From Enormous Structured Models to On-device Federated Learning: Robustness, Heterogeneity and Optimization

Krishna Pillutla

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2022

Reading Committee:

Zaid Harchaoui, Co-Chair
Sham Kakade, Co-Chair
Kevin Jamieson
Jamie Morgenstern

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science and Engineering

© Copyright 2022

Krishna Pillutla

University of Washington

Abstract

From Enormous Structured Models to On-device Federated Learning:

Robustness, Heterogeneity and Optimization

Krishna Pillutla

Co-chairs of the Supervisory Committee:

Zaid Harchaoui

Department of Statistics

Sham Kakade

Paul G. Allen School of Computer Science & Engineering and Department of Statistics

Artificial intelligence has been shaped by three revolutions in recent years: (1) differentiable programming, the practice of writing programs by chaining parameterized modules and learning these parameters from data, (2) an explosion of scale, where increasing model size consistently improves performance, and (3) federated learning, where model training moves to mobile devices, where the data is generated and resides. This dissertation presents diagnostic methods and new algorithms to measure and improve the robustness of machine learning models to heterogeneous operating circumstances across these revolutions.

Differentiable programming and end-to-end learning from examples are challenged by applications that require combinatorial algorithms as a part of their computations. We overcome this problem with smoothed versions of combinatorial algorithms and rigorously show how to construct them from the top-K best outputs. We leverage this smoothing to propose a family of accelerated optimization algorithms for structured prediction problems.

Further, enormous language models have recently gained the ability to compose clear and coherent essays that are up to several hundred words long. We propose a comparison tool that directly measures how

close the distribution of generated text is to that of human-written text. We show experimentally that the proposed measure correlates the strongest with human evaluations of machine text and can quantify many qualitative properties of machine-generated text, such as the effect of model size and decoding algorithms.

Finally, the move to massively distributed on-device federated learning of models gives rise to new challenges due to the natural heterogeneity of underlying user data and privacy requirements of model aggregation. We propose a federated learning method that is robust to corrupted updates sent by malicious users and proves effective where traditional outlier detection or filtering methods are not applicable due to privacy requirements. We also propose another federated learning method that improves performance for users who do not conform to population trends. In both cases, we introduce federated optimization algorithms that are both communication efficient and privacy-preserving.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisors, Zaid Harchaoui and Sham Kakade. I am grateful to Sham for taking a chance on me at the start of my PhD. His deep intuition for highly technical matters, reflected in incisive questions to get to the heart of any problem and the creative solutions he subsequently suggests, has had a formative effect on my way of thinking. Zaid has been a *guru* to me in every respect — he has helped me grow not just scientifically and professionally, but also personally and philosophically. His passion for research, endless reserves of deep ideas, and drive to persevere have been a great source of motivation for me. Together, Zaid and Sham provided me with a comprehensive education and steadfastly supported me through thick and thin. I continue to be inspired by their clarity of thought and how they work to constantly broaden their horizons across different areas.

I have been fortunate to work closely with Yejin Choi and Jérôme Malick through various stages of my PhD. Our regular interactions have led me to be more open-minded and think big picture while not losing track of the details. I am grateful to my other committee members — Kevin Jamieson, Jamie Morgenstern, and Lalit Jain — for their detailed feedback and career advice. I would also like to thank the other machine learning faculty — Maryam Fazel, Sewoong Oh, and Ludwig Schmidt — for creating a stellar research environment at UW.

I would like to thank my mentors for two wonderful internships at Facebook AI Research: Awni Hannun and Gabriel Synnaeve in the first one and Lin Xiao in the second one. Each internship has been pivotal to my growth as a researcher — the first one for learning to perform real applied research and the second for drawing inspiration for research questions from applied problems.

I wish to thank my academic siblings for making the daily grind an enjoyable journey: Rahul Kidambi, John Thickstun, Aravind Rajeswaran, Corinne Jones, Alec Greaves-Tunnell, Yassine Laguel, Vincent Roulet, Lang Liu, Ramya Vinayak, Aditya Kusupati, Motoya Ohnishi, Weihao Kong, Nick Irons, Kristof Glauninger, Joshua Cutler, Jillian Fisher, and Ronak Mehta. I could not have asked for better peers than John and Lang — I have learned so much from observing, interacting with, and simply being around them on a daily basis. A special thanks to Yassine, who has been a great collaborator and an even better friend. I would also like to acknowledge the role of Vincent Roulet and Swabha Swayamdipta in mentoring me, from feedback on talks and statements to sharing their academic and professional experiences. It has been extremely rewarding to work with Jillian and Ronak — their enthusiasm is contagious and their work ethic is inspiring.

I owe my productivity during the pandemic to the people of the Padelford bubble — Lang, Vincent, Ema, and Zaid. They kept me grounded and allowed me to stay focused. The sense of camaraderie we shared made the pandemic a little less difficult and I have fond memories of our time together. I have also been incredibly fortunate to have Rahul, John, and Vincent in my corner. Their support on so many levels — technical, emotional, and personal — helped me get through the toughest stretches of the PhD journey. Finally, my sincere thanks to Elise Dorough and the graduate advising team, who have helped me and countless other students get through graduate school.

I thank all my co-authors (in chronological order): Sham Kakade, Rahul Kidambi, Zaid Harchaoui, Vincent Roulet, Yassine Laguel, Jérôme Malick, Yejin Choi, Swabha Swayamdipta, John Thickstun, Lang Liu, Rowan Zellers, Sean Welleck, Sewoong Oh, Aditya Kusupati, Lin Xiao, Mike Rabbat, Abdel-rahman Mohamed, Maziar Sanjabi, Kshitiz Malik, Jillian Fisher, Ronak Mehta.

I have been fortunate to share this journey with some fantastic individuals who have become dear friends — Walter Cai, Swati Padmanabhan, Romain Camilleri, and Sam Ainsworth to name a few. The UW ML group, particularly Jennifer Brennan, Ian Covert, and Andrew Wagenmaker made the ML lab a wonderful home. I have fond memories of jam sessions with CSE band mates Esther Jang, Dave Wadden, and others. I also acknowledge my badminton buddies (Lang, Vincent, Gang, Zhaoqi), squash buddies (Ian, Mitchell, Aaron), pandemic soccer (+ gossip) buddies (Romain, Swati, Andrew, Pascal, Oscar), and my trainer Dante for adding fun interludes to graduate school. I am grateful to Zeba Islam, Anna Jeszeck, Morelle Arian, Srini Iyer for their friendship and for the good times. I also appreciate my housemates, notably Josh Katz and Vidur Joshi, for making our house a home and a true place of refuge.

I am grateful to all my teachers and mentors for making me the person I am today. I would particularly like to thank Dhruv Mahajan and Sundararajan Sellamanickam for introducing me to machine learning during an undergraduate internship, and Saketha Nath and Nina Balcan for guiding me on my initial forays into research. I also acknowledge the role of Dr. Edwin, my college basketball coach, in building my character. I still carry the valuable lessons they have taught me.

Above all, I would like to thank my family: my parents, the kindest, most selfless and most hard-working people I know; my sister and also my *favoritest* person in the whole world, Ramya, for being an inextinguishable source of encouragement and inspiration; my brother-in-law, Karthik, the epitome of patience and virtue; *Tata*, whose affection and strength of character are second to none; *Ammamma*, for teaching me how to appreciate the finer things in life (and who I miss every single day); both my *Pinni*s for being only a phone call and a flight away, and whose care and love rival my mother's; all my grandparents, cousins, aunts, and uncles, for supporting my every endeavor. Lastly and most importantly, I am forever grateful to my wonderful wife, Parul, for being my greatest cheerleader and for always being there for me. All of their sacrifices and hard work have allowed me to pursue my dreams — this PhD is their achievement as much as it is mine.

Funding from the National Science Foundation, the J.P. Morgan PhD Fellowship and the Anne Dinning - Michael Wolf Endowed Regental Fellowship is gratefully acknowledged.

DEDICATION

To *Amma* and *Nanna*,
My two greatest role models.

Contents

1 Introduction **11**
- 1.1 The Three Revolutions . . . 11
- 1.2 Robustness, Heterogeneity, and Optimization: The Challenges . . . 15
- 1.3 Motivation, Contributions, and Outline . . . 16

2 Casimir: A Smoother Way to Train Structured Prediction Models **19**
- 2.1 Background . . . 20
- 2.2 Smooth Structured Prediction . . . 24
- 2.3 Inference Oracles . . . 28
- 2.4 Implementation of Inference Oracles . . . 30
- 2.5 Smoothable Convex Optimization: the Casimir Algorithm . . . 38
- 2.6 Experiments . . . 44
- 2.7 Related Work . . . 49
- 2.8 Discussion and Conclusion . . . 53

3 The Trade-offs of Incremental Linearization Algorithms for Nonsmooth Composite Problems **57**
- 3.1 Background . . . 58
- 3.2 Tradeoffs of the Prox-linear Method in Statistical Settings . . . 60
- 3.3 The Prox-linear Method with Incremental Inner Loops . . . 62
- 3.4 Experiments . . . 68

4 MAUVE: Measuring the Gap Between Neural Text and Human Text **73**
- 4.1 Background . . . 74
- 4.2 MAUVE . . . 76
- 4.3 Related Work . . . 82
- 4.4 Experiments . . . 84
- 4.5 Discussion and Conclusion . . . 90

5 RFA: Robust Aggregation for Federated Learning **91**
- 5.1 Background: Federated Learning . . . 92
- 5.2 Problem Setup: Federated Learning with Corruptions . . . 98
- 5.3 Robust Aggregation and the RFA Algorithm . . . 101
- 5.4 Experiments . . . 108
- 5.5 Related Work . . . 112
- 5.6 Discussion and Conclusion . . . 113

6 Δ-FL: A Superquantile Optimization Approach for Federated Learning with Heterogeneous Devices **115**
6.1 Background . . . 117
6.2 Problem Setup . . . 119
6.3 Handling Heterogeneity with Δ-FL . . . 121
6.4 Theoretical Analysis . . . 124
6.5 Discussion . . . 131
6.6 Related Work . . . 132
6.7 Experiments . . . 133
6.8 Discussion and Conclusion . . . 137

7 Conclusion **139**

References **140**

A Casimir: Full Proofs **169**
A.1 Smoothing Basics . . . 169
A.2 Inference Oracles with Branch and Bound Search . . . 172
A.3 Smoothness of Related Methods . . . 173
A.4 SPIGOT: Analytical Form . . . 174
A.5 Casimir: Convergence Proofs . . . 177
A.6 Prox-Linear with Incremental Gradient Methods: Convergence Proofs . . . 186
A.7 Local Quadratic Convergence of the Prox-Linear Method . . . 192

B MAUVE: Proofs and Experimental Details **195**
B.1 Proof of the Statistical Bound . . . 195
B.2 Experimental Details . . . 197
B.3 Additional Experimental Results . . . 201
B.4 Human Evaluation Details . . . 206
B.5 Human Evaluation: Protocol and Full Results . . . 206

C RFA: Experimental Details **215**
C.1 Experimental Details . . . 215
C.2 Additional Numerical Results . . . 220
C.3 Numerical Results: Convergence of The Smoothed Weiszfeld Algorithm . . . 223

D Δ-FL: Proofs and Experimental Details **225**
D.1 Convergence Analysis . . . 225
D.2 Privacy Analysis . . . 240
D.3 Numerical Experiments: Complete Results . . . 243

Chapter 1

Introduction

Artificial intelligence is witnessing breakthroughs in diverse applications, ranging from natural language processing, computer vision, and speech recognition to protein structure prediction and vaccine development (LeCun et al., 2015; Hannun et al., 2014; Jumper et al., 2021; Arshadi et al., 2020). These advances are being ushered in by a resurgence of deep neural networks, and modern machine learning models are enabling artificial agents to master challenging tasks, from playing the complex strategic game of Go (Mnih et al., 2015; Silver et al., 2016) to robot-aided surgery (Kassahun et al., 2016).

The resurgence of neural networks and their rapid rise to ubiquity in everyday life has been driven by three revolutions: differentiable programming, a massive explosion of scale, and federated learning. The first revolution enabled deep models to be trained with a variety of modular components in an end-to-end fashion using automatic differentiation and first-order stochastic optimization. Driven by increased hardware capacity and unprecedented availability of web-scale data, the second revolution yielded new and surprising emergent capabilities of machine learning models and striking improvements in existing abilities to match or even exceed human performance for a variety of tasks. The third revolution, federated learning, has moved machine learning from the cloud to mobile phones or other decentralized entities, allowing users to leverage their collective statistical power without explicitly sharing sensitive data.

In the rest of this chapter, we illustrate the paradigm shifts brought in by these three revolutions through the common lens of language modeling. We then describe the emerging challenges of the robustness of modern machine learning systems in heterogeneous operating environments. We conclude this chapter with a summary of the chapter-wise contributions of this dissertation in tackling these issues.

1.1 The Three Revolutions

We highlight the impact of each revolution by examining the task of language modeling. Emerging in the 1970s as an important component of speech recognition systems (Baker, 1975; Jelinek, 1976), language modeling has been a major driver of progress in natural language processing in recent years (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020). Given a previous context of natural language tokens (which could be words or parts of words), a language model aims to "guess" the next token. Formally, given a sequence $y_1, \ldots, y_t$ of tokens, a language model P assigns a probability $P(\cdot \,|y_1, \ldots, y_t)$ over the next token y_{t+1}.

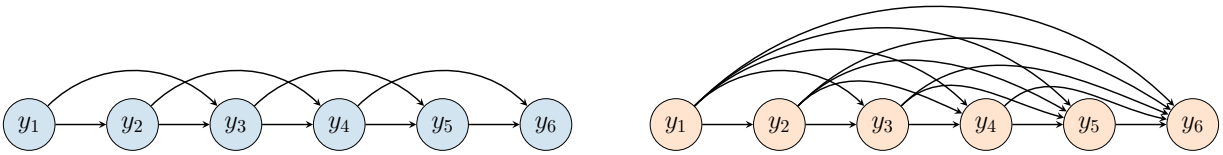


Figure 1.1: Graphical models depicting the temporal dependence assumptions behind (**left**) a 3-gram language model, and (**right**) modern neural language models based on recurrent neural networks and transformers.

1.1.1 Differentiable Programming

Differentiable programming refers to the practice of constructing a program by chaining together generic parameterized functional blocks, which are trained to exhibit the desired behavior from input-output examples using gradient-based optimization. This approach evolved from the realization that the techniques employed to train neural networks, namely stochastic gradient optimization with reverse-mode automatic differentiation (Baydin et al., 2017), also known as backpropagation, could be used to optimize arbitrary compositions of differentiable functions.

To illustrate the impact of this revolution on language modeling, consider that the dominant approach to language modeling, from its inception in the 1970s until around 2012, was based on n-grams (e.g., Jurafsky and Martin, 2009, Chap. 4). The n-gram model imposes a Markovian assumption of order $n-1$, as depicted in Figure 1.1 (left) and expressed as

$$P(y_{t+1} \,|\, y_1, \cdots, y_t) = P(y_{t+1} \,|\, y_{t-n+2}, \cdots, y_t)\,,$$

building upon the seminal works of Markov (1913) and Shannon (1948). Here, the model is simply a transition matrix, which can be estimated by counting n-grams. The first neural language model of Bengio et al. (2003) still held the Markovian assumption, but it was parameterized by a non-linear model defined by a chain of 5 compositions. This assumption was relaxed by Mikolov et al. (2010), who used recurrent neural networks (Hopfield, 1982; Rumelhart et al., 1986); see also Figure 1.1 (right). Here, each time step included 5 compositions, but the chain now extended over the time dimension.

These chains of computations gradually grew more complex (Sundermeyer et al., 2015), as illustrated in Figure 1.2. The next dominant architecture was based on long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997); each LSTM cell contained 7 successive compositions per time-step. Similar to RNNs, the computation graph extended over the time dimension. Typical models contained 2-5 LSTM layers stacked atop each other (Peters et al., 2018). As of 2022, the dominant architecture for language models is the transformer (Vaswani et al., 2017). A transformer block is a chain of 10 compositions, where all operations are performed in parallel across time. Transformer language models run quite deep, from 12 layers for BERT to 96 for GPT-3.

The overwhelming empirical success of differentiable programming leads to a natural question: can it be extended to handle problems that are not compositions of elementary differentiable functions? Examples of such problems include outputs of combinatorial algorithms and nonlinear control (Roulet et al., 2019), solutions to continuous optimization problems (Domke, 2012; Amos, 2019), and computations including stochastic operations (Schulman et al., 2015) or partial differential equations (Chen et al., 2018c; Ainsworth et al., 2021). These settings require us to define appropriate surrogate objectives or gradient estimators and show how to implement these algorithms in a computationally efficient manner. Additional challenges include analyzing their usefulness and reasoning about the convergence of the resulting

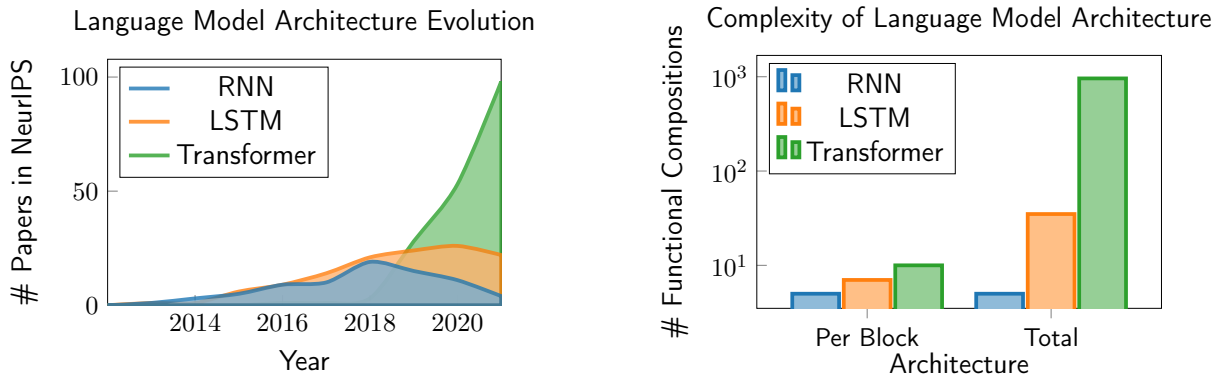


Figure 1.2: Evolution in the complexity of language modeling architectures over the decade 2012-2022. **Left**: The number of papers published in the NeurIPS conference each year that used RNN/LSTM/transformer language models. Papers with the phrases "language model" and "RNN" (resp. "LSTM" and "transformer") at least thrice (case-insensitive) were included in the count. **Right**: The number n of functions $f_1, f_2, \cdots, f_n$ whose composition $f = f_n \circ \cdots \circ f_1$ makes up a RNN/LSTM/transformer block, as well as the total number of functional compositions in such a language model. Specifically, we give the counts for the RNN language model of Mikolov et al. (2010), the LSTM language model ELMo (Peters et al., 2018), and the transformer language model GPT-3 (Brown et al., 2020). Each component f_i is either a linear map (with no two consecutive f_i, f_{i+1} both linear) or simple non-linear maps, such as component-wise activations, the softmax function, a product of gates in the LSTM, or layer normalization.

stochastic first-order optimization algorithms.

1.1.2 Explosion of Scale

The differentiable programming revolution was immediately followed by the observation that deeper models (i.e., longer chains of compositions) with more data consistently improved performance not just in natural language processing, but also in computer vision (Dosovitskiy et al., 2021), speech processing (Hsu et al., 2021), and multi-modal learning (Radford et al., 2021). It was demonstrated that these enormous models exhibited many remarkable emergent properties for which they were not trained (Bommasani et al., 2021). For instance, the GPT-2 model (Radford et al., 2019) wrote a long and high-quality essay of text in response to a prompt although it was not trained to generate text. The in-context learning of GPT-3 (Brown et al., 2020), which can be adapted to downstream tasks simply with their natural language description as a prompt, is another striking example.

Figure 1.3 illustrates the effect of scaling up the model size for language modeling. Recent scaling studies (Kaplan et al., 2020) show that one attains consistent improvements not just in language modeling but also in downstream tasks across all these orders of magnitude of model size. For instance, the test loss L for language modeling itself scales with the model size M as

$$L = a - b \log M \,,$$

where $a = 4.33$ and $b = 0.28$; this is illustrated in Figure 1.3 (left). Figure 1.3 (center) shows how model sizes have evolved over the last four years: starting from ELMo and BERT, with 93.6 and 117 million parameters respectively, to GPT-3, with 175 billion parameters, and Wu Dao 2.0, with a whopping 1.75 trillion parameters (wud, 2021). Likewise, the rightmost plot of Figure 1.3 shows that the amount of data

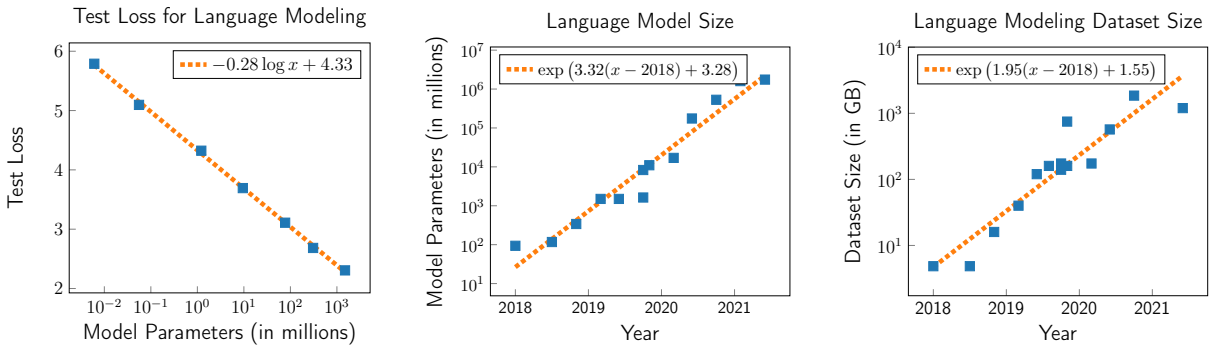


Figure 1.3: Scaling of neural language models over 2018-2022. **Left**: As the model size increases across orders of magnitude, the performance of the model at language modeling, as measured by the cross entropy on unseen data, consistently improves. The numbers are taken from (Kaplan et al., 2020). **Center & Right**: The size of the recent neural language models (center) and the amount of data they were trained on (right) have consistently increased across orders of magnitude.

used to train these models has likewise exploded: starting from about 5 gigabytes of text for ELMo and BERT to 570 gigabytes for GPT-3 and 1.8 terabytes for Megatron Turing-NLG (Smith et al., 2022).

To summarize, large-scale models demonstrate emergent behavior, but there is little understanding of the mechanism behind this phenomenon. This calls for more research on diagnostic measures as well as methodologies to understand and channel the emergent behavior for downstream tasks.

1.1.3 Federated Learning

A driving force behind the success of differentiable programming is the widespread access to data. The amount of data generated by interactions between users and smart devices has exploded with the proliferation of mobile phones, wearables, and smart appliances – an estimated 6.6 billion people use smartphones worldwide. Rich user data generated by mobile phones, sensors, or organizations (referred to as *clients*) are inherently privacy-sensitive. Laws such as the European Union General Data Protection Regulation (GDPR) (2016) aim to minimize privacy risks associated with the transfer of such data. *Federated learning*, where various clients collaboratively train a model while keeping their data fully decentralized (i.e., at the client), aims to overcome the data-access barrier.

