Bach. Efficient primal-dual algorithms for large-scale multiclass classification. *arXiv preprint arXiv:1902.03755*, 2019.

[49] Francis Bach. Polynomial magic i: Chebyshev polynomials. `https://francisbach.com/chebyshev-polynomials/`, 2019.

[50] Michel Baes, Michael Bürgisser, and Arkadi Nemirovski. A randomized mirror-prox method for solving structured large-scale matrix saddle-point problems. *SIAM J. Optimization*, 23(2):934–962, 2013.

[51] Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control variates for stochastic gradient mcmc. *Statistics and Computing*, 29(3):599–615, 2019.

[52] Dominique Bakry, Ivan Gentil, and Michel Ledoux. *Analysis and Geometry of Markov Diffusion Operators*. Springer, 2014.

[53] Ainesh Bakshi and Adarsh Prasad. Robust linear regression: Optimal rates in polynomial time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, 2021.

[54] Sivaraman Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pages 169–212, 2017.

[55] Palaniappan Balamurugan and Francis R. Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, 2016.

[56] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680, 2008.

[57] Afonso S Bandeira, Edgar Dobriban, Dustin G Mixon, and William F Sawin. Certifying the restricted isometry property is hard. *IEEE transactions on information theory*, 59(6):3448–3450, 2013.

[58] Boaz Barak, Fernando GSL Brandao, Aram W Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 307–326, 2012.

[59] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on information theory*, 56(4):1982–2001, 2010.

[60] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[61] M Barkhagen, NH Chau, É Moulines, M Rásonyi, S Sabanis, and Y Zhang. On stochastic gradient langevin dynamics with dependent data streams in the logconcave case. *arXiv preprint arXiv:1812.02709*, 2018.

[62] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010.

[63] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 464–473. IEEE Computer Society, 2014.

[64] Heinz H. Bauschke, Jérôme Bolte, and Marc Teboulle. A descent lemma beyond lipschitz gradient continuity: First-order methods revisited and applications. *Math. Oper. Res.*, 42(2):330–348, 2017.

[65] Amir Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, 2017.

[66] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[67] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[68] Stephen Becker, Jérôme Bobin, and Emmanuel J Candès. Nesta: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2011.

[69] Soheil Behnezhad, Mahsa Derakhshan, Hossein Esfandiari, Elif Tan, and Hadi Yami. Brief announcement: Graph matching in massive datasets. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 133–136, 2017.

[70] Pierre C Bellec. The noise barrier and the large signal bias of the lasso and other convex estimators. *arXiv preprint arXiv:1804.01230*, 2018.

[71] Michele Benzi and Miroslav Tûma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2):305–340, 1999.

[72] Aaron Bernstein. Improved bounds for matching in random-order streams. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 12:1–12:13, 2020.

[73] Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1123–1134, 2020.

[74] Espen Bernton. Langevin monte carlo and JKO splitting. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1777–1798, 2018.

[75] Dimitris Bertsimas and Santosh S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[76] Julian Besag. Comments on "representations of knowledge in complex systems" by u. grenander and mi miller. *Journal of the Royal Statistical Society, Series B*, 56:591–592, 1994.

[77] Aditya Bhaskara, Aravinda Kanchana Ruwanpathirana, and Maheshakya Wijewardena. Principal component regression with semirandom observations via matrix completion. In *International Conference on Artificial Intelligence and Statistics*, pages 2665–2673. PMLR, 2021.

[78] Vijay Bhattiprolu, Mrinalkanti Ghosh, Venkatesan Guruswami, Euiwoong Lee, and Madhur Tulsiani. Approximability of $p \to q$ matrix norms: generalized Krivine rounding and hypercontractive hardness. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1358–1368. SIAM, 2019.

[79] Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of statistics*, 37(4):1705–1732, 2009.

[80] Joris Bierkens, Paul Fearnhead, Gareth Roberts, et al. The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

[81] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[82] Jose H. Blanchet, Arun Jambulapati, Carson Kent, and Aaron Sidford. Towards optimal running times for optimal transport. *CoRR*, abs/1810.07717, 2018.

[83] Jose H. Blanchet and Yang Kang. Distributionally robust groupwise regularization estimator. In *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017, Seoul, Korea, November 15-17, 2017.*, pages 97–112, 2017.

[84] Avrim Blum. Machine learning: My favorite results, directions, and open problems. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 2–2. IEEE, 2003.

[85] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[86] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[87] Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of selected topics in signal processing*, 4(2):298–309, 2010.

[88] Sergey G Bobkov and Michel Ledoux. Poincaré's inequalities and talagrand's concentration phenomenon for the exponential distribution. *Probability Theory and Related Fields*, 107(3):383–400, 1997.

[89] Sergey G Bobkov and Michel Ledoux. From brunn-minkowski to brascamp-lieb and to logarithmic sobolev inequalities. *GAFA, Geometric and Functional Analysis*, 10:1028–1052, 2000.

[90] Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, 2011.

[91] Digvijay Boob, Saurabh Sawlani, and Di Wang. Faster width-dependent algorithm for mixed packing and covering lps. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 15253–15262, 2019.

[92] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018.

[93] Nawaf Bou-Rabee and Andreas Eberle. Mixing time guarantees for unadjusted hamiltonian monte carlo. *CoRR*, abs/2105.00887, 2021.

[94] Nawaf Bou-Rabee and Martin Hairer. Nonasymptotic mixing of the mala algorithm. *IMA Journal of Numerical Analysis*, 33(1):80–110, 2012.

[95] Herm Jan Brascamp and Elliott H Lieb. On extensions of the brunn-minkowski and prékopa-leindler theorems, including inequalities for log concave functions, and with an application to the diffusion equation. *Journal of Functional Analysis*, 22(4):366–389, 1976.

[96] Nicolas Brosse, Alain Durmus, Eric Moulines, and Marcelo Pereyra. Sampling from a log-concave distribution with compact support with proximal langevin monte carlo. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 319–342, 2017.

[97] S. Charles Brubaker and Santosh S. Vempala. Isotropic PCA and affine-invariant clustering. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 551–560, 2008.

[98] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[99] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[100] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for l_p regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1130–1137, 2018.

[101] Sébastien Bubeck, Ronen Eldan, and Joseph Lehec. Sampling from a log-concave distribution with projected langevin monte carlo. *Discret. Comput. Geom.*, 59(4):757–783, 2018.

[102] Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13900–13909, 2019.

[103] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[104] Emmanuel J Candes and Justin K Romberg. Signal recovery from random projections. In *Computational Imaging III*, volume 5674, pages 76–86. International Society for Optics and Photonics, 2005.

[105] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[106] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[107] Yu Cao, Jianfeng Lu, and Lihan Wang. Complexity of randomized algorithms for underdamped langevin dynamics. *CoRR*, abs/2003.09906, 2020.

[108] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 589–623, 2019.