To illustrate how federated learning changes the usual machine learning pipeline, consider the task of predicting the next word on a mobile keyboard. Given a context $y_1, \ldots, y_t$, a standard approach is to predict the word $\arg\max_{y'} P(y' \mid y_1, \cdots, y_t)$ with the highest probability under a language model P. The prevailing GBoard model prior to the advent of federated learning was a n-gram language model trained on data available at the server (Ouyang et al., 2017). A data-hungry deep network such as an LSTM or transformer requires access to a much larger amount of data; federated learning makes this possible (Hard et al., 2018). In addition, the language models are trained with cryptographic secure multi-party computation (Bonawitz et al., 2019) and differential privacy (Kairouz et al., 2021a) to give rigorous privacy guarantees.

To summarize, federated learning calls for novel algorithmic solutions to emerging problems while respecting communication constraints of mobile devices and providing rigorous privacy guarantees on the sensitive user data accessed.

1.2 Robustness, Heterogeneity, and Optimization: The Challenges

The three revolutions – differentiable programming, the explosion of scale, and federated learning – have greatly expanded the capabilities and applicability of machine learning models, which continue to grow in popularity. Across each of these paradigm shifts, however, machine learning systems remain brittle and exhibit poor robustness when exposed to heterogeneous operating conditions. For example, chatbot Tay posted a number of inflammatory tweets (Metz, 2018), and modern language models have been shown to produce racist, sexist, and toxic text (Gehman et al., 2020). Some failures also pose a direct threat to human life, such as self-driving cars injuring or killing pedestrians (Knight, 2018).

A typical example is the lack of robustness of machine learning systems to "outlier" data at training time, i.e., data that takes values outside its nominal range. Machine learning pipelines usually include non-robust operations such as averaging losses or gradients over a minibatch, where an outlier can have an outsized effect on the resulting aggregate. This can be formalized by the notation of the breakdown point of an estimator (Donoho and Huber, 1983), which is the fraction of data that must be replaced before the estimator takes arbitrary values; the arithmetic mean has a breakdown point of zero, meaning that is susceptible to a single outlier. Further, the massive amounts of data that enormous models are pretrained on are not always released to the public (e.g., Radford et al., 2019; 2021); the lack of established conventions for data release and sanitization coupled with the tendency of massive models to memorize training data (Carlini et al., 2021) can lead to undesirable behavior of the resulting model, including on downstream tasks it is finetuned for (e.g., Schuster et al., 2021). Similarly, in federated learning, this problem is greatly exacerbated by privacy requirements, which stipulate that the data on, or the updates sent by (potentially compromised) clients cannot be inspected by the service provider. Across all these settings, solutions to address robustness are complicated by the presence of data heterogeneity; apparently adversarial outliers might really be manifestations of heterogeneous inputs.

Machine learning systems also exhibit a lack of robustness at deployment time. Supervised learning usually assumes that training and test data are drawn from the same distribution. Real data is, however, heterogeneous, resulting in a violation of this assumption when models are deployed in the real world. In federated learning, for instance, a model is trained on the average population distribution but deployed on individual clients that might differ from the population. More generally, machine learning models have been found to perform poorly on subgroups of data whose distribution is shifted relative to the training distribution. This can have massive societal consequences when the deployed model exhibits poor performance on, for instance, minority groups in terms of ethnicity, race, religion, etc. (Buolamwini and Gebru, 2018; Sap et al., 2019). Enormous pretrained models with the same in-distribution performance have been found to exhibit a large variance in performance on out-of-distribution data due to factors as innocuous as a random seed (D'Amour et al., 2020). This is also true when finetuning the model on downstream tasks.

Yet another source of mismatch between deployment and training conditions is the emergent behavior from massively scaling up the model size. For instance, while GPT-3 is trained for the task of language modeling, it can be deployed for a range of natural language tasks by leveraging its emergent in-context learning. Here, we include a text description of a new task and a few examples in the prompt, and the resulting output of the model looks like the model had "learned" this new task. This emergent behavior can fail to be robust to different deployment conditions. For example, small changes in the text of the prompt for GPT-3, or in the order of the in-context examples can lead to vastly different model behavior (Scao and Rush, 2021; Lu et al., 2022).

These problems highlight a need for novel diagnostic methods that can quantify the robustness or lack thereof and the extent and type of heterogeneity, as well as training time solutions that build greater

robustness into the model. Solutions of the latter kind often involve nonsmooth objective formulations, which suffer from slow convergence relative to the usual (non-robust) approach. Nonsmooth optimization problems also show up when considering extensions of differentiable programming to handle combinatorial algorithms or solutions to other optimization problems. The difficulties associated with slow convergence are compounded by the need to tune additional smoothing hyperparameters and the lack of unbiased stochastic gradient estimators in some formulations.

1.3 Motivation, Contributions, and Outline

This thesis addresses specific aspects of robustness and optimization of machine learning models to heterogeneous environments across each of these three revolutions. It contains five chapters and a conclusion.

1.3.1 Contributions and Dissertation Structure

Chapters 2 and 3: Differentiable Programming and Optimization of Deep Structured Models

Combinatorial algorithms like dynamic programming play an important role in natural language processing and computer vision. This setting is collectively referred to as structured prediction since it involves the forecast of highly structured objects such as sequences, trees, and lattices. More broadly, combinatorial algorithms are key to pipelines with a search step, with examples ranging from robotics and simulators to symbolic reasoning and formal logic. The paradigm of differentiable programming and end-to-end learning from examples is challenged by applications that require seemingly incompatible computations such as combinatorial algorithms. Further, typical approaches to learning structured prediction models rely on nonsmooth optimization algorithms that suffer from slow convergence.

Contributions. In Chapter 2, we investigate smoothed versions of combinatorial algorithms (Pillutla et al., 2018) in order to make them compatible with differentiable programming. We rigorously show how to construct smoothed versions of combinatorial algorithms from the top-K best outputs; these K-best outputs can be computed efficiently with extensions of the combinatorial routines. We leverage this smoothing to propose a family of accelerated optimization algorithms for structured prediction problems. This framework, called Casimir, exhibits the optimal rate of $n + \sqrt{n/\lambda\varepsilon}$ (up to constants and polylog factors) to find an ε-accurate solution for non-smooth and λ-strongly convex problems written as an average of n functions, while automatically tuning the smoothing parameter. We demonstrate faster optimization and superior task performance compared to the previous state-of-the-art.

In Chapter 3, we study extensions of Casimir based on the prox-linear approach that make it applicable to deep learning (Pillutla et al., 2018; 2022c). While such algorithms have been found to exhibit rapid local convergence, we formalize an example in the statistical setting where rapid local convergence is not reached before reaching the problem's noise level. Empirically, we delineate a regime in terms of the signal-to-noise ratio of the problem where the stochastic subgradient method performs better than approaches based on the prox-linear algorithm.

Chapter 4: Diagnosing Text Generation Models with MAUVE

A property of enormous language models that emerges from the explosion of scale is open-ended text generation: models can write remarkably fluent and grammatical essays of text in response to a prompt (gpt,

2019). Unlike previous generations of models, contemporary models do not commit obvious errors of repetition and syntax; instead, they make subtler errors in semantics, narration, or continuity. Though the capability of open-ended generation is recent, it has already been widely deployed commercially in the form of artificial writing assistants, dialog agents, and chatbots.

Open-ended text generation suffers from a mismatch between training and deployment conditions: the model is trained for language modeling, but it is used to generate long-form text. As a consequence, the resulting model distribution $P(y) = \prod_{t=1}^{T} P(y_t \,|\, y_1, \cdots, y_{t-1})$ over sequences $y = (y_1, \cdots, y_T)$ diverges significantly from the human distribution over text sequences. This necessitates the use of decoding algorithms as heuristic post hoc fixes to improve the quality of the generated text (Holtzman et al., 2020). Moreover, due to the open-ended nature of the problem, it is unclear how to quantify the contribution of the heuristics and the value of the text generation model.

Contributions. In Chapter 4, we present a diagnostic measure, MAUVE, to quantify the "goodness" of text generation models (Pillutla et al., 2021b). We overcome the challenges posed by open-endedness, i.e., a diverse set of continuations are possible by directly comparing the resulting distribution of generated text to the distribution of human text. We find that MAUVE correlates strongly with human judgments of how human-like machine-written text is. MAUVE also quantifies several properties of generated text that were previously observed qualitatively, e.g., the effect of the model size or the decoding algorithm.

Chapters 5 and 6: Robustness and Heterogeneity in Federated Learning

Federated learning is now deployed on a global scale with millions of clients. The practical requirements of privacy and the natural characteristics of data heterogeneity raise the need for novel algorithmic solutions that respect the scale and systems requirements of federated learning to classical problems. Robustness is one such problem: existing secure aggregation routines for privacy preservation make it impossible to apply traditional outlier detection and filtration methods to defend the system against corrupted updates sent by malicious clients. Moreover, federated learning suffers from a train-test mismatch due to statistical heterogeneity. The model is trained to minimize the average error across all clients, but it is deployed on individual clients whose distribution might differ from the average, leading to poor predictive performance.

Contributions. In Chapter 5, we study how to make the aggregation process of federated learning robust to corrupted updates contributed by potentially malicious clients (Pillutla et al., 2022a). In particular, we propose RFA, a robust aggregation algorithm based on the classical geometric median (a high-dimensional median analogue), and show how to make it compatible with existing secure aggregation mechanisms for privacy preservation. We establish convergence guarantees for moderate-to-low data heterogeneity.

In Chapter 6, we address the train-test mismatch of federated learning by minimizing the tail statistics of the per-user predictive errors (Pillutla et al., 2021a; Laguel et al., 2021a) through a statistical summary called the superquantile (Rockafellar and Uryasev, 2002). We address the challenges raised by optimizing a nonsmooth functional such as the superquantile by proposing a federated optimization algorithm that is compatible with existing secure aggregation mechanisms and differential privacy. We establish a rate of $O(1/\varepsilon^4)$ communication rounds in the nonsmooth, nonconvex case for an ε-approximate stationary point, identical to rates for the usual mean objective in the smooth case. We also establish a $O(\kappa^{3/2} \log(1/\varepsilon) + \kappa/\varepsilon)$ rate in the nonsmooth but strongly convex case for an ε-approximate minimizer, where κ is a condition number; this is only slightly worse in a lower order term when compared to the $O(\kappa \log(1/\varepsilon) + \kappa/\varepsilon)$ rate for the usual mean objective in the smooth and strongly convex case. In both

chapters, we demonstrate empirically that the proposed algorithms outperform several competitive and state-of-the-art baselines.

Chapter 7: Conclusion

We conclude with a discussion of future research directions that build upon the contributions of this dissertation.

1.3.2 Authorship Details

The work in this dissertation is the result of several research collaborations. Prior publications of the work contained in this thesis are listed below. Unless stated otherwise, the author of this dissertation is the primary author of these papers.

- Chapter 2:

  K. Pillutla, V. Roulet, S. M. Kakade, and Z. Harchaoui. A Smoother Way to Train Structured Prediction Models. In *NeurIPS*, 2018.

- Chapter 3:

  K. Pillutla, V. Roulet, S. M. Kakade, and Z. Harchaoui. The Trade-offs of Incremental Linearization Algorithms for Nonsmooth Composite Problems. *Preprint*, 2022.

- Chapter 4:

  K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui. MAUVE: Measuring the Gap Between Neural Text and Human Text with Divergence Frontiers. In *NeurIPS*, 2021.

- Chapter 5:

  K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust Aggregation for Federated Learning. *IEEE Transactions on Signal Processing*, 2022.

- Chapter 6:

  K. Pillutla, Y. Laguel, J. Malick, and Z. Harchaoui. Federated Learning with Heterogeneous Devices: A Superquantile Optimization Approach. *Submitted*, 2021.

  Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. A Superquantile Approach to Federated Learning with Heterogeneous Devices. In *IEEE CISS*, 2021.

  These works were jointly co-authored with Yassine Laguel, with shared responsibility for the problem formulation, theoretical contributions, and experimental work.

This dissertation does not include work on personalized federated learning (Pillutla et al., 2022b), for which this dissertation author is the primary author and conducted during an internship. It also does not include the following works where the dissertation author is a supporting author: (1) the theoretical study of divergence frontier methods (Liu et al., 2021a), (2) a survey of the applications of the superquantile in machine learning and its smoothing (Laguel et al., 2021b), (3) a study of stochastic and incremental optimization algorithms for objectives based on order statistics (Mehta et al., 2022), and, (4) a formalization of model revision to edit or unlearn particular model behaviors (Fisher et al., 2022).

Chapter 2

Casimir: A Smoother Way to Train Structured Prediction Models

In this chapter, we consider structured prediction, a supervised learning framework to predict discrete data structures such as sequences, trees, and lattices. Structured prediction is ubiquitous in natural language processing, computer vision, and speech processing, with applications such as named entity recognition, machine translation, object detection, and automatic speech recognition.

The enormous size of the output space is a central challenge in structured prediction. For instance, the number of sequences of length p over a base vocabulary V is $|V|^p$, so that brute-force search is often impossible. One overcomes this in practice with efficient combinatorial optimization algorithms, which are required to interact with continuous optimization algorithms while learning structured prediction models. In this chapter, we describe how these combinatorial algorithms can be integrated into differentiable programming pipelines to design smooth and accelerated first-order optimization algorithms.

We focus in particular on maximum margin structured prediction models (Taskar et al., 2004b; Tsochantaridis et al., 2004). The usual empirical risk minimization problem arising in this case is

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left[F(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \right], \tag{2.1}$$

where each $f^{(i)}$ is the structural hinge loss, defined as $f^{(i)}(\boldsymbol{w}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \psi^{(i)}(\boldsymbol{y};\boldsymbol{w})$. Here, $\psi^{(i)}(\boldsymbol{y};\boldsymbol{w})$ measures the advantage of choosing $\boldsymbol{y}$ instead of the true label $\boldsymbol{y}^{(i)}$ for input $\boldsymbol{x}^{(i)}$ (see Section 2.1 for a detailed description). This loss attains its minimum value of 0 if the true label $\boldsymbol{y}^{(i)}$ is the best output to choose under the model for each training example $i = 1, \cdots, n$. The computation of the structural hinge loss itself requires searching over the structured output space $\mathcal{Y}$ and is a *nonsmooth* function of the model parameters, denoted by $\boldsymbol{w} \in \mathbb{R}^d$.

Batch nonsmooth optimization algorithms such as cutting plane methods are appropriate for problems with small or moderate sample sizes (Tsochantaridis et al., 2004; Joachims et al., 2009). Stochastic nonsmooth optimization algorithms such as stochastic subgradient methods can tackle problems with large sample sizes (Ratliff et al., 2007; Shalev-Shwartz et al., 2011). However, both families of methods achieve the typical worst-case complexity bounds of nonsmooth optimization algorithms and cannot easily leverage a possible hidden smoothness of the objective.

Furthermore, as significant progress is being made on incremental smooth optimization algorithms for training unstructured prediction models (Lin et al., 2018), we would like to transfer such advances and design faster optimization algorithms to train structured prediction models. Indeed if each term in the finite-

sum were L-smooth, incremental optimization algorithms such as MISO (Mairal, 2013), SAG (Le Roux et al., 2012; Schmidt et al., 2017), SAGA (Defazio et al., 2014), SDCA (Shalev-Shwartz and Zhang, 2013), and SVRG (Johnson and Zhang, 2013) could leverage the finite-sum structure of the objective (2.1) and achieve faster convergence than batch algorithms on large-scale problems.

Incremental optimization algorithms can be further accelerated, either on a case-by-case basis (Shalev-Shwartz and Zhang, 2014; Frostig et al., 2015; Allen-Zhu, 2017; Defazio, 2016) or using the Catalyst acceleration scheme (Lin et al., 2015; 2018), to achieve near-optimal convergence rates (Woodworth and Srebro, 2016). Accelerated incremental optimization algorithms demonstrate stable and fast convergence behavior on a wide range of problems, in particular for ill-conditioned ones.

We introduce a general framework that allows us to bring the power of accelerated incremental optimization algorithms to the realm of structured prediction problems while leveraging differentiable programming. We seek primal optimization algorithms, as opposed to saddle-point or primal-dual optimization algorithms, in order to be able to tackle structured prediction models with affine mapping as well as deep structured prediction models with nonlinear mappings. We show how to overcome the inherent nonsmoothness of the objective while still being able to rely on efficient inference algorithms. The key contributions of the chapter are as follows.

Smooth Inference Oracles. We introduce a notion of smooth inference oracles that gracefully fits the framework of black-box first-order optimization. While the exp inference oracle reveals the relationship between max-margin and probabilistic structured prediction models, the top-K inference oracle can be efficiently computed using simple modifications of efficient inference algorithms in many cases of interest.

Incremental Optimization Algorithms. We present a new algorithm built on top of SVRG, blending an extrapolation scheme for acceleration and an adaptive smoothing scheme. We establish the worst-case complexity bounds of the proposed algorithm, which equal or improve over the state-of-the-art, as shown in Table 2.1. Finally, we demonstrate its effectiveness compared to competing algorithms on two tasks, namely named entity recognition and visual object localization.

The code is publicly available as a software library called `Casimir`[1]. We start this chapter with a review of structured prediction in Section 2.1. We discuss smoothing for structured prediction in Section 2.2. We define and study the properties of inference oracles in Section 2.3 and describe efficient combinatorial algorithms to implement the inference oracles in Section 2.4. Then, we switch gears to study accelerated incremental algorithms in the convex case in Section 2.5; we study the extensions of the algorithms to deep structured prediction in Chapter 3. Finally, we evaluate the proposed algorithms on two tasks, namely named entity recognition and visual object localization in Section 2.6. Finally, we review related work in Section 2.7 and conclude the chapter with future directions in Section 2.8.

2.1 Background

Structured prediction refers to a supervised learning setting where we aim to predict a structured object $\boldsymbol{y} \in \mathcal{Y}$ such as a sequence or a lattice, from an input $\boldsymbol{x} \in \mathcal{X}$. More concretely, we aim to search for *score functions* ϕ parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ that model the compatibility of input $\boldsymbol{x} \in \mathcal{X}$ and output $\boldsymbol{y} \in \mathcal{Y}$ as

[1] `https://github.com/krishnap25/casimir`

Table 2.1: Convergence rates, given in terms of the number of calls to various oracles, for different optimization algorithms on the learning problem (2.1) for maximum margin structured prediction with linear $\psi^{(i)}(\cdot; \boldsymbol{w})$. The rates are specified in terms of the target accuracy ε, the number of training examples n, the regularization λ, the size of the structured space $|\mathcal{Y}|$, the maximum feature norm $R = \max_{\boldsymbol{y} \in \mathcal{Y}, i \in [n]} \|\psi^{(i)}(\boldsymbol{y}; \boldsymbol{w})\|_2$ and $\widetilde{R} \geq R$ (see Remark 2.24 for explicit form). The rates are specified up to constants and factors logarithmic in the problem parameters. The dependence on the initial error is ignored. * denotes algorithms that make $O(1)$ oracle calls per iteration.

Algo. (*exp* oracle)	**# Oracle calls**
Exponentiated gradient* (Collins et al., 2008)	$\frac{(n + \log \lvert\mathcal{Y}\rvert) R^2}{\lambda \varepsilon}$
Excessive gap reduction (Zhang et al., 2014)	$nR\sqrt{\frac{\log \lvert\mathcal{Y}\rvert}{\lambda \varepsilon}}$
Proposition 2.25*, entropy smoother	$\sqrt{\frac{nR^2 \log \lvert\mathcal{Y}\rvert}{\lambda \varepsilon}}$
Proposition 2.26*, entropy smoother	$n + \frac{R^2 \log \lvert\mathcal{Y}\rvert}{\lambda \varepsilon}$

Algo. (*max* oracle)	**# Oracle calls**
BMRM (Teo et al., 2009)	$\frac{nR^2}{\lambda \varepsilon}$
QP 1-slack (Joachims et al., 2009)	$\frac{nR^2}{\lambda \varepsilon}$
Stochastic subgradient* (Shalev-Shwartz et al., 2011)	$\frac{R^2}{\lambda \varepsilon}$
Block-Coordinate Frank-Wolfe* (Lacoste-Julien et al., 2013)	$n + \frac{R^2}{\lambda \varepsilon}$

Algo. (*top-K* oracle)	**# Oracle calls**
Proposition 2.25*, ℓ_2^2 smoother	$\sqrt{\frac{n\widetilde{R}^2}{\lambda \varepsilon}}$
Proposition 2.26*, ℓ_2^2 smoother	$n + \frac{\widetilde{R}^2}{\lambda \varepsilon}$

$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$. Given such a score function, we can make predict the best output for a given input $\boldsymbol{x}$ as

$$\boldsymbol{y}^*(\boldsymbol{x}; \boldsymbol{w}) \in \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) \,. \tag{2.2}$$

This is also known as the *inference problem*.

The key challenge in structured prediction is that the output space $\mathcal{Y}$ is typically too large for a brute-force search. This is overcome in practice by modeling the structure of the outputs in a manner that the inference problem (2.2) can be computed (or approximated) efficiently by combinatorial algorithms.

In this section, we discuss modeling structured prediction problems with graphical models and then discuss the learning problem. Controlling the computation complexity of the inference problem will be a key theme we shall keep returning to throughout this chapter.

2.1.1 Structured Prediction: Modeling and Inference

A structured prediction model aims to capture the interdependence between the various components of an output $\boldsymbol{y} = (y_1, \cdots, y_p)$. In this chapter, we consider modeling the output structure through a graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The nodes $\mathcal{V} = \{1, \cdots, p\}$ represent the components of the output $\boldsymbol{y}$ while the edges $\mathcal{E}$ define the dependencies between various components. The value of each component y_v for $v \in \mathcal{V}$ represents the state of the node v and takes values from a finite set $\mathcal{Y}_v$. The set of all output structures $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_p$ is finite but exponentially large in the number p of components.

The structure of the graph (i.e., its edge structure) depends on the task. For the task of sequence labeling, the graph is a chain (Figure 1.1, left), while for the task of parsing, the graph is a tree. On the other hand, the graph used in image segmentation is a grid. As we shall see, the techniques in this chapter are most effective when the structure of the graph admits efficient combinatorial search algorithms. This precludes examples such as the complete graph (Figure 1.1, right) that are used in autoregressive neural language models — we shall return to this setting in Chapter 4.

In this chapter, we consider score functions that decompose over the nodes and edges of the graph as

$$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \sum_{v \in \mathcal{V}} \phi_v(\boldsymbol{x}, y_v; \boldsymbol{w}) + \sum_{(v,v') \in \mathcal{E}} \phi_{v,v'}(\boldsymbol{x}, y_v, y_{v'}; \boldsymbol{w}) \,. \tag{2.3}$$

For a fixed $\boldsymbol{w}$, each input $\boldsymbol{x}$ defines a specific compatibility function $\phi(\boldsymbol{x}, \cdot\,; \boldsymbol{w})$. The nature of the learning problem and the corresponding optimization algorithms we consider hinge upon whether ϕ is an affine function of $\boldsymbol{w}$ or not. We consider two settings in this dissertation:

(a) **Pre-defined Feature Map.** In this structured prediction framework, a pre-specified feature map $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is employed and the score ϕ is defined as the linear function

$$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \Phi(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{w} \rangle = \sum_{v \in \mathcal{V}} \langle \Phi_v(\boldsymbol{x}, y_v), \boldsymbol{w} \rangle + \sum_{(v,v') \in \mathcal{E}} \langle \Phi_{v,v'}(\boldsymbol{x}, y_v, y_{v'}), \boldsymbol{w} \rangle \,. \tag{2.4}$$

(b) **Learning the Feature Map.** We also consider the setting where the feature map Φ is parameterized by $\boldsymbol{w}_0$, for example, using a neural network, and is learned from the data. The score function can then be written as

$$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \Phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}_0), \boldsymbol{w}_1 \rangle \tag{2.5}$$

where $\boldsymbol{w} = (\boldsymbol{w}_0, \boldsymbol{w}_1)$ and the scalar product decomposes into nodes and edges as above.