[109] Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, and Kevin Tian. Acceleration with a ball optimization oracle. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[110] Yair Carmon, Arun Jambulapati, Yujia Jin, and Aaron Sidford. Thinking inside the ball: Near-optimal minimization of the maximal loss. *CoRR*, abs/2105.01778, 2021.

[111] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Variance reduction for matrix games. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 11377–11388, 2019.

[112] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Coordinate methods for matrix games. In *61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 283–293, 2020.

[113] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1), 2017.

[114] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[115] Deeparnab Chakrabarty and Sanjeev Khanna. Better and simpler error analysis of the sinkhorn-knopp algorithm for matrix scaling. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 4:1–4:11, 2018.

[116] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

[117] Yi-Jun Chang, Martin Farach-Colton, Tsan-sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 34:1–34:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[118] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60, 2017.

[119] Niladri Chatterji, Nicolas Flammarion, Yian Ma, Peter Bartlett, and Michael Jordan. On the theory of variance reduction for stochastic gradient monte carlo. In *International Conference on Machine Learning*, pages 764–773, 2018.

[120] Niladri S. Chatterji, Peter L. Bartlett, and Philip M. Long. Oracle lower bounds for stochastic gradient sampling algorithms. *CoRR*, abs/2002.00291, 2020.

[121] Tatjana Chavdarova, Gauthier Gidel, François Fleuret, and Simon Lacoste-Julien. Reducing noise in GAN training with variance reduced extragradient. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 391–401, 2019.

[122] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, pages 195–199, 1969.

[123] Changyou Chen, Wenlin Wang, Yizhe Zhang, Qinliang Su, and Lawrence Carin. A convergence analysis for a class of practical variance-reduction stochastic gradient mcmc. *arXiv preprint arXiv:1709.01180*, 2017.

[124] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *arXiv preprint arXiv:2203.00671*, 2022.

[125] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 570–583. ACM, 2021.

[126] Sitan Chen, Frederic Koehler, Ankur Moitra, and Morris Yau. Classification under misspecification: Halfspaces, generalized linear models, and connections to evolvability. *arXiv preprint arXiv:2006.04787*, 2020.

[127] Yuansi Chen. An almost constant lower bound of the isoperimetric coefficient in the kls conjecture. *CoRR*, abs/2011.13661, 2021.

[128] Yuansi Chen, Raaz Dwivedi, Martin J. Wainwright, and Bin Yu. Fast mixing of metropolized hamiltonian monte carlo: Benefits of multi-step gradients. *CoRR*, abs/1905.12247, 2019.

[129] Zongchen Chen and Santosh S. Vempala. Optimal convergence rate of hamiltonian monte carlo for strongly logconcave distributions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, pages 64:1–64:12, 2019.

[130] Xiang Cheng, Niladri S. Chatterji, Peter L. Bartlett, and Michael I. Jordan. Underdamped langevin MCMC: A non-asymptotic analysis. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 300–323, 2018.

[131] Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2755–2771, 2019.

[132] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David Woodruff. Faster algorithms for high-dimensional robust covariance estimation. *arXiv preprint arXiv:1906.04661*, 2019.

[133] Yu Cheng and Rong Ge. Non-convex matrix completion against a semi-random adversary. In *Conference On Learning Theory*, pages 1362–1394. PMLR, 2018.

[134] Yeshwanth Cherapanamjeri, Efe Aras, Nilesh Tripuraneni, Michael I. Jordan, Nicolas Flammarion, and Peter L. Bartlett. Optimal robust linear regression in nearly linear time. *CoRR*, abs/2007.08137, 2020.

[135] Yeshwanth Cherapanamjeri, Nicolas Flammarion, and Peter L. Bartlett. Fast mean estimation with sub-gaussian rates. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 786–806, 2019.

[136] Yeshwanth Cherapanamjeri, Sidhanth Mohanty, and Morris Yau. List decodable mean estimation in nearly linear time. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 141–148, 2020.

[137] Sinho Chewi, Chen Lu, Kwangjun Ahn, Xiang Cheng, Thibaut Le Gouic, and Philippe Rigollet. Optimal dimension dependence of the metropolis-adjusted langevin algorithm. *CoRR*, abs/2012.12810, 2020.

[138] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344, 2016.

[139] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282, 2011.

[140] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. In *51th Annual IEEE Symposium on Foundations of Computer Science*, pages 449–457, 2010.

[141] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 81–90. ACM, 2013.

[142] Michael Cohen, Jelena Diakonikolas, and Lorenzo Orecchia. On acceleration with noise-corrupted gradients. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1018–1027, 2018.

[143] Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 9–21. ACM, 2016.

[144] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *J. ACM*, 68(1):3:1–3:39, 2021.

[145] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 902–913, 2017.

[146] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[147] Michael B. Cohen, Aaron Sidford, and Kevin Tian. Relative lipschitzness in extragradient methods and a direct recipe for acceleration. In *12th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 62:1–62:18, 2021.

[148] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[149] Ben Cousins and Santosh S. Vempala. Gaussian cooling and $o^{*}(n^3)$ algorithms for volume and gaussian volume. *SIAM J. Comput.*, 47(3):1237–1273, 2018.

[150] Benjamin Cousins and Santosh Vempala. Bypassing kls: Gaussian cooling and an oˆ*(n3) volume algorithm. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 539–548, 2015.

[151] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[152] Arnak S. Dalalyan. Further and stronger analogy between sampling and optimization: Langevin monte carlo and gradient descent. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 678–689, 2017.

[153] Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 3(79):651–676, 2017.

[154] Arnak S Dalalyan, Mohamed Hebiri, and Johannes Lederer. On the prediction performance of the lasso. *Bernoulli*, 23(1):552–581, 2017.

[155] Arnak S Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019.

[156] Arnak S. Dalalyan and Lionel Riou-Durand. On sampling from a log-concave density using kinetic langevin diffusions. *CoRR*, abs/1807.09382, 2018.

[157] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1953.

[158] Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 30:1–30:14, 2020.

[159] Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 634–644. IEEE, 1999.

[160] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.

[161] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2019.

[162] Alexandre d'Aspremont. Smooth optimization with approximate gradient. *SIAM J. Optim.*, 19(3):1171–1183, 2008.

[163] Alexandre d'Aspremont. Subsampling algorithms for semidefinite programming. *Stochastic Systems*, 1(2):209–436, 2011.

[164] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[165] Mark A Davenport, Marco F Duarte, Yonina C Eldar, and Gitta Kutyniok. Introduction to compressed sensing., 2012.

[166] Mark A Davenport, Deanna Needell, and Michael B Wakin. Signal space cosamp for sparse recovery with redundant dictionaries. *IEEE Transactions on Information Theory*, 59(10):6820–6829, 2013.

[167] Aaron Defazio. A simple practical accelerated method for finite sums. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 676–684, 2016.

[168] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 1646–1654, 2014.