This framework captures both generative probabilistic models such as Hidden Markov Models (HMMs) that model the joint distribution between $\boldsymbol{x}$ and $\boldsymbol{y}$ as well as discriminative probabilistic models, such as conditional random fields (LeCun et al., 1998; Lafferty et al., 2001) where dependencies among the input variables $\boldsymbol{x}$ do not need to be explicitly represented. In these cases, the log joint and conditional probabilities respectively play the role of the score ϕ.

Example 2.1 (Sequence Tagging)**.** Consider the task of sequence tagging in natural language processing where each $\boldsymbol{x} = (x_1, \cdots, x_p) \in \mathcal{X}$ is a sequence of words and $\boldsymbol{y} = (y_1, \cdots, y_p) \in \mathcal{Y}$ is a sequence of labels, both of length p. Common examples include part-of-speech tagging and named entity recognition. Each word x_v in the sequence $\boldsymbol{x}$ comes from a finite dictionary $\mathcal{D}$, and each tag y_v in $\boldsymbol{y}$ takes values from a finite set $\mathcal{Y}_v = \mathcal{Y}_{\text{tag}}$. The corresponding graph is simply a linear chain.

The score function measures the compatibility of a sequence $\boldsymbol{y} \in \mathcal{Y}$ for the input $\boldsymbol{x} \in \mathcal{X}$ using parameters $\boldsymbol{w} = (\boldsymbol{w}_{\text{unary}}, \boldsymbol{w}_{\text{pair}})$ as, for instance,

$$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \sum_{v=1}^{p} \langle \Phi_{\text{unary}}(x_v, y_v), \boldsymbol{w}_{\text{unary}} \rangle + \sum_{v=0}^{p} \langle \Phi_{\text{pair}}(y_v, y_{v+1}), \boldsymbol{w}_{\text{pair}} \rangle \,,$$

where, using $\boldsymbol{w}_{\text{unary}} \in \mathbb{R}^{|\mathcal{D}||\mathcal{Y}_{\text{tag}}|}$ and $\boldsymbol{w}_{\text{pair}} \in \mathbb{R}^{|\mathcal{Y}_{\text{tag}}|^2}$ as node and edge weights respectively, we define for each $v \in [p]$,

$$\langle \Phi_{\text{unary}}(x_v, y_v), \boldsymbol{w}_{\text{unary}} \rangle = \sum_{x \in \mathcal{D},\, j \in \mathcal{Y}_{\text{tag}}} w_{\text{unary},\, x,j}\, \mathbb{I}(x = x_v)\, \mathbb{I}(j = y_v) \,.$$

The pairwise term $\langle \Phi_{\text{pair}}(y_v, y_{v+1}), \boldsymbol{w}_{\text{pair}} \rangle$ is analogously defined. Here, y_0, y_{p+1} are special "start" and "stop" symbols respectively. This can be written as a dot product of $\boldsymbol{w}$ with a pre-specified feature map as in (2.4), by defining

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \big(\sum_{v=1}^{p} \boldsymbol{e}_{x_v} \otimes \boldsymbol{e}_{y_v} \big) \oplus \big(\sum_{v=0}^{p} \boldsymbol{e}_{y_v} \otimes \boldsymbol{e}_{y_{v+1}} \big) \,,$$

where $\boldsymbol{e}_{x_v}$ is the unit vector $(\mathbb{I}(x = x_v))_{x \in \mathcal{D}} \in \mathbb{R}^{|\mathcal{D}|}$, $\boldsymbol{e}_{y_v}$ is the unit vector $(\mathbb{I}(j = y_v))_{j \in \mathcal{Y}_{\text{tag}}} \in \mathbb{R}^{|\mathcal{Y}_{\text{tag}}|}$, $\otimes$ denotes the Kronecker product between vectors and $\oplus$ denotes vector concatenation.

Inference with Combinatorial Algorithms. As mentioned earlier, the inference problem (2.2) cannot typically be solved by brute-force search because the size $m = |\mathcal{Y}|$ of the output space is exponentially large in the size p of each output structure. For sequences, the inference problem can be solved with the famous dynamic programming algorithm of Viterbi (1967). The Viterbi algorithm maintains a table $\pi_v(y_v)$ with the score of the best possible length-v sequence ending in y_v. Starting with a base case of $\pi_1(y_1) = \phi_1(\boldsymbol{x}, y_1; \boldsymbol{w})$ for each label y_1, this table can be recursively updated as

$$\pi_v(y_v) = \phi_v(\boldsymbol{x}, y_v; \boldsymbol{w}) + \max_{y_{v-1}} \left\{ \pi_{v-1}(y_{v-1}) + \phi_{v,v-1}(\boldsymbol{x}, y_v, y_{v-1}; \boldsymbol{w}) \right\} .$$

The maximum value can be obtained from

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \max_{y_p} \pi_p(y_p) ,$$

and the maximizer $\boldsymbol{y}^*(\boldsymbol{x}; \boldsymbol{w})$ can be reconstructed by tracing the maximizers in the recursion (using another table of back-pointers). Assuming each label y_v can take r different values, the Viterbi algorithm runs in time $O(pr^2)$ and requires space $O(pr)$, compared to $O(r^p)$ time for brute-force search.

More generally, the max-product algorithm generalizes the Viterbi algorithm to tree structures (Pearl, 1988). Other combinatorial algorithms such as branch and bound search, minimum graph cuts, or bipartite matching might be applicable, depending on the graph structure.

2.1.2 Learning Structured Prediction Models

Given a dataset of n input-output pairs $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, the learning problem in structured prediction consists in finding model parameters $\boldsymbol{w}$ such that the inferred prediction $\boldsymbol{y}^*(\boldsymbol{x}^{(i)}; \boldsymbol{w}) \approx \boldsymbol{y}^{(i)}$ for $i = 1, \cdots, n$. The approximation in "$\approx$" is quantified by a task loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such as the Hamming loss for sequence prediction. An ideal learning procedure would aim to find the parameters $\boldsymbol{w}$ so as to minimize the task loss over the training dataset:

$$\inf_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell \left(\boldsymbol{y}^{(i)}, \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{w}) \right) . \tag{2.6}$$

Since this objective is piecewise constant in the parameters (due to the argmax), it is not amenable to first-order optimization algorithms.

In this chapter, we focus on a surrogate to the task loss, known as the structural hinge loss (Altun et al., 2003; Taskar et al., 2004a; Tsochantaridis et al., 2004). It is defined on an input-output pair $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ as

$$f^{(i)}(\boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \psi^{(i)}(\boldsymbol{y}; \boldsymbol{w}) := \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{w}) + \ell(\boldsymbol{y}^{(i)}, \boldsymbol{y}) - \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{w}) \right\} . \tag{2.7}$$

The maximization over $\mathcal{Y}$ within the definition of $f^{(i)}$ is known as loss-augmented inference, and we refer to $\psi^{(i)}$ as *the augmented score function*. The combinatorial algorithms to solve the inference problem (2.2) can also handle this maximization, provided the task loss ℓ decomposes along the nodes $\mathcal{V}$ and edges $\mathcal{E}$ of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defining the output structure.

With the structural hinge loss as the surrogate, the optimization problem of interest is the usual empirical risk minimization problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \left\{ F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} f^{(i)}(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right\} , \tag{2.8}$$

where $\lambda \geq 0$ is a regularization parameter.

2.1.3 Notation Review

Vectors are denoted by bold lowercase characters as $\boldsymbol{w} \in \mathbb{R}^d$ while matrices are denoted by bold uppercase characters as $\boldsymbol{A} \in \mathbb{R}^{d \times n}$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, define the norm for $\alpha, \beta \in \{1, 2, \infty\}$,

$$\|\boldsymbol{A}\|_{\beta,\alpha} = \max\{\langle \boldsymbol{y}, \boldsymbol{A}\boldsymbol{x} \rangle \mid \|\boldsymbol{y}\|_\alpha \leq 1\,,\ \|\boldsymbol{x}\|_\beta \leq 1\}\,. \tag{2.9}$$

For any function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, its convex conjugate $f^* : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$f^*(\boldsymbol{z}) = \sup_{\boldsymbol{w} \in \mathbb{R}^d} \{\langle \boldsymbol{z}, \boldsymbol{w} \rangle - f(\boldsymbol{w})\}\ .$$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be L-smooth with respect to an arbitrary norm $\|\cdot\|$ if it is continuously differentiable and its gradient ∇f is L-Lipschitz with respect to $\|\cdot\|$. When left unspecified, $\|\cdot\|$ refers to $\|\cdot\|_2$. Given a continuously differentiable map $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^m$, its Jacobian $\nabla \boldsymbol{g}(\boldsymbol{w}) \in \mathbb{R}^{m \times d}$ at $\boldsymbol{w} \in \mathbb{R}^d$ is defined so that its $(i,j)^{\text{th}}$ entry is $[\nabla \boldsymbol{g}(\boldsymbol{w})]_{ij} = \partial g_i(\boldsymbol{w})/w_j$ where g_i is the i^{th} element of $\boldsymbol{g}$ and w_j is the j^{th} element of $\boldsymbol{w}$. A vector-valued function $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^m$ is said to be L-smooth with respect to $\|\cdot\|$ if it is continuously differentiable and its Jacobian $\nabla \boldsymbol{g}$ is L-Lipschitz with respect to $\|\cdot\|$.

For a vector $\boldsymbol{z} \in \mathbb{R}^m$, $z_{(1)} \geq \cdots \geq z_{(m)}$ refer to its components enumerated in non-increasing order where ties are broken arbitrarily. Further, we let $\boldsymbol{z}_{[k]} = (z_{(1)}, \cdots, z_{(k)}) \in \mathbb{R}^k$ denote the vector of the k largest components of $\boldsymbol{z}$. We denote by Δ^{m-1} the standard probability simplex in $\mathbb{R}^m$. When the dimension is clear from the context, we shall simply denote it by Δ. Moreover, for a positive integer p, $[p]$ refers to the set $\{1, \ldots, p\}$. Lastly, $\widetilde{O}$ in the big-O notation hides factors logarithmic in problem parameters.

Throughout we consider a finite output set $\mathcal{Y}$ with $|\mathcal{Y}| = m$. For some fixed bijection $\sigma : \mathcal{Y} \to [m]$, we let $\boldsymbol{y}^{\text{th}}$ component $u_{\boldsymbol{y}}$ of a vector $\boldsymbol{u} \in \mathbb{R}^m$ to refer to the $\sigma(\boldsymbol{y})^{\text{th}}$ component $u_{\sigma(\boldsymbol{y})}$.

2.2 Smooth Structured Prediction

Recall that we defined the structural hinge loss w.r.t. a fixed input-output pair $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ as (dropping the index i for simplicity)

$$f(\boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})\,, \tag{2.10}$$

where ψ is the augmented score function. The task loss ℓ is assumed to possess appropriate structure so that the maximization inside (2.10), known as *loss-augmented inference*, is no harder than the inference problem in (2.2).

When the map $\boldsymbol{w} \mapsto \psi(\boldsymbol{y}; \boldsymbol{w})$ is affine, the structural hinge loss f and the objective F from (2.1) are both convex — we refer to this case as the structural support vector machine. When $\boldsymbol{w} \mapsto \psi(\boldsymbol{y}; \boldsymbol{w})$ is a nonlinear but smooth map, then the structural hinge loss f and the objective F are nonconvex.

2.2.1 Smoothing Strategy

We first rewrite the structural hinge loss as a composition

$$\boldsymbol{g} : \begin{cases} \mathbb{R}^d & \to \mathbb{R}^m \\ \boldsymbol{w} & \mapsto (\psi(\boldsymbol{y}; \boldsymbol{w}))_{\boldsymbol{y} \in \mathcal{Y}}, \end{cases} \qquad h : \begin{cases} \mathbb{R}^m & \to \mathbb{R} \\ \boldsymbol{z} & \mapsto \max_{i \in [m]} z_i, \end{cases} \tag{2.11}$$

where $m = |\mathcal{Y}|$ so that the structural hinge loss reads

$$f(\boldsymbol{w}) = h \circ \boldsymbol{g}(\boldsymbol{w})\,. \tag{2.12}$$

The nonsmoothness of the structural hinge loss arises from the nonsmoothness of the max function h. Any nonsmooth convex function can be smoothed taking its infimal convolution with a smooth function (Nesterov, 2005b; Beck and Teboulle, 2012). Its dual representation leads to the smoothing $h_{\mu\omega}$ of h with a strongly convex function $\omega : \operatorname{dom} h^* \to \mathbb{R}$ and a smoothing parameter $\mu > 0$ as

$$h_{\mu\omega}(\boldsymbol{z}) = \max_{\boldsymbol{u} \in \operatorname{dom} h^*} \left\{ \langle \boldsymbol{u}, \boldsymbol{z} \rangle - h^*(\boldsymbol{u}) - \mu\omega(\boldsymbol{u}) \right\} .$$

Its gradient $h_{\mu\omega}(\boldsymbol{z})$ is simply the maximizer in the expression above; see Appendix A.1 for a review. We smooth the structural hinge loss (2.12) by simply smoothing the non-smooth max function h as

$$f_{\mu\omega} = h_{\mu\omega} \circ \boldsymbol{g}.$$

When $\boldsymbol{g}$ is smooth and Lipschitz continuous, $f_{\mu\omega}$ is a smooth approximation of the structural hinge loss, whose gradient is readily given by the chain rule. The smoothing parameter μ controls both the approximation quality and the smoothness. For any $\mu_1 \geq \mu_2 \geq 0$, we have the approximation error bound

$$(\mu_1 - \mu_2) \min_{\boldsymbol{u} \in \Delta^{m-1}} \omega(\boldsymbol{u}) \leq f_{\mu_2\omega}(\boldsymbol{w}) - f_{\mu_1\omega}(\boldsymbol{w}) \leq (\mu_1 - \mu_2) \max_{\boldsymbol{u} \in \Delta^{m-1}} \omega(\boldsymbol{u}) . \tag{2.13}$$

Next, we turn to smoothness. When ω is 1-strongly convex w.r.t. $\|\cdot\|_\alpha$, and $\boldsymbol{g}$ is an affine map $\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}$, it follows that $f_{\mu\omega}$ is $(\|\boldsymbol{A}\|^2_{\beta,\alpha}/\mu)$-smooth with respect to $\|\cdot\|_\beta$ (cf. Lemma A.3 in Appendix A.1).

2.2.2 Smoothing Variants

In the context of smoothing the max function, we now describe two popular choices for the smoothing function ω, followed by computational considerations.

Entropy and ℓ_2^2 smoothing. When h is the max function, the smoothing operation can be computed analytically for the *entropy* smoother and the ℓ_2^2 smoother, denoted respectively as

$$-H(\boldsymbol{u}) := \langle \boldsymbol{u}, \log \boldsymbol{u} \rangle \qquad \text{and} \qquad \ell_2^2(\boldsymbol{u}) := \tfrac{1}{2}(\|\boldsymbol{u}\|_2^2 - 1) .$$

These lead respectively to the log-sum-exp function (Nesterov, 2005b, Lemma 4)

$$h_{-\mu H}(\boldsymbol{z}) = \mu \log \left(\sum_{i=1}^m e^{z_i/\mu} \right), \quad \nabla h_{-\mu H}(\boldsymbol{z}) = \left[\frac{e^{z_i/\mu}}{\sum_{j=1}^m e^{z_j/\mu}} \right]_{i=1,\ldots,m},$$

and an orthogonal projection onto the simplex,

$$h_{\mu\ell_2^2}(\boldsymbol{z}) = \langle \boldsymbol{z}, \operatorname{proj}_{\Delta^{m-1}}(\boldsymbol{z}/\mu) \rangle - \tfrac{\mu}{2} \| \operatorname{proj}_{\Delta^{m-1}}(\boldsymbol{z}/\mu) \|^2 + \tfrac{\mu}{2}, \quad \nabla h_{\mu\ell_2^2}(\boldsymbol{z}) = \operatorname{proj}_{\Delta^{m-1}}(\boldsymbol{z}/\mu) .$$

Furthermore, (2.13) translates, for all $\mu_1 \geq \mu_2 \geq 0$, to,

$$0 \leq h_{-\mu_1 H}(\boldsymbol{z}) - h_{-\mu_2 H}(\boldsymbol{z}) \leq (\mu_1 - \mu_2) \log m, \quad \text{and,} \quad 0 \leq h_{\mu_1\ell_2^2}(\boldsymbol{z}) - h_{\mu_2\ell_2^2}(\boldsymbol{z}) \leq \tfrac{1}{2}(\mu_1 - \mu_2) .$$

Top-K Strategy. Though the gradient of the composition $f_{\mu\omega} = h_{\mu\omega} \circ \boldsymbol{g}$ can be written using the chain rule, its actual computation for structured prediction problems involves computing $\nabla \boldsymbol{g}$ over all $m = |\mathcal{Y}|$ of its components, which may be intractable. However, in the case of ℓ_2^2 smoothing, projections onto the simplex are sparse, as pointed out by the following proposition.

Proposition 2.2. *Consider the Euclidean projection $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \Delta^{m-1}} \|\boldsymbol{u} - \boldsymbol{z}/\mu\|_2^2$ of $\boldsymbol{z}/\mu \in \mathbb{R}^m$ onto the simplex, where $\mu > 0$. The projection $\boldsymbol{u}^*$ has exactly $k \in [m]$ non-zeros if and only if*

$$\sum_{i=1}^{k} \big(z_{(i)} - z_{(k)}\big) < \mu \leq \sum_{i=1}^{k} \big(z_{(i)} - z_{(k+1)}\big) \;, \tag{2.14}$$

where $z_{(1)} \geq \cdots \geq z_{(m)}$ are the components of $\boldsymbol{z}$ in non-decreasing order and $z_{(m+1)} := -\infty$. In this case, $\boldsymbol{u}^*$ is given by

$$u_i^* = \max\left\{0,\; \tfrac{1}{k\mu} \sum_{j=1}^{k} \big(z_i - z_{(j)}\big) + \tfrac{1}{k}\right\}.$$

Proof. The projection $\boldsymbol{u}^*$ satisfies $u_i^* = (z_i/\mu + \rho^*)_+$, where ρ^* is the unique solution of ρ in the equation (Held et al., 1974)

$$\sum_{i=1}^{m} \left(\frac{z_i}{\mu} + \rho\right)_+ = 1 \,, \tag{2.15}$$

where $\alpha_+ = \max\{0, \alpha\}$ (e.g., Held et al., 1974). Note that $z_{(i)}/\mu + \rho^* \leq 0$ implies that $z_{(j)}/\mu + \rho^* \leq 0$ for all $j \geq i$. Therefore $\boldsymbol{u}^*$ has k non-zeros if and only if $z_{(k)}/\mu + \rho^* > 0$ and $z_{(k+1)}/\mu + \rho^* \leq 0$.

Now suppose that $\boldsymbol{u}^*$ has exactly k non-zeros, we can then solve (2.15) to obtain $\rho^* = \varphi_k(\boldsymbol{z}/\mu)$, which is defined as

$$\varphi_k\left(\frac{\boldsymbol{z}}{\mu}\right) := \frac{1}{k} - \frac{1}{k} \sum_{i=1}^{k} \frac{z_{(i)}}{\mu} \,. \tag{2.16}$$

Plugging in the value of ρ^* in $z_{(k)}/\mu + \rho^* > 0$ gives $\mu > \sum_{i=1}^{k} \big(z_{(i)} - z_{(k)}\big)$. Likewise, $z_{(k+1)}/\mu + \rho^* \leq 0$ gives $\mu \leq \sum_{i=1}^{k} \big(z_{(i)} - z_{(k+1)}\big)$.

Conversely assume (2.14) and let $\widehat{\rho} = \varphi_k(\boldsymbol{z}/\mu)$. Eq. (2.14) can be written as $z_{(k)}/\mu + \widehat{\rho} > 0$ and $z_{(k+1)}/\mu + \widehat{\rho} \leq 0$. Furthermore, we verify that $\widehat{\rho}$ satisfies Eq. (2.15), and so $\widehat{\rho} = \rho^*$ is its unique root. It follows, therefore, that the sparsity of $\boldsymbol{u}^*$ is k. □

Thus, the projection of $\boldsymbol{z}/\mu$ onto the simplex picks out some number $K_{\boldsymbol{z}/\mu}$ of the largest entries of $\boldsymbol{z}/\mu$ — we refer to this as the sparsity of $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}/\mu)$. This fact motivates the *top-K strategy*: given $\mu > 0$, fix an integer K *a priori* and consider as surrogates for $h_{\mu\ell_2^2}$ and $\nabla h_{\mu\ell_2^2}$ respectively

$$h_{\mu,K}(\boldsymbol{z}) := \max_{\boldsymbol{u} \in \Delta^{K-1}} \big\{\big\langle \boldsymbol{z}_{[K]}, \boldsymbol{u}\big\rangle - \mu \ell_2^2(\boldsymbol{u})\big\} \;, \quad \text{and,} \quad \widetilde{\nabla} h_{\mu,K}(\boldsymbol{z}) := \Omega_K(\boldsymbol{z})^\top \,\mathrm{proj}_{\Delta^{K-1}}\left(\frac{\boldsymbol{z}_{[K]}}{\mu}\right) \,,$$

where $\boldsymbol{z}_{[K]}$ denotes the vector composed of the K largest entries of $\boldsymbol{z}$ and $\Omega_K : \mathbb{R}^m \to \{0,1\}^{K \times m}$ defines their extraction, i.e., $\Omega_K(\boldsymbol{z}) = (\boldsymbol{e}_{j_1}^\top, \ldots, \boldsymbol{e}_{j_K}^\top)^\top \in \{0,1\}^{K \times m}$ where $j_1, \cdots, j_K$ satisfy $z_{j_1} \geq \cdots \geq z_{j_K}$ such that $\boldsymbol{z}_{[K]} = \Omega_K(\boldsymbol{z})\boldsymbol{z}$. A surrogate of the ℓ_2^2 smoothing is then given by

$$f_{\mu,K} := h_{\mu,K} \circ \boldsymbol{g} \,, \quad \text{and,} \quad \widetilde{\nabla} f_{\mu,K}(\boldsymbol{w}) := \nabla \boldsymbol{g}(\boldsymbol{w})^\top \widetilde{\nabla} h_{\mu,K}(\boldsymbol{g}(\boldsymbol{w})) \,. \tag{2.17}$$

Exactness of Top-K Strategy. We say that the top-K strategy is *exact* at $\boldsymbol{z}$ for $\mu > 0$ when it recovers the first order information of $h_{\mu\ell_2^2}$, i.e., when $h_{\mu\ell_2^2}(\boldsymbol{z}) = h_{\mu,K}(\boldsymbol{z})$ and $\nabla h_{\mu\ell_2^2}(\boldsymbol{z}) = \widetilde{\nabla} h_{\mu,K}(\boldsymbol{z})$. The next proposition outlines when this is the case. Note that if the top-K strategy is exact at $\boldsymbol{z}$ for a smoothing parameter $\mu > 0$ then it will be exact at $\boldsymbol{z}$ for any $\mu' < \mu$.