[169] Erik Demaine. Lecture 19. Class notes, MIT 6.851: Advanced Data Structures, scribed by Justin Holmgren, Jing Jian, Maksim Stepaneko, Mashhood Ishaque, 2012.

[170] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[171] Jules Despersin and Guillaume Lecué. Robust subgaussian estimation of a mean vector in nearly linear time. *CoRR*, abs/1906.03058, 2019.

[172] Olivier Devolder, François Glineur, and Yurii E. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146(1-2):37–75, 2014.

[173] Ilias Diakonikolas, Themis Gouleakis, and Christos Tzamos. Distribution-independent pac learning of halfspaces with massart noise. *arXiv preprint arXiv:1906.10075*, 2019.

[174] Ilias Diakonikolas, Russell Impagliazzo, Daniel Kane, Rex Lei, Jessica Sorrell, and Christos Tzamos. Boosting in the presence of massart noise. *arXiv preprint arXiv:2106.07779*, 2021.

[175] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM J. Comput.*, 48(2):742–864, 2019.

[176] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 1596–1606, 2019.

[177] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 999–1008. JMLR. org, 2017.

[178] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robustly learning a gaussian: Getting optimal error, efficiently. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2683–2702. SIAM, 2018.

[179] Ilias Diakonikolas, Daniel Kane, and Daniel Kongsgaard. List-decodable mean estimation via iterative multi-filtering. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[180] Ilias Diakonikolas, Daniel Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. List-decodable mean estimation in nearly-pca time. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 10195–10208, 2021.

[181] Ilias Diakonikolas, Daniel Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. Clustering mixture models in almost-linear time via list-decodable mean estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022*, 2022.

[182] Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

[183] Ilias Diakonikolas and Daniel M Kane. Hardness of learning halfspaces with massart noise. *arXiv preprint arXiv:2012.09720*, 2020.

[184] Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Threshold phenomena in learning halfspaces with massart noise. *arXiv preprint arXiv:2108.08767*, 2021.

[185] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–84. IEEE, 2017.

[186] Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1047–1060, 2018.

[187] Ilias Diakonikolas, Daniel M Kane, and Christos Tzamos. Forster decomposition and learning halfspaces with noise. *arXiv preprint arXiv:2107.05582*, 2021.

[188] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2745–2754, 2019.

[189] Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Learning halfspaces with massart noise under structured distributions. In *Conference on Learning Theory*, pages 1486–1513. PMLR, 2020.

[190] Ilias Diakonikolas, Jongho Park, and Christos Tzamos. Relu regression with massart noise. *arXiv preprint arXiv:2109.04623*, 2021.

[191] Jelena Diakonikolas, Maryam Fazel, and Lorenzo Orecchia. Width-independence beyond linear objectives: Distributed fair packing and covering algorithms. *CoRR*, abs/1808.02517, 2018.

[192] Jelena Diakonikolas and Lorenzo Orecchia. Accelerated extra-gradient descent: A novel accelerated first-order method. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 23:1–23:19, 2018.

[193] Jelena Diakonikolas and Lorenzo Orecchia. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019.

[194] Yihe Dong, Samuel Hopkins, and Jerry Li. Quantum entropy scoring for fast robust mean estimation and improved outlier detection. In *Advances in Neural Information Processing Systems*, pages 6065–6075, 2019.

[195] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[196] David L Donoho and Philip B Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

[197] Radu-Alexandru Dragomir, Adrien Taylor, Alexandre d'Aspremont, and Jérôme Bolte. Optimal complexity and certification of bregman first-order methods. *arXiv e-prints*, abs/1911.08510, 2019.

[198] Yoel Drori, Shoham Sabach, and Marc Teboulle. A simple algorithm for a class of nonsmooth convex-concave saddle-point problems. *Operations Research Letters*, 43(2):209–214, 2015.

[199] Vladimir Druskin and Leonid Knizhnerman. Error bounds in the simple lanczos procedure for computing functions of symmetric matrices and eigenvalues. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 31(7):970–983, 1991.

[200] Vladimir Druskin and Leonid Knizhnerman. Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic. *Numerical Linear Algebra with Applications*, 2(3):205–217, 1995.

[201] Simon S. Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *34th International Conference on Machine Learning (ICML)*, pages 1049–1058, 2017.

[202] Kumar Avinava Dubey, Sashank J Reddi, Sinead A Williamson, Barnabas Poczos, Alexander J Smola, and Eric P Xing. Variance reduction in stochastic gradient langevin dynamics. In *Advances in neural information processing systems*, pages 1154–1162, 2016.

[203] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[204] David Durfee, Yu Gao, Gramoz Goranci, and Richard Peng. Fully dynamic spectral vertex sparsifiers and applications. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 914–925. ACM, 2019.

[205] Alain Durmus, Szymon Majewski, and Blazej Miasojedow. Analysis of langevin monte carlo via convex optimization. *J. Mach. Learn. Res.*, 20:73:1–73:46, 2019.

[206] Alain Durmus and Éric Moulines. High-dimensional bayesian inference via the unadjusted langevin algorithm. *Bernoulli*, 25(4A):2854–2882, 2019.

[207] Alain Durmus, Umut Simsekli, Eric Moulines, Roland Badeau, and Gaël Richard. Stochastic gradient richardson-romberg markov chain monte carlo. In *Advances in Neural Information Processing Systems*, pages 2047–2055, 2016.

[208] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1366–1375, 2018.

[209] Pavel E. Dvurechensky and Alexander V. Gasnikov. Stochastic intermediate gradient method for convex problems with stochastic inexact oracle. *J. Optim. Theory Appl.*, 171(1):121–145, 2016.

[210] Raaz Dwivedi, Yuansi Chen, Martin J. Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast! In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 793–797, 2018.

[211] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 11–20. ACM, 2014.

[212] Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[213] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

[214] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge university press, 2012.

[215] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Math. Program.*, 171(1-2):115–166, 2018.

[216] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233, 2015.

[217] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. SPIDER: near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, 2018.

[218] Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1773–1785, 2020.

[219] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001.

[220] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures & Algorithms*, 16(2):195–208, 2000.

[221] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

[222] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

[223] Mário AT Figueiredo and Robert D Nowak. An em algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[224] Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model. *CoRR*, abs/2106.04179, 2021.

[225] G. E. Forsythe and E. G. Straus. On best conditioned matrices. *Proceedings of the American Mathematical Society*, 6(3):340–345, 1955.

[226] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[227] Roy Frostig, Rong Ge, Sham M. Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *32nd International Conference on Machine Learning (ICML)*, pages 2540–2548, 2015.

[228] Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500, 2019.

[229] Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Global convergence of stochastic gradient hamiltonian monte carlo for non-convex stochastic optimization: Non-asymptotic performance bounds and momentum-based acceleration. *arXiv preprint arXiv:1809.04618*, 2018.

[230] Yu Gao, Yang P. Liu, and Richard Peng. Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 516–527. IEEE, 2021.

[231] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Math. Program.*, 158(1-2):329–361, 2016.

[232] Dan Garber, Elad Hazan, and Tengyu Ma. Online learning of eigenvectors. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 560–568, 2015.

[233] Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. Operator scaling: Theory and applications. *Found. Comput. Math.*, 20(2):223–290, 2020.

[234] Ankit Garg and Rafael Oliveira. Recent progress on scaling algorithms and applications. *Bulletin of EATCS*, 2(125), 2018.

[235] Rong Ge, Holden Lee, and Jianfeng Lu. Estimating normalizing constants for log-concave distributions: algorithms and lower bounds. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 579–586, 2020.

[236] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis R. Bach. Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3432–3440, 2016.

[237] Gauthier Gidel, Tony Jebara, and Simon Lacoste-Julien. Frank-Wolfe algorithms for saddle point problems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[238] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[239] Wayne Goddard. Part 2: Greedy algorithms, dynamic programming, graph algorithms. Introduction to Algorithms, 2004.

[240] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

[241] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[242] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.

[243] Andrew V. Goldberg and Robert Endre Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4):873–886, 1989.

[244] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020.

[245] Gramoz Goranci. Dynamic graph algorithms and graph sparsification: New techniques and connections. *CoRR*, abs/1909.06413, 2019.

[246] Gramoz Goranci, Monika Henzinger, and Pan Peng. The power of vertex sparsifiers in dynamic graph algorithms. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[247] Gramoz Goranci, Monika Henzinger, and Pan Peng. Dynamic Effective Resistances and Approximate Schur Complement on Separable Graphs. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[248] Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2212–2228. SIAM, 2021.

[249] Rachel A. Gordon. *Regression Analysis for the Social Sciences*. Routledge, 2010.

[250] Fabrizio Grandoni, Stefano Leonardi, Piotr Sankowski, Chris Schwiegelshohn, and Shay Solomon. $(1+\epsilon)$-approximate incremental matching in constant deterministic amortized time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1886–1898, 2019.

[251] A. Greenbaum and G. H. Rodrigue. Optimal preconditioners of a given sparsity pattern. *BIT Numerical Mathematics*, 29(4):610–634, 1989.

[252] William H. Greene. *Econometric analysis*. Prentice Hall, 1990.

[253] Michael D. Grigoriadis and Leonid G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operation Research Letters*, 18(2):53–58, 1995.

[254] Marcus J. Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[255] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[256] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridagional eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995.

[257] Osman Güler. New proximal point algorithms for convex minimization. *SIAM J. Optim.*, 2(4):649–664, 1992.

[258] Manoj Gupta. Maintaining approximate maximum matching in an incremental bipartite graph in polylogarithmic update time. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 227–239, 2014.

[259] Manoj Gupta and Richard Peng. Fully dynamic $(1+\epsilon)$-approximate matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 548–557, 2013.

[260] Neha Gupta and Aaron Sidford. Exploiting numerical sparsity for efficient learning: faster eigenvector computation and regression. In *Advances in Neural Information Processing Systems*, pages 5269–5278, 2018.

[261] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013.

[262] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Fully-dynamic all-pairs shortest paths: Improved worst-case time and space bounds. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2562–2574. SIAM, 2020.

[263] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: the Approach based on Influence Functions*. Wiley Series in Probability and Mathematical Statistics, 1986.

[264] Kathrin Hanauer, Monika Henzinger, and Christian Schulz. Recent advances in fully dynamic graph algorithms. *CoRR*, abs/2102.11169, 2021.

[265] Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated bregman proximal gradient methods for relatively smooth convex optimization. *arXiv e-prints*, abs/1808.03045, 2018.

[266] Moritz Hardt. The zen of gradient descent. http://blog.mrtz.org/2013/09/07/the-zen-of-gradient-descent.html, 2013. Accessed: 2018-07-30.

[267] Gilles Hargé. A convex/log-concave correlation inequality for gaussian measure and an application to abstract wiener spaces. *Probability theory and related fields*, 130(3):415–440, 2004.

[268] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2016.

[269] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 489–498. ACM, 2016.

[270] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30, 2015.

[271] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. Springer, New York, 1993.

[272] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[273] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[274] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[275] Samuel B Hopkins and Jerry Li. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1021–1034, 2018.

[276] Ya-Ping Hsieh, Ali Kavis, Paul Rolland, and Volkan Cevher. Mirrored langevin dynamics. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 2883–2892, 2018.

[277] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 6936–6946, 2019.

[278] Peter J Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, pages 73–101, 1964.

[279] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004.

[280] Prateek Jain, Ambuj Tewari, and Purushottam Kar. On iterative hard thresholding methods for high-dimensional m-estimation. In *NIPS*, 2014.

[281] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *J. ACM*, 58(6):30:1–30:27, 2011.

[282] Rahul Jain and Penghui Yao. A parallel approximation algorithm for positive semidefinite programming. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 463–471, 2011.

[283] Rahul Jain and Penghui Yao. A parallel approximation algorithm for mixed packing and covering semidefinite programs. *CoRR*, abs/1201.6090, 2012.

[284] Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Regularized box-simplex games and dynamic decremental bipartite matching. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, 2022.

[285] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: Mixed, parallel, and width-independent. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020.

[286] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: Mixed, parallel, and width-independent. *CoRR*, abs/2002.04830v3, 2021.

[287] Arun Jambulapati, Jerry Li, Christopher Musco, Aaron Sidford, and Kevin Tian. Fast and near-optimal diagonal preconditioning. *CoRR*, abs/2008.01722, 2021.

[288] Arun Jambulapati, Jerry Li, Tselil Schramm, and Kevin Tian. Robust regression revisited: Acceleration and improved estimation rates. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4475–4488, 2021.

[289] Arun Jambulapati, Jerry Li, and Kevin Tian. Robust sub-gaussian principal component analysis and width-independent schatten packing. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[290] Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1664–1686. IEEE Computer Society, 2019.

[291] Arun Jambulapati, Kirankumar Shiragur, and Aaron Sidford. Efficient structured matrix recovery and nearly-linear time algorithms for solving inverse symmetric m-matrices. *CoRR*, abs/1812.06295, 2018.

[292] Arun Jambulapati and Aaron Sidford. Ultrasparse ultrasparsifiers and faster laplacian system solvers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 540–559, 2021.

[293] Arun Jambulapati, Aaron Sidford, and Kevin Tian. A direct tilde{O}(1/epsilon) iteration parallel algorithm for optimal transport. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11355–11366, 2019.

[294] Mark Jerrum. Large cliques elude the metropolis process. *Random Struct. Algorithms*, 3(4):347–360, 1992.

[295] He Jia, Aditi Laddha, Yin Tat Lee, and Santosh S. Vempala. Reducing isotropy and volume to KLS: an o($n^3\psi^2$) volume algorithm. *CoRR*, abs/2008.02146, 2020.

[296] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 944–953. ACM, 2020.