Proposition 2.3. *The top-K strategy is exact at $\boldsymbol{z}$ for $\mu > 0$ if*

$$\mu \leq \sum_{i=1}^{K} \left(\boldsymbol{z}_{(i)} - \boldsymbol{z}_{(K+1)} \right) . \tag{2.18}$$

Moreover, for any fixed $\boldsymbol{z} \in \mathbb{R}^m$ such that the vector $\boldsymbol{z}_{[K+1]} = \Omega_{K+1}(\boldsymbol{z})\boldsymbol{z}$ has at least two unique elements, the top-K strategy is exact at $\boldsymbol{z}$ for all μ satisfying $0 < \mu \leq z_{(1)} - z_{(K+1)}$.

Proof. First, we note that the top-K strategy is exact when the sparsity $K_{\boldsymbol{z}/\mu}$ of the projection $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}/\mu)$ satisfies $K_{\boldsymbol{z}/\mu} \leq K$. From Proposition 2.2, the condition that $K_{\boldsymbol{z}/\mu} \in \{1, 2, \cdots, K\}$ happens when

$$\mu \in \bigcup_{k=1}^{K} \left(\sum_{i=1}^{k} \left(z_{(i)} - z_{(k)} \right), \sum_{i=1}^{k} \left(z_{(i)} - z_{(k+1)} \right) \right] = \left(0, \sum_{i=1}^{K} \left(z_{(i)} - z_{(K+1)} \right) \right] ,$$

since the intervals in the union are contiguous. This establishes (2.18).

The only case when (2.18) cannot hold for any value of $\mu > 0$ is when the right hand size of (2.18) is zero. In the opposite case when $\boldsymbol{z}_{[K+1]}$ has at least two unique components, or equivalently, $z_{(1)} - z_{(K+1)} > 0$, the condition $0 < \mu \leq z_{(1)} - z_{(K+1)}$ implies (2.18). □

If the top-K strategy is exact at $\boldsymbol{g}(\boldsymbol{w})$ for μ, then

$$f_{\mu,K}(\boldsymbol{w}) = f_{\mu\ell_2^2}(\boldsymbol{w}) \quad \text{and} \quad \widetilde{\nabla} f_{\mu,K}(\boldsymbol{w}) = \nabla f_{\mu\ell_2^2}(\boldsymbol{w}) ,$$

where the latter follows from the chain rule. When used instead of ℓ_2^2 smoothing in the algorithms presented in Section 2.5, the top-K strategy provides a computationally efficient heuristic to smooth the structural hinge loss. Though we do not have theoretical guarantees using this surrogate, experiments presented in Section 2.6 show its efficiency and its robustness to the choice of K.

Gobal and Local Smoothness Bounds. Consider the case of affine inner functions $\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{d \times m}$. For entropy smoothing $f_{-\mu H}$, we get a global bound on the smoothness as (cf. Lemma A.3 in Appendix A.1)

$$L_{-\mu H} = \frac{\|\boldsymbol{A}\|_{2,1}^2}{\mu} = \max_{i=1,\cdots,m} \frac{\|\boldsymbol{a}_i\|_2^2}{\mu} ,$$

where $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m$ are the rows of $\boldsymbol{A}$. On the other hand, for ℓ_2^2, the corresponding global smoothness bound is

$$L_{-\mu\ell_2^2} = \frac{\|\boldsymbol{A}\|_{2,2}^2}{\mu} .$$

Note that $\|\boldsymbol{A}\|_{2,2}$ is the spectral norm of $\boldsymbol{A}$, and we have the relation $\|\boldsymbol{A}\|_{2,1} \leq \|\boldsymbol{A}\|_{2,2} \leq m \|\boldsymbol{A}\|_{2,1}$. However, this global bound can be quite loose locally due to the sparsity of ℓ_2^2 smoothing. A careful analysis can give a tighter local smoothness bound of $K_{\boldsymbol{w}} L_{-\mu H}$ in the neighborhood of $\boldsymbol{w}$, where $K_{\boldsymbol{w}}$ is the sparsity of the projection $\mathrm{proj}_{\Delta^{m-1}}\big((\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b})/\mu\big)$.

Lemma 2.4. *Let $h(\boldsymbol{z}) = \max_{j\in[m]} z_j$ be the max function and let $h_\mu \equiv h_{\mu\ell_2^2}$ denote its Euclidean smoothing. Let $g_j : \mathbb{R}^d \to \mathbb{R}$ be B_j-Lipschitz and L_j-smooth w.r.t. $\|\cdot\|_2$ for $j \in [m]$ and define $\boldsymbol{g} = (g_1, \cdots, g_m) : \mathbb{R}^d \to \mathbb{R}^m$. Further, for any $\boldsymbol{w} \in \mathbb{R}^d$, let*

$$S(\boldsymbol{w}) = \{j \in [m] \,:\, [\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}))]_j \neq 0\} = \left\{ j \in [m] \,:\, \left[\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{g}(\boldsymbol{w})/\mu)\right]_j \right\} .$$

Then, we have for any $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$ that

$$\begin{aligned}\|\nabla(h_\mu \circ \boldsymbol{g})(\boldsymbol{w}) - \nabla(h_\mu \circ \boldsymbol{g})(\boldsymbol{w}')\|_2 &\leq \left(\Big(\sum_{j\in S(\boldsymbol{w})\cup S(\boldsymbol{w}')} B_j^2 \Big)^{1/2} + \max_{j\in[m]} L_j \right) \|\boldsymbol{w} - \boldsymbol{w}'\|_2 \\ &\leq \left(\big|S(\boldsymbol{w}) \cup S(\boldsymbol{w}')\big| \max_{j\in m} B_j + \max_{j\in m} L_j \right) \|\boldsymbol{w} - \boldsymbol{w}'\|_2 .\end{aligned}$$

As a consequence, $h_\mu \circ \boldsymbol{g}$ is globally $\left(\frac{1}{\mu}\sum_{j=1}^m B_j^2 + \max_{j\in[m]} L_j\right)$-smooth w.r.t. $\|\cdot\|_2$ over $\mathbb{R}^d$.

The proof of this lemma is given in Appendix A.1.

2.3 Inference Oracles

We now define inference oracles as first order oracles in structured prediction. These are used later to understand the information-based complexity of optimization algorithms.

First Order Oracles in Structured Prediction. A first order oracle for a function $f : \mathbb{R}^d \to \mathbb{R}$ is a routine which, given a point $\boldsymbol{w} \in \mathbb{R}^d$, returns on output a value $f(\boldsymbol{w})$ and a (sub)gradient $\boldsymbol{v} \in \partial f(\boldsymbol{w})$, where ∂f is the Fréchet (or regular) subdifferential (Rockafellar and Wets, 2009, Def. 8.3). We now define inference oracles as first order oracles for the structural hinge loss f and its smoothed variants $f_{\mu\omega}$. Note that these definitions are independent of the graphical structure. However, as we shall see, the graphical structure plays a crucial role in the implementation of the inference oracles.

Definition 2.5. Consider an augmented score function ψ, a level of smoothing $\mu > 0$ and the structural hinge loss $f(\boldsymbol{w}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$. For a given $\boldsymbol{w} \in \mathbb{R}^d$,

(i) the *max oracle* returns $f(\boldsymbol{w})$ and $\boldsymbol{v} \in \partial f(\boldsymbol{w})$.

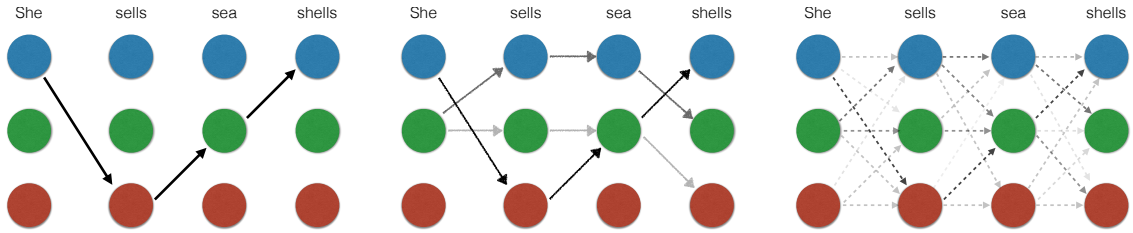
(ii) the *exp oracle* returns $f_{-\mu H}(\boldsymbol{w})$ and $\nabla f_{-\mu H}(\boldsymbol{w})$.

(iii) the *top-K oracle* returns $f_{\mu,K}(\boldsymbol{w})$ and $\widetilde{\nabla} f_{\mu,K}(\boldsymbol{w})$ as surrogates for $f_{\mu\ell_2^2}(\boldsymbol{w})$ and $\nabla f_{\mu\ell_2^2}(\boldsymbol{w})$ respectively.

Note that the exp oracle gets its name since it can be written as an expectation over all $\boldsymbol{y}$, as revealed by the next lemma, which gives analytical expressions for the gradients returned by the oracles.

Lemma 2.6. *Consider the setting of Definition 2.5. We have the following:*

(i) *For any $\boldsymbol{y}^* \in \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$, we have that $\nabla_{\boldsymbol{w}} \psi(\boldsymbol{y}^*; \boldsymbol{w}) \in \partial f(\boldsymbol{w})$. That is, the max oracle can be implemented by inference.*



(a) Non-smooth. **(b)** ℓ_2^2 smoothing. **(c)** Entropy smoothing.

Figure 2.1: Viterbi trellis for a chain graph with $p = 4$ nodes and 3 labels.

(ii) *The output of the exp oracle satisfies* $\nabla f_{-\mu H}(\boldsymbol{w}) = \sum_{\boldsymbol{y}\in\mathcal{Y}} P_{\psi,\mu}(\boldsymbol{y};\boldsymbol{w})\nabla\psi(\boldsymbol{y};\boldsymbol{w})$*, where*

$$P_{\psi,\mu}(\boldsymbol{y};\boldsymbol{w}) = \frac{\exp\left(\frac{1}{\mu}\psi(\boldsymbol{y};\boldsymbol{w})\right)}{\sum_{\boldsymbol{y}'\in\mathcal{Y}}\exp\left(\frac{1}{\mu}\psi(\boldsymbol{y}';\boldsymbol{w})\right)}\,.$$

(iii) *The output of the top-K oracle satisfies* $\widetilde{\nabla} f_{\mu,K}(\boldsymbol{w}) = \sum_{i=1}^{K} u^*_{\psi,\mu,i}(\boldsymbol{w})\nabla\psi(\boldsymbol{y}_{(i)};\boldsymbol{w})\,,$ *where* $Y_K = \{\boldsymbol{y}_{(1)},\cdots,\boldsymbol{y}_{(K)}\}$ *is the set of K largest scoring outputs satisfying*

$$\psi(\boldsymbol{y}_{(1)};\boldsymbol{w}) \geq \cdots \geq \psi(\boldsymbol{y}_{(K)};\boldsymbol{w}) \geq \max_{\boldsymbol{y}\in\mathcal{Y}\setminus Y_K}\psi(\boldsymbol{y};\boldsymbol{w})\,,$$

and $\boldsymbol{u}^*_{\psi,\mu} = \mathrm{proj}_{\Delta^{K-1}}\left(\left[\psi(\boldsymbol{y}_{(1)};\boldsymbol{w}),\cdots,\psi(\boldsymbol{y}_{(K)};\boldsymbol{w})\right]^\top\right)$.

Proof. Part (ii) deals with the composition of differentiable functions, and follows from the chain rule. Part (iii) follows from the definition in Eq. (2.17). The proof of Part (i) follows from the chain rule for Fréchet subdifferentials of compositions (Rockafellar and Wets, 2009, Theorem 10.6) together with the fact that by convexity and Danskin's theorem (Bertsekas, 1999, Proposition B.25), the subdifferential of the max function is given by $\partial h(\boldsymbol{z}) = \mathrm{conv}\{\boldsymbol{e}_i \,|\, i \in [m] \text{ such that } z_i = h(\boldsymbol{z})\}$. □

Example 2.7. Consider the task of sequence tagging from Example 2.1. The inference problem (2.2) is a search over all $|\mathcal{Y}| = |\mathcal{Y}_{\text{tag}}|^p$ label sequences. For chain graphs, this is equivalent to searching for the shortest path in the associated trellis, shown in Figure 2.1. An efficient dynamic programming approach called the Viterbi algorithm (Viterbi, 1967) can solve this problem in space and time polynomial in p and $|\mathcal{Y}_{\text{tag}}|$. The structural hinge loss is non-smooth because a small change in $\boldsymbol{w}$ might lead to a large change in the best scoring path shown in Figure 2.1.

When smoothing f with $\omega = \ell_2^2$, the smoothed function $f_{\mu\ell_2^2}$ is given by a projection onto the simplex, which picks out some number $K_{\psi/\mu}$ of the highest scoring outputs $\boldsymbol{y} \in \mathcal{Y}$ or equivalently, $K_{\psi/\mu}$ shortest paths in the Viterbi trellis (Figure 2.1b). The top-K oracle then uses the top-K strategy to approximate $f_{\mu\ell_2^2}$ with $f_{\mu,K}$.

On the other hand, with entropy smoothing $\omega = -H$, we get the log-sum-exp function and its gradient is obtained by averaging over paths with weights such that shorter paths have a larger weight (cf. Lemma 2.6(ii)). This is visualized in Figure 2.1c.

Top-K Oracle and Differentiable Programming. The computation of $\tilde{\nabla} f_{\mu,K}$ of the top-K oracle within a differentiable programming framework is straightforward. First, we obtain $Y_K = \{\boldsymbol{y}_{(1)}, \cdots, \boldsymbol{y}_{(K)}\}$ with efficient combinatorial algorithms that we describe in the upcoming section. Second, we calculate $\boldsymbol{u}^*_{\psi,\mu}$ by projecting the top-K scores $\psi(\boldsymbol{y}_{(i)}; \boldsymbol{w})$ on the simplex — this operation can be hidden from the automatic differentiation engine, e.g., with `torch.no_grad()` in PyTorch. Finally, we compute the linear combination $f_{\mu,K}(\boldsymbol{w}) = \sum_{i=1}^{K} u^*_{\psi,\mu,i} \psi(\boldsymbol{y}_{(i)}); \boldsymbol{w})$ so that the gradient $\nabla f_{\mu,K}(\boldsymbol{w})$ can be directly obtained using a automatic differentiation.

Exp Oracles and Conditional Random Fields. Recall that a *Conditional Random Field (CRF)* (LeCun et al., 1998; Lafferty et al., 2001) with augmented score function ψ and parameters $\boldsymbol{w} \in \mathbb{R}^d$ is a probabilistic model that assigns to output $\boldsymbol{y} \in \mathcal{Y}$ the probability

$$\mathbb{P}(\boldsymbol{y} \mid \psi; \boldsymbol{w}) = \exp\left(\psi(\boldsymbol{y}; \boldsymbol{w}) - A_\psi(\boldsymbol{w})\right) , \tag{2.19}$$

where $A_\psi(\boldsymbol{w})$ is known as the log-partition function, a normalizer so that the probabilities sum to one. Gradient-based maximum likelihood learning algorithms for CRFs require computation of the log-partition function $A_\psi(\boldsymbol{w})$ and its gradient $\nabla A_\psi(\boldsymbol{w})$. The next proposition relates the computational costs of the exp oracle and the log-partition function.

Proposition 2.8. *The exp oracle for an augmented score function ψ with parameters $\boldsymbol{w} \in \mathbb{R}^d$ is equivalent in hardness to computing the log-partition function $A_\psi(\boldsymbol{w})$ and its gradient $\nabla A_\psi(\boldsymbol{w})$ for a conditional random field with augmented score function ψ.*

Proof. Fix a smoothing parameter $\mu > 0$. Consider a CRF with an augmented score function $\psi'(\boldsymbol{y}; \boldsymbol{w}) = \mu^{-1}\psi(\boldsymbol{y}; \boldsymbol{w})$. Its log-partition function $A_{\psi'}(\boldsymbol{w})$ satisfies $\exp(A_{\psi'}(\boldsymbol{w})) = \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp\left(\mu^{-1}\psi(\boldsymbol{y}; \boldsymbol{w})\right)$. The claim now follows from the bijection $f_{-\mu H}(\boldsymbol{w}) = \mu\, A_{\psi'}(\boldsymbol{w})$ between $f_{-\mu H}$ and $A_{\psi'}$. □

2.4 Implementation of Inference Oracles

We now turn to the concrete implementation of the inference oracles. This depends crucially on the structure of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. If the graph $\mathcal{G}$ is a tree, then the inference oracles can be computed exactly with efficient procedures, as we shall see in Section 2.4.1. When the graph $\mathcal{G}$ is not a tree, we study special cases when a specific discrete structure can be exploited to efficiently implement some of the inference oracles in Section 2.4.2. The results of this section are summarized in Table 2.2.

Setting. Consider a structured output space $\mathcal{Y}$, whose structure is encoded by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In this section we will construct smooth surrogates to $\max_{\boldsymbol{y}\in\mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$, where $\psi : \mathcal{Y} \times \mathbb{R}^d \to \mathbb{R}$ is any function which decomposes along nodes $\mathcal{V}$ and edges $\mathcal{E}$ of $\mathcal{G}$ as

$$\psi(\boldsymbol{y}; \boldsymbol{w}) = \sum_{v\in\mathcal{V}} \psi_v(y_v; \boldsymbol{w}) + \sum_{(v,v')\in\mathcal{E}} \psi_{v,v'}(y_v, y_{v'}; \boldsymbol{w}) . \tag{2.20}$$

This structural hinge loss considered earlier is encompassed by this setting. Indeed, for an input-output pair $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, consider the augmented score function $\psi(\boldsymbol{y}; \boldsymbol{w}) = \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}; \boldsymbol{w}) + \ell(\boldsymbol{y}^{(i)}, \boldsymbol{y}) - \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{w})$ it defines, where the index of the sample is dropped by convenience. The decomposition (2.20) follows from (2.3) and the decomposability of the loss.

To be consistent with the previous notation, we shall continue to call ψ as the augmented score function. When $\boldsymbol{w}$ is clear from the context, we denote $\psi(\cdot\,; \boldsymbol{w})$ by ψ. Likewise for ψ_v and $\psi_{v,v'}$.

Table 2.2: Smooth inference oracles, algorithms, and complexity. Here, p is the size of each $\boldsymbol{y} \in \mathcal{Y}$. The time complexity is phrased in terms of the time complexity $\mathcal{T}$ of the max oracle.

Max oracle	**Top-K oracle**		**Exp oracle**	
Algorithm	Algorithm	Time	Algorithm	Time
Max-product	Top-K max-product	$O(K\mathcal{T} \log K)$	Sum-product	$O(\mathcal{T})$
Graph cut	BMMF	$O(pK\mathcal{T})$	Intractable	
Graph matching	BMMF	$O(K\mathcal{T})$	Intractable	
Branch & bound search	Top-K search	N/A	Intractable	

2.4.1 Inference Oracles in Trees

We first consider algorithms implementing the inference algorithms in trees and examine their computational complexity. Since the max oracle in trees is given by dynamic programming, it follows that implementations of smooth inference oracles show us how to smooth dynamic programming.

Max Oracle. In tree-structured graphical models, the inference problem (2.2), and thus the max oracle (cf. Lemma 2.6(i)) can always be solved exactly in polynomial time by the max-product algorithm (Pearl, 1988), which uses the technique of dynamic programming (Bellman, 1957). The Viterbi algorithm for chain graphs from Example 2.7 is a special case.

Top-K Oracle. The top-K oracle uses a generalization of the max-product algorithm that we name top-K max-product algorithm. It keeps track of the K-best intermediate structures while the max-product algorithm just tracks the single best intermediate structure (cf. Seroussi and Golmard, 1994). Formally, the kth largest element from a discrete set S is defined as

$$\max_{x \in S}^{(k)} f(x) = \begin{cases} k^{\text{th}} \text{ largest element of } \{f(y) \,|\, y \in S\} & k \leq |S| \\ -\infty, & k > |S| \,. \end{cases}$$

We present the algorithm in the simple case of chain structured graphical models in Algorithm 2.1. Note that it requires $\widetilde{O}(K)$ times the time and space of the max oracle.

Exp oracle. The relationship of the exp oracle with CRFs (Proposition 2.8) leads directly to Algorithm 2.2, which is based on marginal computations from the sum-product algorithm.

Remark 2.9. *We note that clique trees allow the generalization of the algorithms of this section to general graphs with cycles. However, the construction of a clique tree requires time and space exponential in the* treewidth *of the graph.*

Example 2.10. Consider the task of sequence tagging from Example 2.1. The top-K Viterbi algorithm (Algorithm 2.1) must store in $\pi_v^{(k)}(y_v)$ the score of k^{th} best length-v prefix that ends in y_v for each $k \in [K]$. In contrast, the usual Viterbi algorithm (Viterbi, 1967) maintains a table $\pi_v(y_v)$, which stores the best length-v prefix ending in label y_v. Here, the entry $\pi_v(y_v)$ is updated by looking at the previous

Algorithm 2.1. Top-K max-product (top-K Viterbi) algorithm for chain graphs

Input: Augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$ defined on chain graph $\mathcal{G}$, integer $K > 0$.

1: For $k = 1, \cdots, K$, set $\pi_1^{(k)}(y_1) \leftarrow \psi_1(y_1)$ if $k = 1$ and $-\infty$ otherwise for all $y_1 \in \mathcal{Y}_1$.

2: **for** $v = 2, \cdots p$ and $k = 1, \cdots, K$ **do**

3: For all $y_v \in \mathcal{Y}_v$, set

$$\pi_v^{(k)}(y_v) \leftarrow \psi_v(y_v) + \max_{y_{v-1} \in \mathcal{Y}_{v-1}, \ell \in [K]}^{(k)} \left\{ \pi_{v-1}^{(\ell)}(y_{v-1}) + \psi_{v,v-1}(y_v, y_{v-1}) \right\} . \tag{2.21}$$

4: Assign to $\delta_v^{(k)}(y_v), \kappa_v^{(k)}(y_v)$ the y_{v-1}, ℓ that attain the $\max^{(k)}$ above for each $y_v \in \mathcal{Y}_v$.

5: For $k = 1, \cdots, K$, set $\psi^{(k)} \leftarrow \max_{y_p \in \mathcal{Y}_p, k \in [K]}^{(k)} \pi_p^{(k)}(y_p)$ and store in $y_p^{(k)}, \ell^{(k)}$ respectively the maximizing assignments of y_p, k.

6: **for** $v = p - 1, \cdots 1$ and $k = 1, \cdots, K$ **do**

7: Set $y_v^{(k)} \leftarrow \delta_{v+1}^{(\ell^{(k)})}\big(y_{v+1}^{(k)}\big)$ and $\ell^{(k)} \leftarrow \kappa_{v+1}^{(\ell^{(k)})}\big(y_{v+1}^{(k)}\big)$.

8: **return** $\left\{ \psi^{(k)}, \boldsymbol{y}^{(k)} := (y_1^{(k)}, \cdots, y_p^{(k)}) \right\}_{k=1}^{K}$.

Algorithm 2.2. Entropy smoothed max-product algorithm

Input: Augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$ defined on tree-structured graph $\mathcal{G}$, $\mu > 0$.