[297] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Minmax optimization: Stable limit points of gradient descent ascent are locally optimal. *arXiv preprint arXiv:1902.00618*, 2019.

[298] Yujia Jin and Aaron Sidford. Principal component projection and regression in nearly linear time through asymmetric SVRG. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3863–3873, 2019.

[299] Yujia Jin, Aaron Sidford, and Kevin Tian. Sharper rates for separable minimax and finite sum optimization via primal-dual extragradient methods. *CoRR*, abs/2202.04640, 2022.

[300] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26 (NeurIPS)*, pages 315–323, 2013.

[301] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker-planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

[302] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

[303] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Applications of strong convexity–strong smoothness duality to learning with matrices. *arXiv e-prints*, abs/0910.0610, 2009.

[304] Adam Tauman Kalai and Santosh S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.

[305] Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 15:1–15:21, 2017.

[306] Donggu Kang and James Payor. Flow rounding. *CoRR*, abs/1507.08139, 2015.

[307] Ravindran Kannan, Hadi Salmasian, and Santosh S. Vempala. The spectral method for general mixture models. *SIAM J. Comput.*, 38(3):1141–1156, 2008.

[308] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013.

[309] Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, 2021.

[310] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014.

[311] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014.

[312] Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1753–1772, 2020.

[313] David R. Karger. Better random sampling algorithms for flows in undirected graphs. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California.*, pages 490–499, 1998.

[314] David R. Karger and Matthew S. Levine. Random sampling in residual graphs. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 63–66, 2002.

[315] Sushrut Karmalkar, Adam R. Klivans, and Pravesh Kothari. List-decodable linear regression. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7423–7432, 2019.

[316] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[317] Jonathan Kelner, Frederic Koehler, Raghu Meka, and Dhruv Rohatgi. On the power of preconditioning in sparse linear regression. *arXiv preprint arXiv:2106.09207*, 2021.

[318] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226, 2014.

[319] Jonathan A. Kelner, Jerry Li, Allen Liu, Aaron Sidford, and Kevin Tian. Semi-random sparse recovery in nearly-linear time. *CoRR*, abs/2203.04002, 2022.

[320] Peter Kiss. Deterministic dynamic matching in worst-case update time. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 94:1–94:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[321] Adam R. Klivans, Pravesh K. Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1420–1430, 2018.

[322] Philip A. Knight, Daniel Ruiz, and Bora Uçar. A symmetry preserving algorithm for matrix scaling. *SIAM Journal on Matrix Analysis and Applications*, 35(3):931–955, 2014.

[323] O Kolossoski and Renato DC Monteiro. An accelerated non-Euclidean hybrid proximal extragradient-type algorithm for convex-concave saddle-point problems. *Optimization Methods and Software*, 32(6):1244–1272, 2017.

[324] Vladimir Koltchinskii and Stanislav Minsker. l_1-penalization in functional linear regression with subgaussian design. *Journal de l'Ecole polytechnique-Mathématiques*, 1:269–330, 2014.

[325] Christian Konrad. Maximum matching in turnstile streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 840–852, 2015.

[326] Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 74:1–74:16, 2018.

[327] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012.

[328] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287, 2016.

[329] G. M. Korpelevich. An extragradient method for finding saddle points and for other problems. *Ekonomika i Matematicheskie Metody*, 12(4):747–756, 1976.

[330] Pravesh K Kothari, Jacob Steinhardt, and David Steurer. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1035–1046, 2018.

[331] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 235–244, 2010.

[332] Dmitry Kovalev, Alexander V. Gasnikov, and Peter Richtárik. Accelerated primal-dual gradient method for smooth and convex-concave saddle-point problems with bilinear coupling. *arXiv e-prints*, abs/2112.15199, 2021.

[333] Dmitry Kovalev, Adil Salim, and Peter Richtárik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 18342–18352, 2020.

[334] Hendrik Anthony Kramers. Brownian motion in a field of force and the diffusion model of chemical reactions. *Physica*, 7(4):284–304, 1940.

[335] Gitta Kutyniok. Theory and applications of compressed sensing. *GAMM-Mitteilungen*, 36(1):79–101, 2013.

[336] Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. A graph theoretic additive approximation of optimal transport. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13813–13823, 2019.

[337] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016.

[338] Guanghui Lan, Zhize Li, and Yi Zhou. A unified variance-reduced accelerated gradient method for convex optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence

d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 10462–10472, 2019.

[339] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4), 1950.

[340] Béatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302–1338, 2000.

[341] Michel Ledoux. *Concentration of measure and logarithmic Sobolev inequalities*. Seminaire de probabilities XXXIII, 1999.

[342] Yin Tat Lee. Lecture 8: Stochastic methods and applications. Class notes, UW CSE 599: Interplay between Convex Optimization and Geometry, 2018.

[343] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 755–764, 2013.

[344] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Logsmooth gradient concentration and tighter runtimes for metropolized hamiltonian monte carlo. In *Conference on Learning Theory, COLT 2020*, 2020.

[345] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Lower bounds on metropolized sampling methods for well-conditioned distributions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18812–18824, 2021.

[346] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Structured logconcave sampling with a restricted gaussian oracle. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 2993–3050. PMLR, 2021.

[347] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 147–156, 2013.

[348] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014.

[349] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 230–249, 2015.

[350] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1049–1065. IEEE Computer Society, 2015.

[351] Yin Tat Lee, Zhao Song, and Santosh S. Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities. *CoRR*, abs/1812.06243, 2018.

[352] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 2140–2157, 2019.

[353] Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 678–687, 2017.

[354] David Asher Levin, Yuval Peres, and Elizabeth Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[355] Shlomo Levy and Peter K Fullagar. Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution. *Geophysics*, 46(9):1235–1243, 1981.

[356] Adrian Lewis. Convex analysis on the hermitian matrices. *SIAM Journal on Optimization*, 6(0):164–177, 1996.

[357] Adrian Lewis and Hristo S. Sendov. Twice differentiable spectral functions. *SIAM Journal on Matrix Analysis and Applications*, 23(0):368–386, 2001.

[358] Jason Li. Faster parallel algorithm for approximate shortest path. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 308–321. ACM, 2020.

[359] Jerry Li and Guanghao Ye. Robust gaussian covariance estimation in nearly-matrix multiplication time. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[360] Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.

[361] Jun Z. Li, Devin M. Absher, Hua Tang, Audrey M. Southwick, Amanda M. Casto, Sohini Ramachandran, Howard M. Cann, Gregory S. Barsh, Marcus Feldman, Luigi L. Cavalli-Sforza, and Richard M. Myers. Worldwide human relationships inferred from genome-wide patterns of variation. *Science*, 319:1100–1104, 2008.

[362] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 127–136, 2013.

[363] Henry Lin. Reducing directed max flow to undirected max flow. Unpublished Manuscript, 2009.

[364] Hongzhou Lin, Julien Mairal, and Zaïd Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 3384–3392, 2015.