1: Compute the log-partition function and marginals using the sum-product algorithm (e.g. Koller and Friedman, 2009, Chap. 10)

$$A_{\psi/\mu}, \{P_v \text{ for } v \in \mathcal{V}\}, \{P_{v,v'} \text{ for } (v, v') \in \mathcal{E}\} \leftarrow \textsc{SumProduct}\left(\tfrac{1}{\mu}\psi(\cdot\,; \boldsymbol{w}), \mathcal{G}\right) .$$

2: Set $f_{-\mu H}(\boldsymbol{w}) \leftarrow \mu A_{\psi/\mu}$ and

$$\nabla f_{-\mu H}(\boldsymbol{w}) \leftarrow \sum_{v \in \mathcal{V}} \sum_{y_v \in \mathcal{Y}_v} P_v(y_v) \nabla \psi_v(y_v; \boldsymbol{w}) + \sum_{(v,v') \in \mathcal{E}} \sum_{y_v \in \mathcal{Y}_v} \sum_{y_{v'} \in \mathcal{Y}_{v'}} P_{v,v'}(y_v, y_{v'}) \nabla \psi_{v,v'}(y_v; \boldsymbol{w}) .$$

3: **return** $f_{-\mu H}(\boldsymbol{w}), \nabla f_{-\mu H}(\boldsymbol{w})$.

column π_{v-1}. Compare this to update (2.21) of the top-K Viterbi algorithm. In this case, the exp oracle is implemented by the forward-backward algorithm, a specialization of the sum-product algorithm to chain graphs.

Complexity of Inference Oracles. The next proposition presents the correctness guarantee and complexity of each of the aforementioned algorithms. The full proofs may be found in (Pillutla et al., 2018, Appendix B).

Proposition 2.11. *Consider as inputs an augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$ defined on a tree-structured graph $\mathcal{G}$, an integer $K > 0$ and a smoothing parameter $\mu > 0$.*

(i) *The output $(\psi^*, \boldsymbol{y}^*)$ of the max-product algorithm (the Viterbi algorithm for the special case when $\mathcal{G}$ is chain structured) satisfies $\psi^* = \psi(\boldsymbol{y}^*; \boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$. Thus, the pair $\big(\psi^*, \nabla\psi(\boldsymbol{y}^*; \boldsymbol{w})\big)$ is a correct implementation of the max oracle. It requires time $O(p \max_{v \in \mathcal{V}} |\mathcal{Y}_v|^2)$ and space $O(p \max_{v \in \mathcal{V}} |\mathcal{Y}_v|)$.*

(ii) *The output* $\{\psi^{(k)}, \boldsymbol{y}^{(k)}\}_{k=1}^{K}$ *of the top-*K *max-product algorithm (Algorithm 2.1 for the special case when* $\mathcal{G}$ *is chain structured) satisfies* $\psi^{(k)} = \psi(\boldsymbol{y}^{(k)}) = \max_{\boldsymbol{y}\in\mathcal{Y}}^{(k)} \psi(\boldsymbol{y})$*. Thus, the top-*$K$ *max-product algorithm followed by a projection onto the simplex (Algorithm A.1 in Appendix A.1) is a correct implementation of the top-*K *oracle. It requires time* $O(pK \log K \max_{v\in\mathcal{V}}|\mathcal{Y}_v|^2)$ *and space* $O(pK \max_{v\in\mathcal{V}}|\mathcal{Y}_v|)$*.*

(iii) *Algorithm 2.2 returns* $\big(f_{-\mu H}(\boldsymbol{w}), \nabla f_{-\mu H}(\boldsymbol{w})\big)$*. Thus, Algorithm 2.2 is a correct implementation of the exp oracle. It requires time* $O(p \max_{v\in\mathcal{V}}|\mathcal{Y}_v|^2)$ *and space* $O(p \max_{v\in\mathcal{V}}|\mathcal{Y}_v|)$*.*

Comparison to Differentiable Dynamic Programming. We now compare this form of smoothing to the differentiable dynamic programming approach (Mensch and Blondel, 2018).

Consider a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of N nodes $\mathcal{V} = \{1, \cdots, N\}$ and let P_i denote the set of parents of node i. We make our comparison for a specific parameterization — let $\boldsymbol{w} \in \mathbb{R}^d$ with $d = \sum_{i=1}^{n} |P_i|$ denote the transition scores. Denoting the score of the edge (j, i) by $w_{j,i}$, the total score of a path $j_1, \cdots, j_p$ is then given by $w_{j_1,j_2} + \cdots + w_{j_{p-1},j_p}$.

The score $f(\boldsymbol{w})$ of the best path can be computed by dynamic programming. The dynamic programming recursion maintains $f_i(\boldsymbol{w})$, which stores the cost of the best path prefix ending in node i and is updated as

$$f_0(\boldsymbol{w}) = 0\,, \quad f_i(\boldsymbol{w}) = \max_{j\in P_i} \{f_j(\boldsymbol{w}) + w_{j,i}\}\,, \quad f(\boldsymbol{w}) = f_N(\boldsymbol{w})\,.$$

In this parameterization, the ℓ_2^2 smoothing we consider in this chapter depends on the set $\mathcal{Y}$ of paths in the graph $\mathcal{G}$. In general graphs, its size $m = |\mathcal{Y}|$ is exponential in N. Let $\boldsymbol{A} : \mathbb{R}^d \to \mathbb{R}^m$ be a linear map such that $[\boldsymbol{A}\boldsymbol{w}]_j$ gives score of the j^{th} path. Our ℓ_2^2-smoothing is defined as

$$f_{\mu\ell_2^2}(\boldsymbol{w}) = \max_{\boldsymbol{u}\in\Delta^{m-1}} \Big\{\langle \boldsymbol{u}, \boldsymbol{A}\boldsymbol{w}\rangle - \frac{\mu}{2}\|\boldsymbol{w}\|_2^2\Big\}\,.$$

In contrast, the differentiable dynamic programming approach of (Mensch and Blondel, 2018) smooths each step of the dynamic program. In particular, for a smoothing parameter $\mu > 0$, their approach maintains $f_{i,\mu}(\boldsymbol{w})$, as the smoothed score of the path prefix ending in node i, given by

$$\begin{aligned} f_0(\boldsymbol{w}) &= 0\,, \\ f_i(\boldsymbol{w}) &= \max_{\boldsymbol{u}\in\Delta^{|P_i|-1}} \left\{\left\langle \boldsymbol{u}, \Big(f_j(\boldsymbol{w}) + w_{j,i}\Big)_{j\in P_i}\right\rangle - \frac{\mu}{2}\|\boldsymbol{u}\|_2^2\right\} \\ f_{\mathrm{DP},\mu}(\boldsymbol{w}) &= f_{N,\mu}(\boldsymbol{w})\,. \end{aligned} \tag{2.22}$$

We now compute the smoothness of this smooth variant f_{DP}, see Section A.3 for a proof.

Proposition 2.12. *At a smoothing parameter* $\mu > 0$*, we have that* $f_{\mathrm{DP},\mu}$ *as defined in* (2.22) *is* 1*-Lipschitz globally. Further, it satisfies,*

$$\|\nabla f_{\mathrm{DP},\mu}(\boldsymbol{w}) - \nabla f_{\mathrm{DP},\mu}(\boldsymbol{w}')\|_2 \le \frac{1}{\mu}\sum_{i=1}^{N} |P_i'(\boldsymbol{w}) \cup P_i'(\boldsymbol{w}')|\, \|\boldsymbol{w} - \boldsymbol{w}'\|_2\,,$$

where $P_i'(\boldsymbol{w})$ *denotes the "active" parents of node* i *at* $\boldsymbol{w}$*, i.e.,* $P_i'(\boldsymbol{w}) = \Big\{j \in P_i \,:\, [\boldsymbol{u}_\star^{(i)}(\boldsymbol{w})]_j \neq 0\Big\}$*, where* $\boldsymbol{u}_\star^{(i)}(\boldsymbol{w})$ *denotes the argmax in the definition of* $f_i(\boldsymbol{w})$ *in* (2.22)*. In particular,* $f_{\mathrm{DP},\mu}$ *is* $\sum_{i=1}^{N} |P_i|/\mu$*-smooth w.r.t.* $\|\cdot\|_2$ *globally.*

Overall, the differences between the two approaches can be summarized as:

(a) *Approximation Quality*: We have, $|f_{\mathrm{DP},\mu}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq N\mu/2$ while our ℓ_2^2 smoothing satisfies $|f_{\mu\ell_2^2}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq \mu/2$. That is, the smoothing considered here is N times tighter. For sequences of length p over a finite alphabet of size r, we have that $N = pr^2$.

(b) *Computational Complexity*: The computational complexity of computing $f_{\mathrm{DP},\mu}(\boldsymbol{w})$ and its gradient is the same as that of the max oracle. However, the complexity of $f_{\mu\ell_2^2}$ using the top-K strategy is $\widetilde{O}(K)$ times that of the max oracle, since we need to find the K best outputs.

(c) *Smoothness*: Proposition 2.12 bounds the smoothness of $f_{\mathrm{DP},\mu}$ as $\sum_{i\in[N]} |P_i|/\mu$. While a worst-case global bound on the smoothness of $f_{\mu\ell_2^2}$ can be much worse, Lemma 2.4 shows that its local behavior is much better. We have,

$$\|\nabla f_{\mu\ell_2^2}(\boldsymbol{w}) - \nabla f_{\mu\ell_2^2}(\boldsymbol{w}')\|_2 \leq K_{\boldsymbol{w},\boldsymbol{w}'} \|\boldsymbol{w} - \boldsymbol{w}'\|_2 \,,$$

where $K_{\boldsymbol{w},\boldsymbol{w}'}$ is the combined size of the supports of the $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{A}\boldsymbol{w}/\mu)$ and $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{A}\boldsymbol{w}'/\mu)$. Similarly, the Lipschitz constant of $\nabla f_{\mathrm{DP},\mu}$ depends locally on the sum of the number of "active parents" for each node. These two quantities do not appear to admit a straightforward comparison.

(d) *Generality*: The function $f_{\mathrm{DP},\mu}$ is tailored to dynamic programs, while the ℓ_2^2 smoothing we consider is more general and is applicable whenever we have an oracle for the K best solutions of inference. We see concrete examples for loopy graphs in the next section.

2.4.2 Inference Oracles in Loopy Graphs

For general loopy graphs with high tree-width, the inference problem (2.2) is NP-hard (Cooper, 1990). In particular cases, graph cut, matching or search algorithms can be used for exact inference in dense loopy graphs, and therefore, to implement the max oracle as well (cf. Lemma 2.6(i)). In each of these cases, we find that the top-K oracle can be implemented, but the exp oracle is intractable. We refer to (Pillutla et al., 2018, Appendix C) for a review of the algorithms and guarantees referenced in this section, as well as full proofs.

Inference Oracles using Max-Marginals. We recall a *max-marginal*, which is a constrained maximum of the augmented score ψ.

Definition 2.13. The max-marginal of ψ relative to a variable y_v is defined, for $j \in \mathcal{Y}_v$ as

$$\psi_{v;j}(\boldsymbol{w}) := \max_{\boldsymbol{y}\in\mathcal{Y}\,:\, y_v = j} \psi(\boldsymbol{y}; \boldsymbol{w}) \,. \tag{2.23}$$

In cases where exact inference is tractable using graph cut or matching algorithms, it is possible to extract max-marginals as well. This, as we shall see next, allows the implementation of the max and top-K oracles.

When the augmented score function ψ is *unambiguous*, i.e., no two distinct $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$ have the same augmented score, the output $\boldsymbol{y}^*(\boldsymbol{w})$ is unique and can be decoded from the max-marginals as (Pearl, 1988; Dawid, 1992)

$$y_v^*(\boldsymbol{w}) = \arg\max_{j\in\mathcal{Y}_v} \psi_{v;j}(\boldsymbol{w}) \,. \tag{2.24}$$

If one has access to an algorithm $\mathcal{M}$ that can compute max-marginals, the top-K oracle is also easily implemented via the *Best Max-Marginal First (BMMF)* algorithm (Yanover and Weiss, 2004). This algorithm requires computations of $2K$ sets of max-marginals, where a *set* of max-marginals refers to max-marginals for all y_v in $\boldsymbol{y}$. Therefore, the BMMF algorithm followed by a projection onto the simplex is a correct implementation of the top-K oracle at a computational cost of $2K$ sets of max-marginals.

Graph Cut and Matching Inference. Submodular energy functions over binary variables can be efficiently minimized exactly via a minimum cut algorithm (Kolmogorov and Zabin, 2004), while bipartite matching can be used for inference in a class of alignment problems (e.g., Taskar et al., 2005). In both these cases, max-marginals can be computed exactly and efficiently by combinatorial algorithms. This gives us a way to implement the max and top-K oracles. However, in both settings, computing the log-partition function $A_\psi(\boldsymbol{w})$ of a CRF with score ψ is known to be #P-complete (Jerrum and Sinclair, 1993). Proposition 2.8 immediately extends this result to the exp oracle. This discussion is summarized by the following proposition.

Proposition 2.14. *Consider as inputs an augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$, an integer $K > 0$ and a smoothing parameter $\mu > 0$. Further, suppose that ψ is unambiguous, that is, $\psi(\boldsymbol{y}'; \boldsymbol{w}) \neq \psi(\boldsymbol{y}''; \boldsymbol{w})$ for all distinct $\boldsymbol{y}', \boldsymbol{y}'' \in \mathcal{Y}$. Consider one of the two settings:*

(A) *the output space $\mathcal{Y}_v = \{0, 1\}$ for each $v \in \mathcal{V}$, and the function $-\psi$ is submodular, or,*

(B) *the augmented score corresponds to an alignment task where the inference problem* (2.2) *corresponds to a maximum weight bipartite matching.*

In these cases, we have the following:

(i) *The max oracle can be implemented at a computational complexity of $O(p)$ minimum cut computations in Case* (A)*, and in time $O(p^3)$ in Case* (B)*.*

(ii) *The top-K oracle can be implemented at a computational complexity of $O(pK)$ minimum cut computations in Case* (A)*, and in time $O(p^3 K)$ in Case* (B)*.*

(iii) *The exp oracle is #P-complete in both cases.*

Proposition 2.14 is loose in that the max oracle can be implemented with just one minimum cut computation instead of p in in Case (A) (Kolmogorov and Zabin, 2004).

Branch and Bound Search. Max oracles implemented via search algorithms can often be extended to implement the top-K oracle. We restrict our attention to the best-first branch and bound search such as the celebrated Efficient Subwindow Search (Lampert et al., 2008).

Branch and bound methods partition the search space into disjoint subsets while keeping an upper bound $\widehat{\psi} : \mathcal{X} \times 2^{\mathcal{Y}} \to \mathbb{R}$, on the maximal augmented score for each of the subsets $\widehat{\mathcal{Y}} \subseteq \mathcal{Y}$. Using a best-first strategy, promising parts of the search space are explored first. Parts of the search space whose upper bound indicates that they cannot contain the maximum do not have to be examined further.

The top-K oracle is implemented by simply continuing the search procedure until K outputs have been produced — see Algorithm A.2 in Appendix A.2. Both the max oracle and the top-K oracle can degenerate to an exhaustive search in the worst case, so we do not have sharp running time guarantees. However, we have the following correctness guarantee.

Proposition 2.15. *Consider an augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$, an integer $K > 0$ and a smoothing parameter $\mu > 0$. Suppose the upper bound function $\widehat{\psi}(\cdot, \cdot; \boldsymbol{w}) : \mathcal{X} \times 2^{\mathcal{Y}} \to \mathbb{R}$ satisfies the following properties:*

(a) $\widehat{\psi}(\widehat{\mathcal{Y}}; \boldsymbol{w})$ is finite for every $\widehat{\mathcal{Y}} \subseteq \mathcal{Y}$,

(b) $\widehat{\psi}(\widehat{\mathcal{Y}}; \boldsymbol{w}) \geq \max_{\boldsymbol{y} \in \widehat{\mathcal{Y}}} \psi(\boldsymbol{y}; \boldsymbol{w})$ for all $\widehat{\mathcal{Y}} \subseteq \mathcal{Y}$, and,

(c) $\widehat{\psi}(\{\boldsymbol{y}\}; \boldsymbol{w}) = \psi(\boldsymbol{y}; \boldsymbol{w})$ for every $\boldsymbol{y} \in \mathcal{Y}$.

Then, we have the following:

(i) *Algorithm A.2 with $K = 1$ is a correct implementation of the max oracle.*

(ii) *Algorithm A.2 followed by a projection onto the simplex (Algorithm A.1 in Appendix A.1) is a correct implementation of the top-K oracle.*

The discrete structure that allows inference via branch and bound search cannot be leveraged to implement the exp oracle.

Comparison to SparseMAP. We now compare the ℓ_2^2-smoothing we defined to SparseMAP (Niculae et al., 2018).

Suppose that the structure of output $\boldsymbol{y} \in \mathcal{Y}$ is encoded by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The output space $\mathcal{Y}$ is the space of all labelings of the graph $\mathcal{G}$ from a base label set Y, so that $m = |\mathcal{Y}| = |\mathcal{V}|^{|Y|}$. We consider SparseMAP in the node and edge score parameterization. Let the score of label j for node $v \in \mathcal{V}$ be given by parameter $w_{v,j}$, while the score of the label pair (j, j') for edge $(v, v') \in \mathcal{E}$ be given by $w_{v,v',j,j'}$, and let $\boldsymbol{w} \in \mathbb{R}^d$ denote the concatenation of all these node and edge scores into a vector. Note that $d = |\mathcal{V}||Y| + |\mathcal{E}||Y|^2$ is the total dimension. The score of each output configuration $\boldsymbol{y}$ is given by

$$\psi(\boldsymbol{y}; \boldsymbol{w}) = \sum_{v \in \mathcal{V}} w_{v,y_v} + \sum_{v,v' \in \mathcal{E}} w_{v,v',y_v,y_{v'}} \,.$$

Then, the inference problem corresponds to $f(\boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$, and the ℓ_2^2 smoothing we considered is

$$f_{\mu \ell_2^2}(\boldsymbol{w}) = \max_{\boldsymbol{u} \in \Delta^{|\mathcal{Y}|-1}} \left\{ \left\langle \boldsymbol{u}, (\psi(\boldsymbol{y}; \boldsymbol{w}))_{\boldsymbol{y} \in \mathcal{Y}} \right\rangle - \frac{\mu}{2} \|\boldsymbol{u}\|^2 \right\} ,$$

SparseMAP relies on a vector representation of outputs as the one-hot encoding of each node and edge — denote this encoding by $\varphi : \mathcal{Y} \to \mathbb{R}^d$. The resulting polytope $\mathcal{Z} = \operatorname{conv} \varphi(\mathcal{Y}) \subset \mathbb{R}^d$ is known as the *marginal polytope*. Since the maximum of a linear function over a polytope occurs at one of its vertices, the inference problem can also be written as

$$f(\boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w}) = \max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{z}, \boldsymbol{w} \rangle \,.$$

For a smoothing parameter $\mu > 0$, SparseMAP performs the smoothing

$$f_{\mathrm{SpMap},\mu}(\boldsymbol{w}) = \max_{\boldsymbol{z} \in \mathcal{Z}} \left\{ \langle \boldsymbol{z}, \boldsymbol{w} \rangle - \frac{\mu}{2} \|\boldsymbol{z}\|^2 \right\} \tag{2.25}$$

$$= \max_{\boldsymbol{u} \in \Delta^{|\mathcal{Y}|-1}} \left\{ \left\langle \boldsymbol{u}, (\psi(\boldsymbol{y}; \boldsymbol{w}))_{\boldsymbol{y} \in \mathcal{Y}} \right\rangle - \frac{\mu}{2} \|\mathbb{E}_{\boldsymbol{y} \sim \boldsymbol{u}}[\varphi(\boldsymbol{y})]\|^2 \right\} , \tag{2.26}$$

where we interpret $\boldsymbol{u} \in \Delta^{|\mathcal{Y}|-1}$ as a distribution over $\mathcal{Y}$. Note that SpareMAP's regularization is different from that of $f_{\mu \ell_2^2}$. We have the following smoothness and approximation properties.

Proposition 2.16. *For a smoothing parameter $\mu > 0$, we have that $f_{\mathrm{SpMap},\mu}$ defined in* (2.25) *over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is* 1*-Lipschitz and* $1/\mu$*-smooth w.r.t. $\|\cdot\|_2$. Moreover, its approximation quality is*

$$0 \leq |f_{\mathrm{SpMap},\mu}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq \frac{\mu}{2}(|\mathcal{V}| + |\mathcal{E}|)\,.$$

Proof. The Lipschitzness is obvious and smoothness follows from Property A.2. Since the maximum of the convex function $\|\cdot\|_2^2$ over a convex polytope $\mathcal{Z}$ occurs at one of its corners, and each corner of the marginal polytope is defined by some $\boldsymbol{y} \in \mathcal{Y}$ (cf. Wainwright and Jordan, 2008), we have from Property A.2that

$$0 \leq |f_{\mathrm{SpMap},\mu}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq \frac{\mu}{2} \max_{\boldsymbol{z}\in\mathcal{Z}} \|\boldsymbol{z}\|_2^2 = \frac{\mu}{2} \max_{\boldsymbol{y}\in\mathcal{Y}} \|\varphi(\boldsymbol{y})\|_2^2 = \frac{\mu}{2}(|\mathcal{V}| + |\mathcal{E}|)\,.$$

Note that we used $\|\varphi(\boldsymbol{y})\|_2^2 = |\mathcal{V}| + |\mathcal{E}|$ owing to the one-hot representation. □

We now give an overall comparison of SparseMAP to our smoothing.

(a) *Sparsity*: Both formulations have the property of returning a sparse distribution $\boldsymbol{u} \in \Delta^{|\mathcal{Y}|-1}$ over outputs. We are unaware of any theoretical bound on the sparsity of SparseMAP, while we bound the sparsity of the ℓ_2^2 smoothing we consider in Proposition 2.3.

(b) *Approximation Quality*: Proposition 2.16 gives that $|f_{\mathrm{SpMap},\mu}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq (|\mathcal{V}| + |\mathcal{E}|)\mu/2$, while our ℓ_2^2 smoothing satisfies $|f_{\mu\ell_2^2}(\boldsymbol{w}) - f(\boldsymbol{w})| \leq \mu/2$. For sequences of length p over a finite alphabet Y, we have that $f_{\mathrm{SpMap},\mu}$ is a uniform $O(p|Y|^2\mu)$-approximation to f.

(c) *Computational Complexity*: By the use of the top-K strategy and controlling the smoothing parameter μ, the smoothing $f_{\mu\ell_2^2}$ and its gradient can be computed in $\widetilde{O}(K)$ times the complexity of the max oracle, where K is a small integer fixed a priori (cf. Table 2.2). On the other hand, SparseMAP is computed using a fully-corrective variant of the Frank-Wolfe algorithm. To obtain an ε-approximation, SparseMAP requires at most $d^\gamma \log(1/\varepsilon)$ calls to a max oracle, where $\gamma \geq 1$ is a constant known as *eccentricity*, that depends on the structure of the marginal polytope $\operatorname{conv}\varphi(\mathcal{Y}) \subset \mathbb{R}^d$ (Lacoste-Julien and Jaggi, 2015).[2] For instance, for a sequence of length p from a dictionary Y, the dimensionality of the representation is $d = p|Y| + (p-1)|Y|^2$. Thus, the worst-case computational complexity of SparseMAP is larger than that of $f_{\mu\ell_2^2}$ for $K = O(p|Y|^2)$. In practice, we find that $K = 5$ or 10 suffices, so this is always the case.

(d) *Smoothness*: Proposition 2.16 showed that $f_{\mathrm{SpMap},\mu}$ is $1/\mu$-smooth w.r.t. $\|\cdot\|_2$. While a naïve worst-case bound on the smoothness of $f_{\mu\ell_2^2}$ can be much worse, Lemma 2.4 shows that its local behavior is much better. The Lipschitz constant of $\nabla f_{\mu\ell_2^2}(\boldsymbol{w})$ scales as K/μ, where K is the sparsity of $f_{\mu\ell_2^2}$ at $\boldsymbol{w}$.

(e) *Generality*: SparseMAP requires a max oracle and a tractable lower dimensional representation of the outputs in terms if factor graphs. On the other hand, the top-K strategy to compute $f_{\mu\ell_2^2}$ requires a top-K oracle. While the former appears more general a priori, we show that the latter is implementable in several cases of interest. We also remark that our approach is applicable when a factor graph representation of the outputs is not obvious, e.g., inference by branch and bound search (Lampert et al., 2008).

[2] Determining sharp bounds on the eccentricity of the marginal polytope, which determines the rate of convergence, is an open problem (Lacoste-Julien and Jaggi, 2015). The eccentricity of the simplex Δ^{d-1} is d, which is the smallest possible for any polytope, while that of the unit cube $[0,1]^d$ is d^2.

2.5 Smoothable Convex Optimization: the Casimir Algorithm

We now come back to the optimization problem (2.1) with $f^{(i)}$ defined in (2.12). We assume in this section that the mappings $\boldsymbol{g}^{(i)}$ defined in (2.11) are affine. Problem (2.1) now reads

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \left[F(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} h(\boldsymbol{A}^{(i)}\boldsymbol{w} + \boldsymbol{b}^{(i)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \right] . \tag{2.27}$$

For a single input ($n = 1$), the problem reads

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} h(\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2. \tag{2.28}$$

where h is a simple non-smooth convex function and $\lambda \geq 0$. Nesterov (2005b;a) first analyzed such setting: while the problem suffers from its non-smoothness, fast methods can be developed by considering smooth approximations of the objectives. We combine this idea with the Catalyst acceleration scheme (Lin et al., 2018) to accelerate a linearly convergent smooth optimization algorithm resulting in a scheme called *Casimir*.

2.5.1 Casimir: Catalyst with Smoothing

The Catalyst (Lin et al., 2018) approach minimizes regularized objectives centered around the current iterate. It proceeds by computing approximate proximal point steps instead of the classical (sub)-gradient steps. A proximal point step from a point $\boldsymbol{w}$ with step-size κ^{-1} is defined as the minimizer of

$$\min_{\boldsymbol{z}\in\mathbb{R}^m} F(\boldsymbol{z}) + \frac{\kappa}{2}\|\boldsymbol{z} - \boldsymbol{w}\|_2^2, \tag{2.29}$$

which can also be seen as a gradient step on the Moreau envelope of F — see (Lin et al., 2018) for a detailed discussion. While solving the subproblem (2.29) might be as hard as the original problem we only require an approximate solution returned by a given optimization method $\mathcal{M}$. The Catalyst approach is then an inexact accelerated proximal point algorithm that carefully mixes approximate proximal point steps with the extrapolation scheme of Nesterov (1983). The Casimir scheme extends this approach to non-smooth optimization.

For the overall method to be efficient, subproblems (2.29) must have low complexity. That is, there must exist an optimization algorithm $\mathcal{M}$ that solves them linearly. For the Casimir approach to be able to handle non-smooth objectives, it means that we need not only to regularize the objective but also to smooth it. To this end we define

$$F_{\mu\omega}(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} h_{\mu\omega}(\boldsymbol{A}^{(i)}\boldsymbol{w} + \boldsymbol{b}^{(i)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

as a smooth approximation of the objective F, and,

$$F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z}) := \frac{1}{n}\sum_{i=1}^{n} h_{\mu\omega}(\boldsymbol{A}^{(i)}\boldsymbol{w} + \boldsymbol{b}^{(i)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{\kappa}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2$$

a smooth and regularized approximation of the objective centered around a given point $\boldsymbol{z} \in \mathbb{R}^d$. While the original Catalyst algorithm considered a fixed regularization term κ, we vary κ and μ along the iterations. This enables us to get adaptive smoothing strategies.

The overall method is presented in Algorithm 2.3. We first analyze in Section 2.5.2 its complexity for a generic linearly convergent algorithm $\mathcal{M}$. Thereafter, in Section 2.5.3, we compute the total complexity with SVRG (Johnson and Zhang, 2013) as $\mathcal{M}$. Before that, we specify two practical aspects of the implementation: a proper stopping criterion (2.31) and a good initialization of subproblems (Line 3).

Stopping Criterion. We solve subproblem k in Line 3 to a degree of relative accuracy specified by $\delta_k \in [0, 1)$. In view of the $(\lambda+\kappa_k)$-strong convexity of $F_{\mu_k\omega,\kappa_k}(\cdot\,; \boldsymbol{z}_{k-1})$, the functional gap can be controlled by the norm of the gradient. In particular, $\|\nabla F_{\mu_k\omega,\kappa_k}(\widehat{\boldsymbol{w}}; \boldsymbol{z}_{k-1})\|_2^2 \leq (\lambda+\kappa_k)\delta_k\kappa_k\|\widehat{\boldsymbol{w}} - \boldsymbol{z}_{k-1}\|_2^2$ is a sufficient condition for the stopping criterion (2.31).

A practical alternate stopping criterion used in (Lin et al., 2018) is to fix an iteration budget T_{budget} and run the inner solver $\mathcal{M}$ for exactly T_{budget} steps. We do not have a theoretical analysis for this scheme but find that it works well in experiments.

Warm Start of Subproblems. The rate of convergence of first order optimization algorithms depends on the initialization and we must warm start $\mathcal{M}$ at an appropriate initial point in order to obtain the best convergence of subproblem (2.30) in Line 3 of Algorithm 2.3. We advocate the use of the prox center $\boldsymbol{z}_{k-1}$ in iteration k as the warm start strategy. We also experiment with other warm start strategies in Section 2.6.

2.5.2 Convergence of Casimir

We first state the outer loop complexity results of Algorithm 2.3 for any generic linearly convergent algorithm $\mathcal{M}$. Then, we consider the complexity of each inner optimization problem (2.30) based on properties of $\mathcal{M}$.

Outer Loop Complexity Results. The following theorem states the convergence of the algorithm for general choice of parameters, where we denote $\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w})$ and $F^* = F(\boldsymbol{w}^*)$.

Theorem 2.17. *Consider Problem* (2.27). *Suppose $\delta_k \in [0, 1)$ for all $k \geq 1$, the sequence $(\mu_k)_{k\geq 1}$ is non-negative and non-increasing, and the sequence $(\kappa_k)_{k\geq 1}$ is strictly positive and non-decreasing. Further, suppose the smoothing function $\omega : \operatorname{dom} h^* \to \mathbb{R}$ satisfies $-D_\omega \leq \omega(\boldsymbol{u}) \leq 0$ for all $\boldsymbol{u} \in \operatorname{dom} h^*$ and that $\alpha_0^2 \geq \lambda/(\lambda+\kappa_1)$. Then, the sequence $(\alpha_k)_{k\geq 0}$ generated by Algorithm 2.3 satisfies $0 < \alpha_k \leq \alpha_{k-1} < 1$ for all $k \geq 1$. Furthermore, the sequence $(\boldsymbol{w}_k)_{k\geq 0}$ of iterates generated by Algorithm 2.3 satisfies*

$$F(\boldsymbol{w}_k) - F^* \leq \frac{\mathcal{A}_0^{k-1}}{\mathcal{B}_1^k}\Delta_0 + \mu_k D_\omega + \sum_{j=1}^{k} \frac{\mathcal{A}_j^{k-1}}{\mathcal{B}_j^k}\left(\mu_{j-1} - (1-\delta_j)\mu_j\right) D_\omega\,, \tag{2.35}$$

where $\mathcal{A}_i^j := \prod_{r=i}^{j}(1-\alpha_r)$, $\mathcal{B}_i^j := \prod_{r=i}^{j}(1-\delta_r)$, $\Delta_0 := F(\boldsymbol{w}_0) - F^* + \frac{(\kappa_1+\lambda)\alpha_0^2 - \lambda\alpha_0}{2(1-\alpha_0)}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2$ and $\mu_0 := 2\mu_1$.

The proof is given Appendix A.5. The key innovation is blending the adaptive smoothing and Moreau-Yosida regularization into the analysis of acceleration via extrapolation (Lin et al., 2018).

We now instantiate the rates implied by this theorem in a number of concrete settings, although with the parameter choices — these are summarized in Table 2.3. Overall, the target accuracy sequences δ_k is chosen such that $\mathcal{B}_j^k$ is a constant and the parameters μ_k and κ_k are then carefully chosen for an almost parameter-free algorithm with the right rate of convergence.

Algorithm 2.3. The Casimir algorithm

Input: Smoothable objective F of the form (2.28) with h simple, smoothing function ω, linearly convergent algorithm $\mathcal{M}$, a non-negative and non-increasing sequence of smoothing parameters $(\mu_k)_{k\geq 1}$, positive and non-decreasing sequence of regularization parameters $(\kappa_k)_{k\geq 1}$, non-negative sequence of relative target accuracies $(\delta_k)_{k\geq 1}$ and, initial point $\boldsymbol{w}_0$, $\alpha_0 \in (0,1)$, time horizon K.

Initialize: $\boldsymbol{z}_0 = \boldsymbol{w}_0$.

for $k = 1, \cdots, K$ **do**

Using $\mathcal{M}$ with $\boldsymbol{z}_{k-1}$ as the starting point, find $\boldsymbol{w}_k \approx \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} F_{\mu_k\omega,\kappa_k}(\boldsymbol{w}; \boldsymbol{z}_{k-1})$ where

$$F_{\mu_k\omega,\kappa_k}(\boldsymbol{w}; \boldsymbol{z}_{k-1}) := \frac{1}{n}\sum_{i=1}^{n} h_{\mu_k\omega}(\boldsymbol{A}^{(i)}\boldsymbol{w} + \boldsymbol{b}^{(i)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{\kappa_k}{2}\|\boldsymbol{w} - \boldsymbol{z}_{k-1}\|_2^2 \tag{2.30}$$

such that

$$F_{\mu_k\omega,\kappa_k}(\boldsymbol{w}_k; \boldsymbol{z}_{k-1}) - \min_{\boldsymbol{w}} F_{\mu_k\omega,\kappa_k}(\boldsymbol{w}; \boldsymbol{z}_{k-1}) \leq \tfrac{\delta_k\kappa_k}{2}\|\boldsymbol{w}_k - \boldsymbol{z}_{k-1}\|_2^2 \tag{2.31}$$

Solve for $\alpha_k \geq 0$

$$\alpha_k^2(\kappa_{k+1} + \lambda) = (1 - \alpha_k)\alpha_{k-1}^2(\kappa_k + \lambda) + \alpha_k\lambda. \tag{2.32}$$

Set

$$\boldsymbol{z}_k = \boldsymbol{w}_k + \beta_k(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}), \tag{2.33}$$

where

$$\beta_k = \frac{\alpha_{k-1}(1 - \alpha_{k-1})(\kappa_k + \lambda)}{\alpha_{k-1}^2(\kappa_k + \lambda) + \alpha_k(\kappa_{k+1} + \lambda)}. \tag{2.34}$$

return $\boldsymbol{w}_K$.

The first corollary considers the strongly convex case ($\lambda > 0$) with constant smoothing $\mu_k = \mu$, assuming that ε is known *a priori*. We note that this is, up to constants, the same complexity obtained by the original Catalyst scheme on a fixed smooth approximation $F_{\mu\omega}$ with $\mu = O(\varepsilon D_\omega)$.

Corollary 2.18. *Consider the setting of Theorem 2.17. Let $q = \lambda/(\lambda + \kappa)$. Suppose $\lambda > 0$ and $\mu_k = \mu$, $\kappa_k = \kappa$, for all $k \geq 1$. Choose $\alpha_0 = \sqrt{q}$ and, $\delta_k = \sqrt{q}/(2 - \sqrt{q})$. Then, we have,*

$$F(\boldsymbol{w}_k) - F^* \leq \frac{3 - \sqrt{q}}{1 - \sqrt{q}}\mu D_\omega + 2\left(1 - \frac{\sqrt{q}}{2}\right)^k (F(\boldsymbol{w}_0) - F^*) \ .$$

Next, we consider the strongly convex case where the target accuracy ε is not known in advance. We let smoothing parameters $(\mu_k)_{k\geq 0}$ decrease over time to obtain an adaptive smoothing scheme that gives progressively better surrogates of the original objective.

Corollary 2.19. *Consider the setting of Theorem 2.17. Suppose $\lambda > 0$ and $\kappa_k = \kappa$, for all $k \geq 1$. Let*

Table 2.3: Summary of outer iteration complexity for Algorithm 2.3 for different parameter settings. We use shorthand $\Delta F_0 := F(\boldsymbol{w}_0) - F^*$ and $\Delta_0 = \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2$. Absolute constants are omitted from the rates.

Cor.	$\lambda > 0$	κ_k	μ_k	δ_k	α_0	$F(\boldsymbol{w}_k) - F^*$	Remark
2.18	Yes	κ	μ	$\frac{\sqrt{q}}{2-\sqrt{q}}$	$\sqrt{q}$	$\left(1 - \frac{\sqrt{q}}{2}\right)^k \Delta F_0 + \frac{\mu D}{1-\sqrt{q}}$	$q = \frac{\lambda}{\lambda+\kappa}$
2.19	Yes	κ	$\mu\left(1 - \frac{\sqrt{q}}{2}\right)^{k/2}$	$\frac{\sqrt{q}}{2-\sqrt{q}}$	$\sqrt{q}$	$\left(1 - \frac{\sqrt{q}}{2}\right)^{k/2}\left(\Delta F_0 + \frac{\mu D}{1-\sqrt{q}}\right)$	$q = \frac{\lambda}{\lambda+\kappa}$
2.20	No	κ	μ	k^{-2}	c	$\frac{1}{k^2}\left(\Delta F_0 + \kappa\Delta_0^2\right) + \mu D$	$c = (\sqrt{5}-1)/2$
2.21	No	$\kappa\, k$	μ/k	k^{-2}	c	$\frac{\log k}{k}(\Delta F_0 + \kappa\Delta_0^2 + \mu D)$	$c = (\sqrt{5}-1)/2$

$q = \lambda/(\lambda+\kappa)$ and $\eta = 1 - \sqrt{q}/2$. Choose $\alpha_0 = \sqrt{q}$ and, the sequences $(\mu_k)_{k\geq 1}$ and $(\delta_k)_{k\geq 1}$ as

$$\mu_k = \mu\eta^{k/2}\,, \qquad \text{and,} \qquad \delta_k = \frac{\sqrt{q}}{2-\sqrt{q}}\,,$$

where $\mu > 0$ is any constant. Then, we have,

$$F(\boldsymbol{w}_k) - F^* \leq \eta^{k/2}\left[2\left(F(\boldsymbol{w}_0) - F^*\right) + \frac{\mu D_\omega}{1-\sqrt{q}}\left(2 - \sqrt{q} + \frac{\sqrt{q}}{1-\sqrt{\eta}}\right)\right]\,.$$

The next two corollaries consider the unregularized problem, i.e., $\lambda = 0$ with constant and adaptive smoothing respectively.

Corollary 2.20. *Consider the setting of Theorem 2.17. Suppose $\mu_k = \mu$, $\kappa_k = \kappa$, for all $k \geq 1$ and $\lambda = 0$. Choose $\alpha_0 = (\sqrt{5}-1)/2$ and $\delta_k = (k+1)^{-2}$. Then, we have,*

$$F(\boldsymbol{w}_k) - F^* \leq \frac{8}{(k+2)^2}\left(F(\boldsymbol{w}_0) - F^* + \frac{\kappa}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2\right) + \mu D_\omega\left(1 + \frac{12}{k+2} + \frac{30}{(k+2)^2}\right)\,.$$

Corollary 2.21. *Consider the setting of Theorem 2.17 with $\lambda = 0$. Choose $\alpha_0 = (\sqrt{5}-1)/2$, and for some non-negative constants κ, μ, define sequences $(\kappa_k)_{k\geq 1}, (\mu_k)_{k\geq 1}, (\delta_k)_{k\geq 1}$ as*

$$\kappa_k = \kappa\, k\,, \quad \mu_k = \frac{\mu}{k} \quad \text{and,} \quad \delta_k = \frac{1}{(k+1)^2}\,.$$

Then, for $k \geq 2$, we have,

$$F(\boldsymbol{w}_k) - F^* \leq \frac{\log(k+1)}{k+1}\left(2(F(\boldsymbol{w}_0) - F^*) + \kappa\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 + 27\mu D_\omega\right)\,.$$

For the first iteration (i.e., $k = 1$), this bound is off by a constant factor $1/\log 2$.

Inner Loop Complexity. Consider a class $\mathcal{F}_{L,\lambda}$ of functions defined as

$$\mathcal{F}_{L,\lambda} = \left\{f : \mathbb{R}^d \to \mathbb{R} \text{ such that } f \text{ is } L\text{-smooth and } \lambda\text{-strongly convex}\right\}\,.$$

We now formally define a linearly convergent algorithm on this class of functions.

Table 2.4: Summary of global complexity of Casimir-SVRG, i.e., Algorithm 2.3 with SVRG as the inner solver for various parameter settings. We show $\mathbb{E}[N]$, the expected total number of SVRG iterations required to obtain an accuracy ε, up to constants and factors logarithmic in problem parameters. We denote $\Delta F_0 := F(\boldsymbol{w}_0) - F^*$ and $\Delta_0 = \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2$. Constants D, A are short for D_ω, A_ω (see (2.37)).

Prop.	$\lambda > 0$	μ_k	κ_k	δ_k	$\mathbb{E}[N]$	Remark
2.25	Yes	ε/D	$AD/\varepsilon n - \lambda$	$\sqrt{\frac{\lambda\varepsilon n}{AD}}$	$n + \sqrt{\frac{ADn}{\lambda\varepsilon}}$	fix ε in advance
2.26	Yes	μc^k	λ	c'	$n + \frac{A}{\lambda\varepsilon}\frac{\Delta F_0 + \mu D}{\mu}$	$c, c' < 1$ are universal constants
2.27	No	ε/D	$AD/\varepsilon n$	$1/k^2$	$n\sqrt{\frac{\Delta F_0}{\varepsilon}} + \frac{\sqrt{ADn}\Delta_0}{\varepsilon}$	fix ε in advance
2.28	No	μ/k	$\kappa_0\, k$	$1/k^2$	$\frac{\widehat{\Delta}_0}{\varepsilon}\left(n + \frac{A}{\mu\kappa_0}\right)$	$\widehat{\Delta}_0 = \Delta F_0 + \frac{\kappa_0}{2}\Delta_0^2 + \mu D$

Definition 2.22. A first order algorithm $\mathcal{M}$ is said to be linearly convergent with parameters $C : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ and $\tau : \mathbb{R}_+ \times \mathbb{R}_+ \to (0, 1)$ if the following holds: for all $L \geq \lambda > 0$, and every $f \in \mathcal{F}_{L,\lambda}$ and $\boldsymbol{w}_0 \in \mathbb{R}^d$, $\mathcal{M}$ started at $\boldsymbol{w}_0$ generates a sequence $(\boldsymbol{w}_k)_{k\geq 0}$ that satisfies:

$$\mathbb{E}f(\boldsymbol{w}_k) - f^* \leq C(L,\lambda)\left(1 - \tau(L,\lambda)\right)^k \left(f(\boldsymbol{w}_0) - f^*\right), \tag{2.36}$$

where $f^* := \min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$ and the expectation is over the randomness of $\mathcal{M}$.

The parameter τ determines the rate of convergence of the algorithm. For instance, batch gradient descent is a deterministic linearly convergent algorithm with $\tau(L,\lambda)^{-1} = L/\lambda$ and incremental algorithms such as SVRG and SAGA satisfy (2.36) with $\tau(L,\lambda)^{-1} = c(n + L/\lambda)$ for some universal constant c.

The warm start strategy in step k of Algorithm 2.3 is to initialize $\mathcal{M}$ at the prox center $\boldsymbol{z}_{k-1}$. Since $\mathcal{M}$ is linearly convergent, we expect the inner optimization to take $\widetilde{O}(\tau(L,\lambda)^{-1})$ iterations. This is formalized in the next proposition, due to (Lin et al., 2018, Cor. 16).

Lemma 2.23. *Consider $F_{\mu\omega,\kappa}(\cdot\,;\boldsymbol{z})$ defined in Eq.* (2.30)*, and a linearly convergent algorithm $\mathcal{M}$ with parameters C, τ. Let $\delta \in [0,1)$. Suppose $F_{\mu\omega}$ is $L_{\mu\omega}$-smooth and λ-strongly convex. Then the expected number of iterations $\mathbb{E}[\widehat{T}]$ of $\mathcal{M}$ when started at $\boldsymbol{z}$ in order to obtain $\widehat{\boldsymbol{w}} \in \mathbb{R}^d$ that satisfies*

$$F_{\mu\omega,\kappa}(\widehat{\boldsymbol{w}};\boldsymbol{z}) - \min_{\boldsymbol{w}} F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z}) \leq \tfrac{\delta\kappa}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2$$

is upper bounded by

$$\mathbb{E}[\widehat{T}] \leq \frac{1}{\tau(L_{\mu\omega}+\kappa, \lambda+\kappa)} \log\left(\frac{8C(L_{\mu\omega}+\kappa,\lambda+\kappa)}{\tau(L_{\mu\omega}+\kappa,\lambda+\kappa)} \cdot \frac{L_{\mu\omega}+\kappa}{\kappa\delta}\right) + 1\,.$$

2.5.3 Casimir with SVRG

We now choose SVRG (Johnson and Zhang, 2013) to be the linearly convergent algorithm $\mathcal{M}$, resulting in an algorithm called Casimir-SVRG. The rest of this section analyzes the total iteration complexity of Casimir-SVRG to solve Problem (2.27). The proofs of the results from this section are calculations stemming

from combining the outer loop complexity from Corollary 2.18 to 2.21 with the inner loop complexity from Lemma 2.23, and are relegated to Appendix A.5.3. Table 2.4 summarizes the results of this section.

Recall that if ω is 1-strongly convex with respect to $\|\cdot\|_\alpha$, then $h_{\mu\omega}(\boldsymbol{A}\boldsymbol{w}+\boldsymbol{b})$ is $L_{\mu\omega}$-smooth with respect to $\|\cdot\|_2$, where $L_{\mu\omega} = \|\boldsymbol{A}\|_{2,\alpha}^2/\mu$. Therefore, the complexity of solving problem (2.27) will depend on

$$A_\omega := \max_{i=1,\cdots,n} \|\boldsymbol{A}^{(i)}\|_{2,\alpha}^2 \,. \tag{2.37}$$

Remark 2.24. *We have that $\|\boldsymbol{A}\|_{2,2} = \|\boldsymbol{A}\|_2$ is the spectral norm of $\boldsymbol{A}$ and $\|\boldsymbol{A}\|_{2,1} = \max_j \|\boldsymbol{a}_j\|_2$ is the largest row norm, where $\boldsymbol{a}_j$ is the jth row of $\boldsymbol{A}$. Moreover, we have that $\|\boldsymbol{A}\|_{2,2} \geq \|\boldsymbol{A}\|_{2,1}$.*

We start with the strongly convex case with constant smoothing.