[365] Tianyi Lin, Nhat Ho, and Michael I Jordan. On the acceleration of the sinkhorn and greenkhorn algorithms for optimal transport. *arXiv preprint arXiv:1906.01437*, 2019.

[366] Tianyi Lin, Chi Jin, and Michael I. Jordan. Near-optimal algorithms for minimax optimization. In *33rd Annual Conference on Computational Learning Theory (COLT)*, pages 2738–2779, 2020.

[367] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 644–652, 1998.

[368] S. Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *CoRR*, abs/2009.06106, 2020.

[369] Oren E. Livne and Gene H. Golub. Scaling by binormalization. *Numerical Algorithms*, 35(1):97–120, 2004.

[370] László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an o*(n4) volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006.

[371] László Lovász and Santosh S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 57–68, 2006.

[372] László Lovász and Santosh S. Vempala. Hit-and-run from a corner. *SIAM J. Comput.*, 35(4):985–1005, 2006.

[373] László Lovász and Santosh S. Vempala. The geometry of logconcave functions and sampling algorithms. *Random Struct. Algorithms*, 30(3):307–358, 2007.

[374] Haihao Lu. "relative-continuity" for non-lipschitz non-smooth convex optimization using stochastic (or deterministic) mirror descent. *INFORMS Journal on Optimization*, pages 288–303, 2019.

[375] Haihao Lu, Robert M. Freund, and Yurii E. Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM J. Optim.*, 28(1):333–354, 2018.

[376] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 245–254, 2010.

[377] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 253–262, 2013.

[378] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations (ICLR)*, 2018.

[379] Michael W. Mahoney, Satish Rao, Di Wang, and Peng Zhang. Approximating the solution to mixed packing and covering lps in parallel o~(epsilonˆ{-3}) time. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 52:1–52:14, 2016.

[380] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 367–384, 2012.

[381] Arian Maleki and David L Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):330–341, 2010.

[382] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[383] Oren Mangoubi and Aaron Smith. Mixing of hamiltonian monte carlo on strongly log-concave distributions 2: Numerical integrators. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 586–595, 2019.

[384] Oren Mangoubi and Nisheeth K. Vishnoi. Dimensionally tight bounds for second-order hamiltonian monte carlo. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6030–6040, 2018.

[385] Jelena Marasevic, Clifford Stein, and Gil Zussman. A fast distributed stateless algorithm for alpha-fair packing problems. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 54:1–54:15, 2016.

[386] Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning. *The Annals of Statistics*, 34(5):2326–2366, 2006.

[387] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 170–181, 2005.

[388] Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014.

[389] Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 14:1–14:4, 2018.

[390] Michela Meister and Gregory Valiant. A data prism: Semi-verified learning in the small-alpha regime. In *Conference On Learning Theory*, pages 1530–1546. PMLR, 2018.

[391] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[392] Gérard Meurant. *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*. Society for Industrial and Applied Mathematics, 2006.

[393] Marvin Minsky and Seymour Papert. *Perceptrons—an introduction to computational geometry*. MIT Press, 1987.

[394] Konstantin Mishchenko, Dmitry Kovalev, Egor Shulgin, Peter Richtárik, and Yura Malitsky. Revisiting stochastic extragradient. *arXiv preprint arXiv:1905.11373*, 2019.

[395] Hesameddin Mohammadi, Meisam Razaviyayn, and Mihailo R. Jovanovic. Performance of noisy nesterov's accelerated method for strongly convex optimization problems. In *2019 American Control Conference, ACC 2019, Philadelphia, PA, USA, July 10-12, 2019*, pages 3426–3431, 2019.

[396] Ankur Moitra. What does robustness say about algorithms. ICML '17 Tutorial, 2017.

[397] Ankur Moitra, Amelia Perry, and Alexander S Wein. How robust are reconstruction thresholds for community detection? In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 828–841, 2016.

[398] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. *arXiv preprint arXiv:1901.08511*, 2019.

[399] Cleve B. Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[400] Renato D. C. Monteiro and Benar Fux Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM J. Optim.*, 23(2):1092–1125, 2013.

[401] Wenlong Mou, Nicolas Flammarion, Martin J. Wainwright, and Peter L. Bartlett. An efficient sampling algorithm for non-smooth composite potentials. *CoRR*, abs/1910.00551, 2019.

[402] Wenlong Mou, Yi-An Ma, Martin J. Wainwright, Peter L. Bartlett, and Michael I. Jordan. High-order langevin diffusion yields an accelerated MCMC algorithm. *CoRR*, abs/1908.10859, 2019.

[403] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[404] Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1396–1404, 2015.

[405] Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the lanczos method for matrix function approximation. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1605–1624. SIAM, 2018.

[406] Tigran Nagapetyan, Andrew B Duncan, Leonard Hasenclever, Sebastian J Vollmer, Lukasz Szpruch, and Konstantinos Zygalakis. The true cost of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1706.02692*, 2017.

[407] Hongseok Namkoong and John C Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems*, pages 2208–2216, 2016.

[408] Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 950–961. IEEE Computer Society, 2017.

[409] Radford M Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

[410] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[411] Deanna Needell and Roman Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *IEEE Journal of selected topics in signal processing*, 4(2):310–316, 2010.

[412] Sahand N Negahban, Pradeep Ravikumar, Martin J Wainwright, and Bin Yu. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical science*, 27(4):538–557, 2012.

[413] Christopher Nemeth and Paul Fearnhead. Stochastic gradient markov chain monte carlo. *arXiv preprint arXiv:1907.06986*, 2019.

[414] A. Nemirovski and D.B̃. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

[415] Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[416] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[417] Yurii Nesterov. A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Doklady AN SSSR*, 269:543–547, 1983.

[418] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course, volume I*. Springer, 2003.

[419] Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005.

[420] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Math. Program.*, 109(2-3):319–344, 2007.

[421] Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming, Series A*, 110:245–259, 2007.

[422] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Math. Program.*, 120(1):261–283, 2009.

[423] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[424] Yurii Nesterov and Arkadi Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[425] Yurii E. Nesterov and Sebastian U. Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM J. Optim.*, 27(1):110–123, 2017.

[426] John Von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[427] Jiazhong Nie, Wojciech Kotlowski, and Manfred K. Warmuth. Online PCA with optimal regrets. In Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann, editors, *Algorithmic Learning Theory - 24th International Conference, ALT 2013, Singapore, October 6-9, 2013. Proceedings*, volume 8139 of *Lecture Notes in Computer Science*, pages 98–112. Springer, 2013.

[428] F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. V. Saunders B. R. Mille and, H. S. Cohl, and eds. M. A. McClain. Nist digital library of mathematical functions, 2020.

[429] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{o}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1141–1160, 2012.

[430] Felix Otto and Cédric Villani. Generalization of an inequality by talagrand and links with the logarithmic sobolev inequality. *Journal of Functional Analysis*, 173(2):361–400, 2000.

[431] Yuyuan Ouyang and Yangyang Xu. Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *Mathematical Programming*, 2019.

[432] Balamurugan Palaniappan and Francis R. Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1408–1416, 2016.

[433] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the 31st Annual ACM SIGACT Symposium on Theory of Computing, STOC 1999*, 1999.

[434] Victor M. Panaretos and Yoav Zemel. Amplitude and phase variation of point processes. *Annals of Statistics*, 44(2):771–812, 2016.

[435] Neal Parikh and Stephen P. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, 2014.

[436] Peristera Paschou, Jamey Lewis, Asif Javed, and Petros Drineas. Ancestry informative markers for fine-scale individual assignment to worldwide populations. *J Med Genet.*, 47(12):835–847, 2010.

[437] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[438] Seth Patinkin. Method, apparatus, and system for clustering and classification, August 30 2011. US Patent 8,010,466.

[439] Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2153–2161, 2017.

[440] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110, 1894.

[441] Richard Peng. Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1862–1867, 2016.

[442] Richard Peng, Kanat Tangwongsan, and Peng Zhang. Faster and simpler width-independent parallel algorithms for positive semidefinite programming. *CoRR*, abs/1201.5135, 2016.

[443] Marcelo Pereyra. Proximal markov chain monte carlo algorithms. *Stat. Comput.*, 26(4):745–760, 2016.

[444] Marcelo Alejandro Pereyra, Philip Schniter, Emilie Chouzenoux, Jean-Christophe Pesquet, Jean-Yves Tourneret, Alfred O. Hero, and Steve McLaughlin. A survey of stochastic simulation and optimization methods in signal processing. *IEEE Journal on Selected Topics in Signal Processing*, 10(2):224–241, 2015.

[445] Giorgio Pini and Giuseppe Gambolati. Is a simple diagonal scaling the best preconditioner for conjugate gradients on supercomputers? *Advances in Water Resources*, 13(3):147–153, 1990.

[446] Luisa F Polania, Rafael E Carrillo, Manuel Blanco-Velasco, and Kenneth E Barner. Exploiting prior knowledge in compressed sensing wireless ecg systems. *IEEE journal of Biomedical and Health Informatics*, 19(2):508–519, 2014.

[447] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *Journal of the Royal Statistical Society, Series B (Methodological)*, 82(3):601–627, 2020.

[448] Zhaonan Qu, Yinyu Ye, and Zhengyuan Zhou. Diagonal preconditioning: Theory and algorithms. *arXiv:2003.07545*, 2020.

[449] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling I: algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016.

[450] Kent Quanrud. Approximating optimal transport with linear programs. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 6:1–6:9, 2019.

[451] Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv e-prints*, abs/1908.05659, 2019.

[452] Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3066–3074, 2013.

[453] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Restricted eigenvalue properties for correlated gaussian designs. *The Journal of Machine Learning Research*, 11:2241–2259, 2010.

[454] Oded Regev and Aravindan Vijayaraghavan. On learning mixtures of well-separated gaussians. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 85–96. IEEE, 2017.

[455] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.

[456] Philippe Rigollet and Jan-Christian Hütter. *High-Dimensional Statistics*. 2017.

[457] Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[458] R.T̃yrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[459] R Tyrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[460] R Tyrrell Rockafellar. Monotone operators associated with saddle-functions and minimax problems. *Nonlinear functional analysis*, 18(part 1):397–407, 1970.

[461] Noah A. Rosenberg, Jonathan K. Pritchard, James L. Weber, Howard M. Cann, Kenneth K. Kidd, Lev A. Zhivotovsky, and Marcus W. Feldman. Genetic structure of human populations. *Science*, 298:2381–2385, 2002.

[462] Mark Rudelson and Roman Vershynin. Sparse reconstruction by convex relaxation: Fourier and gaussian measurements. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 207–212. IEEE, 2006.

[463] Mark Rudelson and Roman Vershynin. The smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, 62(12):1707–1739, 2009.

[464] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11(1):1–96, 2018.

[465] Yousef Saad. Analysis of some krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 29(1):209–228, 1992.

[466] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014.

[467] Adil Salim, Dmitry Koralev, and Peter Richtárik. Stochastic proximal langevin algorithm: Potential splitting and nonasymptotic rates. In *Advances in Neural Information Processing Systems*, pages 6653–6664, 2019.

[468] Piotr Sankowski. Maximum weight bipartite matching in matrix multiplication time. *Theor. Comput. Sci.*, 410(44):4480–4488, 2009.

[469] Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

[470] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859*, 2020.

[471] Ludwig Schmidt. *Algorithms above the noise floor*. PhD thesis, Massachusetts Institute of Technology, 2018.

[472] Mark W. Schmidt, Nicolas Le Roux, and Francis R. Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1-2):83–112, 2017.

[473] Shai Shalev-Shwartz. Online learning: Theory, algorithms, and applications. PhD thesis, Hebrew University, 2007.

[474] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[475] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

[476] Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the maximal loss: How and why. In *ICML*, pages 793–801, 2016.

[477] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *J. Mach. Learn. Res.*, 14(1):567–599, 2013.

[478] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Math. Program.*, 155(1-2):105–145, 2016.

[479] R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 385–394, 2012.

[480] Ruoqi Shen and Yin Tat Lee. The randomized midpoint method for log-concave sampling. In *Advances in Neural Information Processing Systems*, pages 2100–2111, 2019.

[481] Ruoqi Shen, Kevin Tian, and Yin Tat Lee. Composite logconcave sampling with a restricted gaussian oracle. *CoRR*, abs/2006.05976, 2020.

[482] Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269, 2013.

[483] Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 452–460. ACM, 2017.

[484] Zhan Shi, Xinhua Zhang, and Yaoliang Yu. Bregman divergence for stochastic variance reduction: Saddle-point and adversarial prediction. In *Advances in Neural Information Processing Systems*, 2017.

[485] Ralph E. Showalter. Monotone operators in banach space and nonlinear partial differential equations. *Mathematical Surveys and Monographs*, 49:162–163, 1997.

[486] Aaron Sidford and Kevin Tian. Coordinate methods for accelerating ℓ_∞ regression and faster approximate maximum flow. In *59th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2018, 7-9 October, 2018, Paris, France*, 2018.

[487] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. Society for Industrial and Applied Mathematics, 2018.

[488] Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[489] Gary Smith. *Essential Statistics, Regression, and Econometrics*. Academic Press, 2012.

[490] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas J. Guibas. Convolutional wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, 2015.

[491] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM SIGACT Symposium on Theory of Computing*, volume 4, 2004.

[492] Jacob Steinhardt. *Robust Learning: Information Theory and Algorithms*. PhD thesis, Stanford University, 2018.

[493] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. Resilience: A criterion for learning in the presence of arbitrary outliers. *arXiv preprint arXiv:1703.04940*, 2017.

[494] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3517–3529, 2017.