Proposition 2.25. *Consider the setting of Theorem 2.17 and fix $\varepsilon > 0$. If we run Algorithm 2.3 with SVRG as the inner solver with parameters: $\mu_k = \mu = \varepsilon/10D_\omega$, $\kappa_k = k$ chosen as*

$$\kappa = \begin{cases} \frac{A}{\mu n} - \lambda \,, & \text{if } \frac{A}{\mu n} > 4\lambda \\ \lambda \,, & \text{otherwise} \end{cases} \,,$$

$q = \lambda/(\lambda+\kappa)$, $\alpha_0 = \sqrt{q}$, and $\delta = \sqrt{q}/(2-\sqrt{q})$. Then, the number of iterations N to obtain $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F(\boldsymbol{w}^*) \leq \varepsilon$ is bounded in expectation as

$$\mathbb{E}[N] \leq \widetilde{O}\left(n + \sqrt{\frac{A_\omega D_\omega n}{\lambda\varepsilon}}\right) \,.$$

Here, we note that κ was chosen to minimize the total complexity (cf. Lin et al. (2018)). This bound is known to be tight, up to logarithmic factors (Woodworth and Srebro, 2016). Next, we turn to the strongly convex case with decreasing smoothing.

Proposition 2.26. *Consider the setting of Theorem 2.17. Suppose $\lambda > 0$ and $\kappa_k = \kappa$, for all $k \geq 1$ and that α_0, $(\mu_k)_{k\geq 1}$ and $(\delta_k)_{k\geq 1}$ are chosen as in Corollary 2.19, with $q = \lambda/(\lambda+\kappa)$ and $\eta = 1 - \sqrt{q}/2$. If we run Algorithm 2.3 with SVRG as the inner solver with these parameters, the number of iterations N of SVRG required to obtain $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(n + \frac{A_\omega}{\mu(\lambda+\kappa)\varepsilon}\left(F(\boldsymbol{w}_0) - F^* + \frac{\mu D_\omega}{1-\sqrt{q}}\right)\right) \,.$$

Unlike the previous case, there is no obvious choice of κ, such as to minimize the global complexity. Notice that we do not get the accelerated rate of Proposition 2.25. We now turn to the case when $\lambda = 0$ and $\mu_k = \mu$ for all k.

Proposition 2.27. *Consider the setting of Theorem 2.17 and fix $\varepsilon > 0$. If we run Algorithm 2.3 with SVRG as the inner solver with parameters: $\mu_k = \mu = \varepsilon/20D_\omega$, $\alpha_0 = (\sqrt{5}-1)/2$, $\delta_k = 1/(k+1)^2$, and $\kappa_k = \kappa = A_\omega/\mu(n+1)$. Then, the number of iterations N to get a point $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(n\sqrt{\frac{F(\boldsymbol{w}_0) - F^*}{\varepsilon}} + \sqrt{A_\omega D_\omega n}\frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2}{\varepsilon}\right) \,.$$

This rate is tight up to log factors (Woodworth and Srebro, 2016). Lastly, we consider the non-strongly convex case ($\lambda = 0$) together with decreasing smoothing. As with Proposition 2.26, we do not obtain an accelerated rate here.

Proposition 2.28. *Consider the setting of Theorem 2.17. Suppose $\lambda = 0$ and that α_0, $(\mu_k)_{k\geq 1}$,$(\kappa_k)_{k\geq 1}$ and $(\delta_k)_{k\geq 1}$ are chosen as in Corollary 2.21. If we run Algorithm 2.3 with SVRG as the inner solver with these parameters, the number of iterations N of SVRG required to obtain $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(\frac{1}{\varepsilon}\left(F(\boldsymbol{w}_0) - F^* + \kappa\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 + \mu D\right)\left(n + \frac{A_\omega}{\mu\kappa}\right)\right).$$

2.6 Experiments

In this section, we study the experimental behavior of the proposed algorithms in two structured prediction tasks, namely named entity recognition and visual object localization. Recall that given training examples $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^n$, we wish to solve the problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d}\left[F(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^n \max_{\boldsymbol{y}'\in\mathcal{Y}(\boldsymbol{x}^{(i)})}\left\{\phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}'; \boldsymbol{w}) + \ell(\boldsymbol{y}^{(i)}, \boldsymbol{y}')\right\} - \phi(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{w})\right].$$

Note that we now allow the output space $\mathcal{Y}(\boldsymbol{x})$ to depend on the instance $\boldsymbol{x}$ - the analysis from the previous sections applies to this setting as well. In all the plots, the shaded region represents one standard deviation over ten random runs.

We compare the performance of various optimization algorithms based on the number of calls to a smooth inference oracle. Moreover, following literature for algorithms based on SVRG (Schmidt et al., 2017; Lin et al., 2018), we exclude the cost of computing the full gradients.

The results must be interpreted keeping in mind that the running time of all inference oracles is not the same. The ultimate yardstick to benchmark the performance of optimization algorithms is wall clock time. However, this depends heavily on implementation, system and ambient system conditions. With regards to the differing running times of different oracles, we find that a small value of K, e.g., 5 suffices, so that our highly optimized implementations of the top-K oracle incurs negligible running time penalties over the max oracle. Moreover, the computations of the batch gradient have been neglected as they are embarrassingly parallel.

The outline of the rest of this section is as follows. First, we describe the datasets and task description in Section 2.6.1, followed by methods compared in Section 2.6.2 and their hyperparameter settings in Section 2.6.3. Lastly, Section 2.6.4 presents the experimental studies.

2.6.1 Dataset and Task Description

For each of the tasks, we specify below the following: (a) the dataset $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^n$, (b) the output structure $\mathcal{Y}$, (c) the loss function ℓ, (d) the score function $\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$, (e) implementation of inference oracles, and lastly, (f) the evaluation metric used to assess the quality of predictions.

CoNLL 2003: Named Entity Recognition. Named entities are phrases that contain the names of persons, organizations, locations, etc., and the task is to predict the label (tag) of each entity. Named entity

recognition can be formulated as a sequence tagging problem where the set $\mathcal{Y}_{\text{tag}}$ of individual tags is of size 7.

Each datapoint $\boldsymbol{x}$ is a sequence of words $\boldsymbol{x} = (x_1, \cdots, x_p)$, and the label $\boldsymbol{y} = (y_1, \cdots, y_p) \in \mathcal{Y}(\boldsymbol{x})$ is a sequence of the same length, where each $y_i \in \mathcal{Y}_{\text{tag}}$ is a tag.

Loss Function. The loss function is the Hamming Loss $\ell(\boldsymbol{y}, \boldsymbol{y}') = \sum_i \mathbb{I}(y_i \neq y'_i)$.

Score Function. We use a chain graph to represent this task. In other words, the observation-label dependencies are encoded as a Markov chain of order 1 to enable efficient inference using the Viterbi algorithm. We only consider the case of linear score $\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$ for this task. The feature map Φ here is very similar to that given in Example 2.1. Following Tkachenko and Simanovsky (2012), we use local context $\Psi_i(\boldsymbol{x})$ around i^{th} word x_i of $\boldsymbol{x}$. In particular, define $\Psi_i(\boldsymbol{x}) = \boldsymbol{e}_{x_{i-2}} \otimes \cdots \otimes \boldsymbol{e}_{x_{i+2}}$, where $\otimes$ denotes the Kronecker product between column vectors, and $\boldsymbol{e}_{x_i}$ denotes a one-hot encoding of word x_i, concatenated with the one-hot encoding of its the part of speech tag and syntactic chunk tag which are provided with the input. Now, we can define the feature map Φ as

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \left[\sum_{v=1}^{p} \Psi_v(\boldsymbol{x}) \otimes \boldsymbol{e}_{y_v} \right] \oplus \left[\sum_{i=0}^{p} \boldsymbol{e}_{y_v} \otimes \boldsymbol{e}_{y_{v+1}} \right] ,$$

where $\boldsymbol{e}_y \in \mathbb{R}^{|\mathcal{Y}_{\text{tag}}|}$ is a one-hot encoding of $y \in \mathcal{Y}_{\text{tag}}$, and $\oplus$ denotes vector concatenation.

Inference. We use the Viterbi algorithm (Viterbi, 1967) as the max oracle and top-K Viterbi algorithm (Algorithm 2.1) for the top-K oracle.

Dataset. The dataset used was CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), which contains about $\sim 20K$ sentences.

Evaluation Metric. We follow the official CoNLL metric: the F_1 measure excluding the 'O' tags. In addition, we report the objective function value measured on the training set ("train loss").

Other Implementation Details. The sparse feature vectors obtained above are hashed onto $2^{16} - 1$ dimensions for efficiency.

PASCAL VOC 2007: Visual Object Localization. Given an image and an object of interest, the task is to localize the object in the given image, i.e., determine the best bounding box around the object. A related, but harder task is object detection, which requires identifying and localizing any number of objects of interest, if any, in the image. Here, we restrict ourselves to pure localization with a single instance of each object. Given an image $\boldsymbol{x} \in \mathcal{X}$ of size $n_1 \times n_2$, the label $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$ is a bounding box, where $\mathcal{Y}(\boldsymbol{x})$ is the set of all bounding boxes in an image of size $n_1 \times n_2$. Note that $|\mathcal{Y}(\boldsymbol{x})| = O(n_1^2 n_2^2)$.

Loss Function. The PASCAL IoU metric (Everingham et al., 2010) is used to measure the quality of localization. Given bounding boxes $\boldsymbol{y}, \boldsymbol{y}'$, the IoU is defined as the ratio of the intersection of the bounding boxes to the union:

$$\text{IoU}(\boldsymbol{y}, \boldsymbol{y}') = \frac{\text{Area}(\boldsymbol{y} \cap \boldsymbol{y}')}{\text{Area}(\boldsymbol{y} \cup \boldsymbol{y}')} .$$

We then use the $1 - \text{IoU}$ loss defined as $\ell(\boldsymbol{y}, \boldsymbol{y}') = 1 - \text{IoU}(\boldsymbol{y}, \boldsymbol{y}')$.

Score Function. The formulation we use is based on the popular R-CNN approach (Girshick et al., 2014). We consider two cases: linear score and non-linear score ϕ, both of which are based on the following definition of the feature map $\Phi(\boldsymbol{x}, \boldsymbol{y})$.

- Consider a patch $\boldsymbol{x}|_{\boldsymbol{y}}$ of image $\boldsymbol{x}$ cropped to box $\boldsymbol{y}$, and rescale it to 64×64. Call this $\Pi(\boldsymbol{x}|_{\boldsymbol{y}})$.
- Consider a convolutional neural network known as AlexNet (Krizhevsky et al., 2012) pre-trained on ImageNet (Russakovsky et al., 2015) and pass $\Pi(\boldsymbol{x}|_{\boldsymbol{y}})$ through it. Take the output of `conv4`, the penultimate convolutional layer as the feature map $\Phi(\boldsymbol{x}, \boldsymbol{y})$. It is of size $3 \times 3 \times 256$.

In the case of linear score functions, we take $\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$. In the case of non-linear score functions, we define the score ϕ as the result of a convolution composed with a non-linearity and followed by a linear map. Concretely, for $\boldsymbol{\theta} \in \mathbb{R}^{H \times W \times C_1}$ and $\boldsymbol{w} \in \mathbb{R}^{C_1 \times C_2}$ let the map $\boldsymbol{\theta} \mapsto \boldsymbol{\theta} \star \boldsymbol{w} \in \mathbb{R}^{H \times W \times C_2}$ denote a two-dimensional convolution with stride 1 and kernel size 1, and $\sigma : \mathbb{R} \to \mathbb{R}$ denote the exponential linear unit, defined respectively as

$$[\boldsymbol{\theta} \star \boldsymbol{w}]_{ij} = \boldsymbol{w}^\top [\boldsymbol{\theta}]_{ij} \quad \text{and} \quad \sigma(x) = x \, \mathbb{I}(x \geq 0) + (\exp(x) - 1) \, \mathbb{I}(x < 0) \,,$$

where $[\boldsymbol{\theta}]_{ij} \in \mathbb{R}^{C_1}$ is such that its lth entry is $\boldsymbol{\theta}_{ijl}$ and likewise for $[\boldsymbol{\theta} \star \boldsymbol{w}]_{ij}$. We overload notation to let $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ denote the exponential linear unit applied element-wise. Notice that σ is smooth. The non-linear score function ϕ is now defined, with $\boldsymbol{w}_1 \in \mathbb{R}^{256 \times 16}$, $\boldsymbol{w}_2 \in \mathbb{R}^{16 \times 3 \times 3}$ and $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2)$, as,

$$\phi(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \sigma(\Phi(\boldsymbol{x}, \boldsymbol{y}) \star \boldsymbol{w}_1), \boldsymbol{w}_2 \rangle \,.$$

Inference. For a given input image $\boldsymbol{x}$, we follow the R-CNN approach (Girshick et al., 2014) and use selective search (Van de Sande et al., 2011) to prune the search space. In particular, for an image $\boldsymbol{x}$, we use the selective search implementation provided by OpenCV (Bradski, 2000) and take the top 1000 candidates returned to be the set $\widehat{\mathcal{Y}}(\boldsymbol{x})$, which we use as a proxy for $\mathcal{Y}(\boldsymbol{x})$. The max oracle and the top-K oracle are then implemented as exhaustive searches over this reduced set $\widehat{\mathcal{Y}}(\boldsymbol{x})$.

Dataset. We use the PASCAL VOC 2007 dataset (Everingham et al., 2010), which contains $\sim 5K$ annotated consumer (real world) images shared on the photo-sharing site Flickr from 20 different object categories. For each class, we consider all images with only a single occurrence of the object, and train an independent model for each class.

Evaluation Metric. We keep track of two metrics. The first is the localization accuracy, also known as CorLoc (for correct localization), following Deselaers et al. (2010). A bounding box with IoU > 0.5 with the ground truth is considered correct and the localization accuracy is the fraction of images labeled correctly. The second metric is average precision (AP), which requires a confidence score for each prediction. We use $\phi(\boldsymbol{x}, \boldsymbol{y}'; \boldsymbol{w})$ as the confidence score of $\boldsymbol{y}'$. As previously, we also plot the objective function value measured on the training examples.

Other Implementation Details. For a given input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ in the dataset, we instead use $(\boldsymbol{x}, \widehat{\boldsymbol{y}})$ as a training example, where $\widehat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}' \in \widehat{\mathcal{Y}}(\boldsymbol{x})} \mathrm{IoU}(\boldsymbol{y}, \boldsymbol{y}')$ is the element of $\widehat{\mathcal{Y}}(\boldsymbol{x})$ which overlaps the most with the true output $\boldsymbol{y}$.

2.6.2 Methods Compared

The experiments compare various convex stochastic and incremental optimization methods for structured prediction.

- **SGD**: Stochastic subgradient method with a learning rate $\gamma_t = \gamma_0/(1 + \lfloor t/t_0 \rfloor)$, where η_0, t_0 are tuning parameters. Note that this scheme of learning rates does not have a theoretical analysis. However, the averaged iterate $\overline{\boldsymbol{w}}_t = 2/(t^2 + t) \sum_{\tau=1}^{t} \tau \boldsymbol{w}_\tau$ obtained from the related scheme $\gamma_t = 1/(\lambda t)$ was shown to have a convergence rate of $O((\lambda \varepsilon)^{-1})$ (Shalev-Shwartz et al., 2011; Lacoste-Julien et al., 2012). It works on the non-smooth formulation directly.

- **BCFW**: The block coordinate Frank-Wolfe algorithm of Lacoste-Julien et al. (2013). We use the version that was found to work best in practice, namely, one that uses the weighted averaged iterate $\overline{\boldsymbol{w}}_t = 2/(t^2+t)\sum_{\tau=1}^{t} \tau \boldsymbol{w}_\tau$ (called `bcfw-wavg` by the authors) with an optimal tuning of learning rates. This algorithm also works on the non-smooth formulation and does not require any tuning.

- **SVRG**: The SVRG algorithm proposed by Johnson and Zhang (2013), with each epoch making one pass through the dataset and using the averaged iterate to compute the full gradient and restart the next epoch. This algorithm requires smoothing.

- **Casimir-SVRG-const**: Algorithm 2.3 with SVRG as the inner optimization algorithm. The parameters μ_k and κ_k as chosen in Proposition 2.25, where μ and κ are hyperparameters. This algorithm requires smoothing.

- **Casimir-SVRG-adapt**: Algorithm 2.3 with SVRG as the inner optimization algorithm. The parameters μ_k and κ_k as chosen in Proposition 2.26, where μ and κ are hyperparameters. This algorithm requires smoothing.

2.6.3 Hyperparameters and Variants

Smoothing. In light of the discussion of Section 2.4, we use the ℓ_2^2 smoother $\omega(\boldsymbol{u}) = \|\boldsymbol{u}\|_2^2/2$ and use the top-K strategy for efficient computation. We then have $D_\omega = 1/2$.

Regularization. The regularization coefficient λ is chosen as c/n, where c is varied in $\{0.01, 0.1, 1, 10\}$.

Choice of K. The experiments use $K = 5$ for named entity recognition where the performance of the top-K oracle is K times slower, and $K = 10$ for visual object localization, where the running time of the top-K oracle is independent of K. We also present results for other values of K in Figure 2.4d and find that the performance of the tested algorithms is robust to the value of K.

Tuning Criteria. Some algorithms require tuning one or more hyperparameters such as the learning rate. We use grid search to find the best choice of the hyperparameters using the following criteria: For the named entity recognition experiments, the train function value and the validation F_1 metric were only weakly correlated. For instance, the 3 best learning rates in the grid in terms of F_1 score, the best F_1 score attained the worst train function value and vice versa. Therefore, we choose the value of the tuning parameter that attained the best objective function value within 1% of the best validation F_1 score in order to measure the optimization performance while still remaining relevant to the named entity recognition task. For the visual object localization task, a wide range of hyperparameter values achieved nearly equal performance in terms of the best CorLoc over the given time horizon, so we choose the value of the hyperparameter that achieves the best objective function value within a given iteration budget.

Learning Rate. The algorithms SVRG and Casimir-SVRG-adapt require tuning of a learning rate, while SGD requires η_0, t_0 and Casimir-SVRG-const requires tuning of the Lipschitz constant L of $\nabla F_{\mu\omega}$, which determines the learning rate $\gamma = 1/(L+\lambda+\kappa)$. Therefore, tuning the Lipschitz parameter is similar to tuning the learning rate. For both the learning rate and Lipschitz parameter, we use grid search on a logarithmic grid, with consecutive entries chosen a factor of two apart.

Choice of κ. For Casimir-SVRG-const, with the Lipschitz constant in hand, the parameter κ is chosen to minimize the overall complexity as in Proposition 2.25. For Casimir-SVRG-adapt, we use $\kappa = \lambda$.

Stopping Criteria. Following the discussion of Section 2.5, we use an iteration budget of $T_{\text{budget}} = n$.

Warm Start. The warm start criterion determines the starting iterate of an epoch of the inner optimization algorithm. Recall that we solve the following subproblem using SVRG for the kth iterate (cf. (2.30)):

$$\boldsymbol{w}_k \approx \operatorname*{arg\,min}_{\boldsymbol{w} \in \mathbb{R}^d} F_{\mu_k \omega, \kappa_k}(\boldsymbol{w}_k; \boldsymbol{z}_{k-1}) \,.$$

Here, we consider the following warm start strategy to choose the initial iterate $\widehat{\boldsymbol{w}}_0$ for this subproblem:

- `Prox-center`: $\widehat{\boldsymbol{w}}_0 = \boldsymbol{z}_{k-1}$.

In addition, we also try out the following warm start strategies of Lin et al. (2018):

- `Extrapolation`: $\widehat{\boldsymbol{w}}_0 = \boldsymbol{w}_{k-1} + c(\boldsymbol{z}_{k-1} - \boldsymbol{z}_{k-2})$ where $c = \frac{\kappa}{\kappa + \lambda}$.
- `Prev-iterate`: $\widehat{\boldsymbol{w}}_0 = \boldsymbol{w}_{k-1}$.

We use the `Prox-center` strategy unless mentioned otherwise.

Level of Smoothing and Decay Strategy. For SVRG and Casimir-SVRG-const with constant smoothing, we try various values of the smoothing parameter in a logarithmic grid. On the other hand, Casimir-SVRG-adapt is more robust to the choice of the smoothing parameter (Figure 2.4a). We use the defaults of $\mu = 2$ for named entity recognition and $\mu = 10$ for visual object localization.

2.6.4 Experimental study of different methods

For the named entity recognition task, Figure 2.2 plots the performance of various methods on CoNLL 2003. On the other hand, Figure 2.3 presents plots for various classes of PASCAL VOC 2007 for visual object localization.

The plots reveal that smoothing-based methods converge faster in terms of training error while achieving a competitive performance in terms of the performance metric on a held-out set. Furthermore, BCFW and SGD make twice as many actual passes as SVRG based algorithms.

2.6.5 Experimental Study of Effect of Hyperparameters

We now study the effects of various hyperparameter choices.

Effect of Smoothing. Figure 2.4a plots the effect of the level of smoothing for Casimir-SVRG-const and Casimir-SVRG-adapt. The plots reveal that, in general, small values of the smoothing parameter lead to better optimization performance for Casimir-SVRG-const. Casimir-SVRG-adapt is robust to the choice of μ. Figure 2.4b shows how the smooth optimization algorithms work when used heuristically on the non-smooth problem.

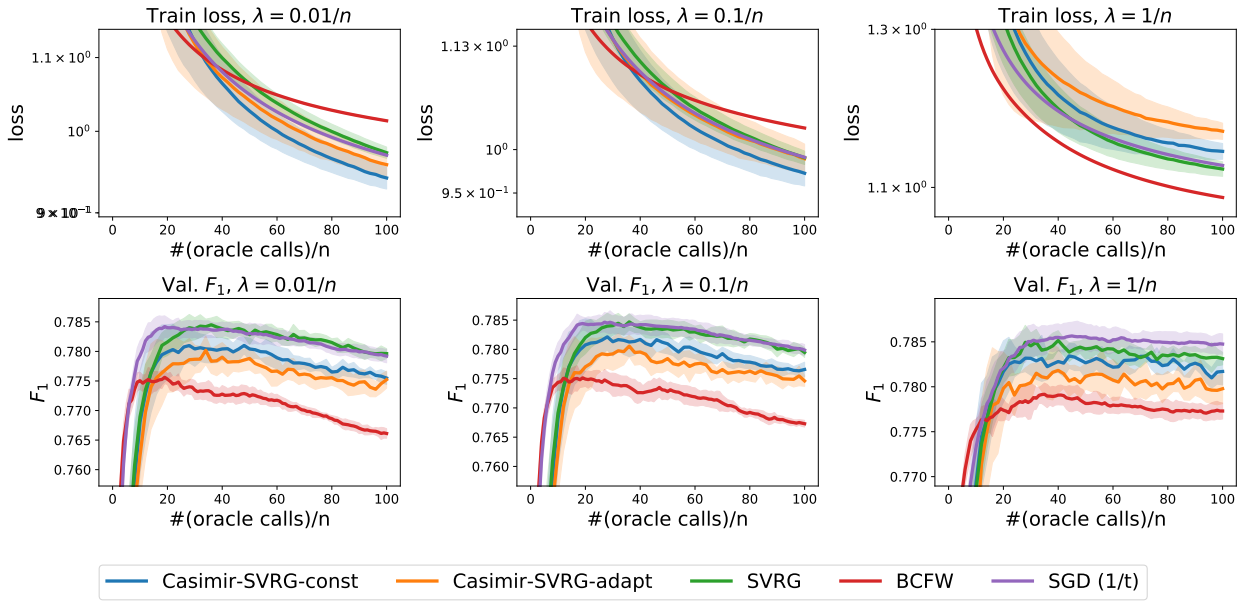


Figure 2.2: Comparison of convex optimization algorithms for the task of Named Entity Recognition on CoNLL 2003.