[495] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Advances in Neural Information Processing Systems*, pages 4439–4447, 2016.

[496] Fedor Stonyakina, Alexander Tyurin, Alexander Gasnikov, Pavel Dvurechensky, Artem Agafonov, Darina Dvinskikh, Dmitry Pasechnyuk, Sergei Artamonov, and Victorya Piskunova. Inexact relative smoothness and strong convexity for optimization and variational inequalities by inexact model. *arXiv e-prints*, abs/2001.09013, 2020.

[497] Thomas Strohmer and Roman Vershynin. A randomized solver for linear systems with exponential convergence. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 499–507, 2006.

[498] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

[499] Conghui Tan, Tong Zhang, Shiqian Ma, and Ji Liu. Stochastic primal-dual method for empirical risk minimization with $o(1)$ per-iteration complexity. In *Advances in Neural Information Processing Systems*, 2018.

[500] Robert Endre Tarjan. *Data structures and network algorithms*, volume 44 of *CBMS-NSF regional conference series in applied mathematics*. SIAM, 1983.

[501] Kiran Koshy Thekumparampil, Niao He, and Sewoong Oh. Lifted primal-dual method for bilinearly coupled smooth minimax optimization. In *25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[502] Kiran Koshy Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Projection efficient subgradient method and optimal nonsmooth frank-wolfe method. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[503] Kevin Tian, Weihao Kong, and Gregory Valiant. Learning populations of parameters. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5778–5787, 2017.

[504] Kevin Tian, Teng Zhang, and James Zou. Cover: Learning covariate-specific vector representations with tensor decompositions. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4933–4942. PMLR, 2018.

[505] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[506] Sumedh Tirodkar. Deterministic algorithms for maximum matching on general graphs in the semi-streaming model. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, pages 39:1–39:16, 2018.

[507] Vladislav Tominin, Yaroslav Tominin, Ekaterina Borodich, Dmitry Kovalev, Alexander Gasnikov, and Pavel Dvurechensky. On accelerated saddle-point problems with composite structure. *arXiv e-prints*, abs/2103.09344v2, 2021.

[508] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018.

[509] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. Society for Industrial and Applied Mathematics, USA, 2012.

[510] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

[511] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

[512] John W Tukey. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, pages 448–485, 1960.

[513] John W Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, pages 523–531, 1975.

[514] Sara Van de Geer and Johannes Lederer. The lasso, correlated design, and improved oracle inequalities. In *From Probability to Statistics and Back: High-Dimensional Models and Processes–A Festschrift in Honor of Jon A. Wellner*, pages 303–316. Institute of Mathematical Statistics, 2013.

[515] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 259–278. SIAM, 2020.

[516] Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 859–869, 2021.

[517] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 775–788. ACM, 2020.

[518] A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14(1):14–23, 1969.

[519] Santosh Vempala. Geometric random walks: A survey. *MSRI Combinatorial and Computational Geometry*, 52:573–612, 2005.

[520] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

[521] Santosh S. Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8092–8104, 2019.

[522] Roman Vershynin. *High-Dimensional Probability, An Introduction with Applications in Data Science*. 2016.

[523] François-Xavier Vialard. An elementary introduction to entropic regularization and proximal methods for numerical optimal transport, 2019.

[524] Eric Vittinghoff, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer, 2005.

[525] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 1944.

[526] Michael D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.

[527] David Wajc. Rounding dynamic matchings against an adaptive adversary. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 194–207, 2020.

[528] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.

[529] Jialei Wang and Lin Xiao. Exploiting strong convexity from data with primal-dual first-order algorithms. In *34th International Conference on Machine Learning (ICML)*, pages 3694–3702, 2017.

[530] Jun-Kun Wang and Jacob D. Abernethy. Acceleration through optimistic no-regret dynamics. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 3828–3838, 2018.

[531] Mengdi Wang. Primal-dual π learning: Sample complexity and sublinear run time for ergodic markov decision problems. *CoRR*, abs/1710.06100, 2017.

[532] Mengdi Wang. Randomized linear programming solves the markov decision problem in nearly linear (sometimes sublinear) time. *Math. Oper. Res.*, 45(2):517–546, 2020.

[533] Yuanhao Wang and Jian Li. Improved algorithms for convex-concave minimax optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 4800–4810, 2020.

[534] Manfred K. Warmuth and Dima Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1481–1488, 2006.

[535] Manfred K. Warmuth and Dima Kuzmin. Online variance minimization. *Machine Learning*, 87(1):1–32, 2012.

[536] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[537] Andre Wibisono. Proximal langevin algorithm: Rapid convergence under isoperimetry. *CoRR*, abs/1911.01469, 2019.

[538] Sławomir T Wierzchoń and Mieczysław A Kłopotek. *Modern algorithms of cluster analysis*. Springer, 2018.

[539] Blake E. Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 3639–3647, 2016.

[540] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[541] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 538–546, 2001.

[542] Yao-Liang Yu. The strong convexity of von neumann's entropy. `http://www.cs.cmu.edu/~yaoliang/mynotes/sc.pdf`, 2013.

[543] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1188–1196, 2017.

[544] Chicheng Zhang and Yinan Li. Improved algorithms for efficient active learning halfspaces with massart and tsybakov noise. *arXiv preprint arXiv:2102.05312*, 2021.

[545] Junyu Zhang, Minyi Hong, and Shuzhong Zhang. On lower iteration complexity bounds for the saddle point problems. *arXiv e-prints*, abs/1912.07481, 2019.

[546] Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Optimal prediction for sparse linear models? lower bounds for coordinate-separable m-estimators. *Electronic Journal of Statistics*, 11(1):752–799, 2017.

[547] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *J. Mach. Learn. Res.*, 18:84:1–84:42, 2017.

[548] Zhimin Zhang, Shoushui Wei, Dingwen Wei, Liping Li, Feng Liu, and Chengyu Liu. Comparison of four recovery algorithms used in compressed sensing for ecg signal processing. In *2016 Computing in Cardiology Conference (CinC)*, pages 401–404. IEEE, 2016.

[549] Kaiwen Zhou, Qinghua Ding, Fanhua Shang, James Cheng, Danli Li, and Zhi-Quan Luo. Direct acceleration of SAGA using sampled negative momentum. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1602–1610. PMLR, 2019.

[550] Banghua Zhu, Jiantao Jiao, and Jacob Steinhardt. Robust estimation via generalized quasi-gradients. *CoRR*, abs/2005.14073, 2020.

[551] Zeyuan Allen Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1606–1614, 2016.

[552] Zeyuan Allen Zhu, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 890–901, 2017.

[553] Zeyuan Allen Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *33rd International Conference on Machine Learning (ICML)*, pages 1110–1119, 2016.

[554] Difan Zou, Pan Xu, and Quanquan Gu. Subsampled stochastic variance-reduced gradient langevin dynamics. In *International Conference on Uncertainty in Artificial Intelligence*, 2018.

[555] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.