Effect of Warm Start Strategies. Figure 2.4c plots different warm start strategies for Casimir-SVRG-const and Casimir-SVRG-adapt. We find that Casimir-SVRG-adapt is robust to the choice of the warm start strategy while Casimir-SVRG-const is not. For the latter, we observe that `Extrapolation` is less stable (i.e., tends to diverge more) than `Prox-center`, which is in turn less stable than `Prev-iterate`, which always works (cf. Figure 2.4c). However, when they do work, `Extrapolation` and `Prox-center` provide greater acceleration than `Prev-iterate`. We use `Prox-center` as the default choice to trade-off between acceleration and applicability.

Effect of K. Figure 2.4d illustrates the robustness of the method to choice of K: we observe that the results are all within one standard deviation of each other.

2.7 Related Work

2.7.1 Surrogates for Structured Prediction

As discussed in Section 2.1, the underlying goal is to minimize the task loss, cf. (2.6). However, this objective is piece-wise constant in the parameters and is not amenable to gradient-based optimization. Therefore, past approaches either (a) construct a proxy for the gradient to this loss to be used in first-order optimization, or, (b) define a surrogate loss, and optimize it using first-order methods in lieu of optimizing the task loss. Instances of the former include the straight-through estimator (Bengio et al., 2013) and SPIGOT (Peng et al., 2018). However, these approaches come with no theoretical guarantees of correctness. Typical examples of the latter include convex surrogates such as the structured perceptron (LeCun et al., 1998; Collins, 2002), the structural hinge loss (Taskar et al., 2004b; Tsochantaridis et al., 2004) and the conditional random field (LeCun et al., 1998; Lafferty et al., 2001), and nonconvex surrogates such as

the structured ramp (Chapelle et al., 2009) and probit (Keshet et al., 2011) losses. More recently, numerous surrogates such as the square loss (Ciliberto et al., 2016; Osokin et al., 2017) and variants including projections (Blondel, 2019) have been the focus of theoretical study for properties such as consistency.

Some approaches which construct a gradient proxy can be viewed as minimizing an appropriate surrogate. For instance, the approach proposed in (Pogancic et al., 2020) coincides with minimization of the structural ramp loss, while (Berthet et al., 2020) minimizes a Gaussian smoothing of the structured perceptron loss. An approach which, at first sight, is disparate from the two mentioned above is direct loss minimization (Hazan et al., 2010; Song et al., 2016). However, it turns out in practice to minimize a surrogate known as the structural ramp loss (cf. McAllester and Keshet, 2011). Direct loss minimization gives an expression for the gradient of the expected task loss as

$$\nabla_{\boldsymbol{w}} \, \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathbb{P}} \left[\ell \left(\boldsymbol{y}, \arg\max_{\boldsymbol{y}'\in\mathcal{Y}} \phi(\boldsymbol{x}, \boldsymbol{y}'; \boldsymbol{w}) \right) \right] = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \mathbb{E} \left[\nabla \phi(\boldsymbol{x}, \widehat{\boldsymbol{y}}_\varepsilon(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}); \boldsymbol{w}) - \nabla \phi(\boldsymbol{x}, \boldsymbol{y}^\star(\boldsymbol{x}; \boldsymbol{w}); \boldsymbol{w}) \right] ,$$

for a continuous distribution $\mathbb{P}$, where $\widehat{\boldsymbol{y}}_\varepsilon(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) \in \arg\max_{\boldsymbol{y}'\in\mathcal{Y}} \{\phi(\boldsymbol{x}, \boldsymbol{y}'; \boldsymbol{w}) + \varepsilon \, \ell(\boldsymbol{y}, \boldsymbol{y}')\}$ is the output of loss augmented inference. However, it is not possible to obtain unbiased stochastic gradients for this expression since, owing to the piecewise constant nature of the argmax, the limit $\varepsilon \to 0$ and expectation (i.e., integral) do not commute. Using direct loss minimization with a fixed $\varepsilon > 0$ gives a biased gradient estimator, which turns out to coincide the the Clarke subdifferential of the (appropriately scaled) structured ramp loss.

While we focus in this work on the structural hinge loss, the approach proposed here can be viewed more generally to construct faster optimization algorithms for nonsmooth Fenchel-Young losses (Blondel et al., 2020) through the use of appropriate smooth surrogates.

2.7.2 Inference in Structured Prediction

Note that the objective in (2.7),(2.8) contains a maximization over an exponentially large (in the size of each $\boldsymbol{y} \in \mathcal{Y}$) number $m = |\mathcal{Y}|$ of outputs, and this is achieved in general by inference algorithms. There exist a number of exact inference algorithms based on the output structure. These use dynamic programming (Pearl, 1988; Dawid, 1992), graph cuts (Greig et al., 1989; Ishikawa and Geiger, 1998), bipartite matchings (Cheng et al., 1996; Taskar et al., 2005) and search algorithms (Daumé III and Marcu, 2005; Lampert et al., 2008; Lewis and Steedman, 2014; He et al., 2017), among others. We note that when exact inference is not tractable, continuous relaxations (Schlesinger, 1976; Johnson, 2008), iterative fixed-point algorithms (McEliece et al., 1998; Murphy et al., 1999) and variational inference (Wainwright et al., 2005; Wainwright and Jordan, 2008).

Top-K Inference. The top-K oracle considered here for ℓ_2^2 smoothing of inference echoes older heuristics in speech and language processing (Jurafsky et al., 2014). Combinatorial algorithms for top-K inference have been studied extensively under the name "M-best MAP", including exact algorithms for trees (Seroussi and Golmard, 1994; Nilsson, 1998), search algorithms (Flerova et al., 2016), or other cases where a max-marginal oracle is available (Yanover and Weiss, 2004). This study has been extended to continuous relaxation as well (Fromer and Globerson, 2009; Batra, 2012). In this work, we show how top-K inference algorithms can be used to approximate ℓ_2^2-smoothed inference.

Smoothing Inference. Smoothing for inference was used to speed up iterative algorithms for continuous relaxations, including the entropy smoother (Johnson, 2008; Jojic et al., 2010; Savchynskyy et al., 2011) and

ℓ_2^2 smoother (Meshi et al., 2012). Explicit smoothing of discrete inference algorithms in order to smooth the learning problem was considered for the entropy (Zhang et al., 2014) and ℓ_2^2 smoothers (Song et al., 2014) as well. Smooth inference is crucially important for backpropagation; we shall return to it shortly.

Alternatives to Inference Oracles. The blackbox view of inference oracles considered here are tailored to the case when exact inference is tractable through combinatorial algorithms, for which there exist highly optimized algorithms. An alternative is the approach of projection oracles, which admit efficient implementations for a variety of structures (Taskar et al., 2006; Blondel, 2019). When such combinatorial algorithms are not available, one must resort to continuous relaxations of inference; the use of inference as a blackbox here is suboptimal, and interleaving optimization and inference steps have been shown to lead to speedups (Hazan et al., 2016).

2.7.3 End-to-End Structured Prediction and Backpropagation Through Inference

The general framework for global training of structured prediction models was introduced by (Bottou and Gallinari, 1990) and applied to handwriting recognition (Bengio et al., 1995) and document processing (LeCun et al., 1998). This approach, now called "deep structured prediction", finds widespread application (Collobert et al., 2011; Belanger and McCallum, 2016).

As we have seen, the training objectives of structured prediction involve combinatorial inference algorithms. End-to-end learning of structured prediction via differentiable programming must therefore "backpropagate through" an inference algorithm. Backpropagation through a combinatorial inference algorithm could either refer to (a) backpropagation through the value of inference algorithm, e.g., Eq. (2.7), or (b) backpropagation through the maximizer obtained from the inference algorithm, e.g., Eq. (2.6). We look at both in turn, and then highlight the connection between the two.

The value of an inference algorithm is the general form $f(\boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$. When $\boldsymbol{w} \mapsto \psi(\cdot; \boldsymbol{w})$ is smooth, the inclusion $\nabla \psi(\boldsymbol{y}^\star(\boldsymbol{w}); \boldsymbol{w}) \in \partial f(\boldsymbol{w})$ follows from the chain rule (Rockafellar and Wets, 2009, Thm. 10.6), where $\boldsymbol{y}^\star(\boldsymbol{w}) \in \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$ is a maximizer and $\partial f(\boldsymbol{w})$ is the regular subdifferential. However, ψ is not a smooth function of $\boldsymbol{w}$, it is possible to construct the Clarke subdifferential of f w.r.t. $\boldsymbol{w}$ (Gao et al., 2019) using the optimal solution $\boldsymbol{y}^\star(\boldsymbol{w})$ again. In this work, we use infimal convolution smoothing to construct a smooth surrogate $f_{\mu\omega}$ to f, i.e., $f_{\mu\omega}$ is continuously differentiable whose gradient is $O(1/\mu)$-Lipschitz and $|f - f_{\mu\omega}| \leq O(\mu)$. The benefit of this approach is that it allows us to consider accelerated optimization algorithms, as we discuss later.

"Backpropagating through the maximizer of inference" refers to computing first-order information of functions of the form $\boldsymbol{w} \mapsto G(\boldsymbol{y}^\star(\boldsymbol{w}))$ for some $G : \mathcal{Y} \to \mathbb{R}$, where $\boldsymbol{y}^\star(\boldsymbol{w}) \in \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$ is a maximizer. As we have seen earlier with (2.6), these functions are piecewise constant in $\boldsymbol{w}$. Research, therefore, has focuses on defining surrogates which admit subgradients. This typically has two ingredients: (a) continuous representation, and, (b) smoothing. For (a), consider a q-dimensional representation $\varphi : \mathcal{Y} \to \mathbb{R}^q$ of outputs, and relaxations $\bar{\psi} : \operatorname{conv} \varphi(\mathcal{Y}) \times \mathbb{R}^d \to \mathbb{R}$, $\bar{G} : \operatorname{conv} \varphi(\mathcal{Y}) \to \mathbb{R}$, which satisfy $\bar{\psi}(\varphi(\boldsymbol{y}); \boldsymbol{w}) = \psi(\boldsymbol{y}; \boldsymbol{w})$ and $\bar{G}(\varphi(\boldsymbol{y})) = G(\boldsymbol{y})$. The other ingredient (b) is smoothing. It is required since the relaxed maximizer $\boldsymbol{z}^\star(\boldsymbol{w}) \in \arg\max_{\boldsymbol{z} \in \operatorname{conv} \varphi(\mathcal{Y})} \bar{\psi}(\boldsymbol{z}; \boldsymbol{w})$ might not be unique, leading to a discontinuity in $\boldsymbol{w} \mapsto \boldsymbol{z}^\star(\boldsymbol{w})$. The smoothing ensures that this map is Lipschitz continuous, which allows appropriate subdifferentials to be well-defined. Past works have considered smoothing with infimal convolution (Martins and Astudillo, 2016; Djolonga and Krause, 2017; Niculae et al., 2018; Wilder et al., 2019), as well as convolution with noise (Papandreou and Yuille, 2011; Berthet et al., 2020). Notably, the former includes the ubiquitous use of the softmax map as a Lipschitz (in fact, continuously differentiable) proxy to the argmax. Smoothing with infimal convolution is preferred to convolution with noise because the

Lipschitz constant of $\boldsymbol{w} \mapsto \boldsymbol{z}^\star(\boldsymbol{w})$ in the latter case scales linearly with the dimension of the ambient space (Nesterov, 2011), while it is dimension-free for the former (Beck and Teboulle, 2012).

To make concrete the relation between backpropagating through the value of inference versus its maximizer, consider the one-hot representation[3] $\varphi(\boldsymbol{y}) \in \{0,1\}^{|\mathcal{Y}|}$, and infimal convolution smoothing with a strongly convex function ω and a smoothing parameter $\mu > 0$. In this case, we have the simple relation: $\boldsymbol{z}^\star_{\mu\omega}(\boldsymbol{w}) = \nabla f_{\mu\omega}(\boldsymbol{w})$ (the notation $\boldsymbol{z}^\star_{\mu\omega}$ highlights the dependence of the maximizer on ω and μ). This tight connection allows advances in backpropagation through inference maximizers to yield first order oracles of smooth surrogates to the value of inference. In this work, we make this connection concrete through inference oracles, and utilize these for accelerated optimization of structural support vector machines.

We provide detailed comparison of smoothness of our approach to the other smooth inference approaches of differentiable dynamic programming (Mensch and Blondel, 2018) and SparseMAP (Niculae et al., 2018) in Section 2.3. In addition, the viewpoint of first-order optimization of structured prediction is absent in these related works. We define smooth inference oracles in the context of black-box first-order optimization and establish worst-case complexity bounds for incremental optimization algorithms making calls to these oracles. Indeed, we relate the amount of smoothing to the resulting complexity of the optimization algorithms relying on smooth inference oracles. This gives a concrete use-case to smooth inference in training structured prediction models.

2.7.4 Optimization Algorithms for Structural Support Vector Machines

We now turn to the structural hinge loss as surrogate loss and optimization algorithms to minimize it. The optimization problem (2.8) is convex iff the score function $\boldsymbol{w} \mapsto \phi(\cdot, \cdot; \boldsymbol{w})$ is affine. We consider each case in turn.

Convex Optimization. Suppose the optimization problem (2.8) is convex. Its Fenchel-Rockafellar dual is the quadratic program

$$\max_{\boldsymbol{u}^{(1)},\cdots,\boldsymbol{u}^{(n)} \in \Delta^{m-1}} \left\{ -\sum_{i=1}^{n} \langle \boldsymbol{b}^{(i)}, \boldsymbol{u}^{(i)} \rangle - \frac{1}{2n\lambda} \left\| \sum_{i=1}^{n} (\boldsymbol{A}^{(i)})^\top \boldsymbol{u}^{(i)} \right\|_2^2 \right\},$$

where $m = |\mathcal{Y}|$ is the size of the output space, $\boldsymbol{b}^{(i)} = \big(\ell(\boldsymbol{y}^{(i)}, \boldsymbol{y})\big)_{\boldsymbol{y} \in \mathcal{Y}} \in \mathbb{R}^m$, $\boldsymbol{A}^{(i)} \in \mathbb{R}^{m \times d}$ is the matrix given by $\boldsymbol{A}^{(i)}\boldsymbol{w} + \boldsymbol{b}^{(i)} = \big(\psi^{(i)}(\boldsymbol{y}; \boldsymbol{w})\big)_{\boldsymbol{y} \in \mathcal{Y}}$. Note that (a) the dimensionality $m = |\mathcal{Y}|$ of the dual variables is extremely large, and, (b) the primal can be written as a quadratic program with $m = |\mathcal{Y}|$ constraints.

Early works (Taskar et al., 2004b; Tsochantaridis et al., 2004; Joachims et al., 2009; Teo et al., 2009) considered batch quadratic programming algorithms. The dual algorithms used reparameterizations to reduce the dimensionality of the dual variables. On the other hand, primal algorithms used constraint generation methods to work with a tractable number of constraints, where choosing the "best" constraint to add could be solved by inference. These algorithms can be viewed as iterative application of dual variable selection using inference and optimization over selected variables.

The move from batch to incremental algorithms started with an application of the stochastic subgradient method, which operated directly on the non-smooth primal formulation (Ratliff et al., 2007; Shalev-Shwartz et al., 2011). Here, the subgradient was again given by inference (cf. max orcale in Section 2.3). The block coordinate Frank-Wolfe (BCFW) (Lacoste-Julien et al., 2013) was proposed based on the observations (a) that a linear minimization oracle of the dual could be computed by inference, and, (b) the dual variables

[3] Formally, consider a bijection $\sigma : \mathcal{Y} \to m = |\mathcal{Y}|$, and set $[\varphi(\boldsymbol{y})]_{\sigma(\boldsymbol{y})} = 1$ and zero elsewhere.

variables need not be explicity constructed. Extensions based on the other variants of the Frank-Wolfe algorithms were proposed as well (Osokin et al., 2016).

Primal-dual or saddle-point approaches include the mirror-prox algorithm (Taskar et al., 2006; Cox et al., 2014; He and Harchaoui, 2015). In the structured prediction setting, one must be careful about the dimensionality m of the dual variables, for instance, by using a compact q-dimensional representation $\varphi : \mathcal{Y} \to \mathbb{R}^q$ for some $q \ll |\mathcal{Y}|$ and using *projection oracles* to efficiently compute projections on $\operatorname{conv} \varphi(\mathcal{Y}) \subseteq \mathbb{R}^q$ (Taskar et al., 2006). Incremental optimization algorithm for saddle-point problems (Palaniappan and Bach, 2016) could potentially be coupled with the tricks noted above to optimize structured prediction models. Incremental optimization algorithms suited to conditional random fields, a related convex surrogate, have also been proposed (Schmidt et al., 2015). An alternate approach is to consider the lower dimensional min-max saddle point problem mentioned above and consider the dual of the inner maximization (Hazan et al., 2016). Strong duality holds, and this results in a min-min problem, which can be tackled by block coordinate descent algorithms. The rate of this algorithm cannot directly be compared to the other rates mentioned here because the cost per iteration is different.

Table 2.1 gives an overview of the rates of convergence of the algorithms referred to above in comparison with the approaches proposed in this work. We note that the dual-based approaches and saddle point algorithms work only for the convex case where the score function $\phi(\cdot, \cdot; \boldsymbol{w})$ is affine in $\boldsymbol{w}$. We focus here on primal optimization algorithms in order to be able to train structured prediction models with affine or nonlinear mappings with a unified approach, and on incremental optimization algorithms which can scale to large datasets.

Nonconvex Optimization. In the general case with a nonlinear, yet smooth feature map ϕ, the problem (2.8) is nonsmooth and nonconvex. Optimization algorithms tailored to structured prediction in this case have not been proposed. Early works (LeCun et al., 1998, e.g.,) simply used the stochastic subgradient method, using a heuristic notion of subgradients at points where the function was not differentiable. Its rate of convergence on nonsmooth problems has recently been established as $O(1/\varepsilon^4)$ for an ε-stationary point (Davis and Drusvyatskiy, 2019). Stationarity is measured here in terms of the gradient norm of the Moreau envelope; see Chapter 3. The classical (inexact) prox-linear algorithm of (Burke, 1985; Wright, 1990), which generalizes the Levenberg-Marquardt algorithm for non-linear least squares, can be used to optimize (2.8) in the nonconvex case. The rate of convergence of this algorithm can be improved to $O(n/\varepsilon^2 + \sqrt{n}/\varepsilon^3)$ with the use of smoothing and acceleration. The notion of inference oracles introduced in this work allows this rate to be transferred to the case of structured prediction.

In the nonconvex case, the stochastic subgradient method can be implemented by using any reverse-mode automatic differentiation software. On the other hand, the prox-linear algorithms and its variants require computation of Jacobian-vector products; this is most efficient with forward-mode automatic differentiation (e.g. Griewank and Walther, 2008). In the upcoming Chapter 3, we will consider algorithms which exhibit worst-case rates of convergence on nonsmooth problems, yet produce acceleration and faster convergence when the problem is convex.

2.8 Discussion and Conclusion

We introduced a general notion of smooth inference oracles in the context of black-box first-order optimization. This allows us to extend the scope of fast incremental optimization algorithms to structured prediction problems owing to a careful blend of a smoothing strategy and an acceleration scheme. We illustrated the potential of our framework by proposing a new incremental optimization algorithm to train

max-margin structured prediction models both enjoying worst-case complexity bounds and demonstrating competitive performance on two real-world problems.

Concurrent with when the work in this chapter was published (Pillutla et al., 2018), there were a number of efforts to incorporate combinatorial algorithms in differentiable programming pipelines, as we discussed in Section 2.7.3. For a combinatorial algorithm appearing at the end of a pipeline, the various proposed techniques, including ours, amount to choosing a surrogate loss function such as a smoothed structural perceptron or hinge losses (cf. Section 2.7.1). On the other hand, the case where the combinatorial algorithm is away from the end is not as well understood. Consider, for instance, the problem

$$\inf_{\boldsymbol{w}_1, \boldsymbol{w}_2} \left[F(\boldsymbol{w}_1, \boldsymbol{w}_2) = f\left(\arg\max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{z}, \boldsymbol{g}(\boldsymbol{w}_1) \rangle; \boldsymbol{w}_2 \right) \right],$$

where a polytope $\mathcal{Z} \subset \mathbb{R}^m$, and maps $\boldsymbol{g} : \mathbb{R}^{d_1} \to \mathbb{R}^m$ and $f : \mathcal{Z} \times \mathbb{R}^{d_2} \to \mathbb{R}$ are given. The problem is fundamentally ill-posed, as $F(\boldsymbol{w}_1, \boldsymbol{w}_2)$ is a piecewise discontinuous function. Following the ideas presented in this chapter, one might consider a smoothing

$$F_{\mu\omega}(\boldsymbol{w}_1, \boldsymbol{w}_2) = f\left(\nabla h_{\mu \ell_2^2}\big(\boldsymbol{g}(\boldsymbol{w}_1)\big); \boldsymbol{w}_2 \right),$$

$h(\boldsymbol{u}) = \max_{\boldsymbol{z} \in \mathcal{Z}} \langle \boldsymbol{u}, \boldsymbol{z} \rangle$ is the support function of the polytope $\mathcal{Z}$ and $h_{\mu \ell_2^2}$ is its smoothing with coefficient $\mu > 0$. Using a small smoothing coefficient μ would overcome discontinuities but make it hard to "jump" from one erstwhile discontinuous region to another. On the other hand, using a large smoothing coefficient would completely transform the landscape of the objective, so the function we actually optimize might not be closely related to the original objective. A promising avenue to solve this problem is to study SPIGOT (Peng et al., 2018) or its variants (Mihaylova et al., 2020), and understand what objective they actually minimize. This might also help construct consistent alternatives which come with a guarantee that minimizing this objective leads to minimizing the original objective in the limit of infinite data.

The difficulty in optimizing pipelines with combinatorial algorithms has led to a move away from structured inference algorithms during learning to local prediction methods over the components of the structure. For instance, consider sequence to sequence models (Sutskever et al., 2014) that aim to predict a sequence $\boldsymbol{y} = (y_1, \cdots, y_p)$ from an in input sequence $\boldsymbol{x}$. The output sequence $\boldsymbol{y}$ is modeled in an autoregressive manner as $P(y_j | y_{1:j-1}, \boldsymbol{x}; \boldsymbol{w})$ using a normalized probabilistic model P with parameters $\boldsymbol{w}$. The learning problem over a component y_j is reduced to multiclass classification, while predictions are made sequentially using a decoding algorithm such as beam search. Such sequential structured prediction methods built atop neural networks such as transformers dominate modern natural language processing. We will return to them in Chapter 4 in the context of text generation.

On the optimization side, we showed that accelerated algorithms based on Casimir are faster in both theory and practice in the convex case. We will consider the nonconvex case in Chapter 3. We find here that algorithms based on the prox-linear method do not outperform the stochastic subgradient methods. Future work must reconcile a number of differences between theoretical results and practical observations. Firstly, optimization algorithms such as Adam (Kingma and Ba, 2015) are wildly successful in practice but fail to converge in simple examples (Reddi et al., 2018). Secondly, the convergence analysis of stochastic gradient algorithms in the nonconvex case does not reflect issues encountered in practice.

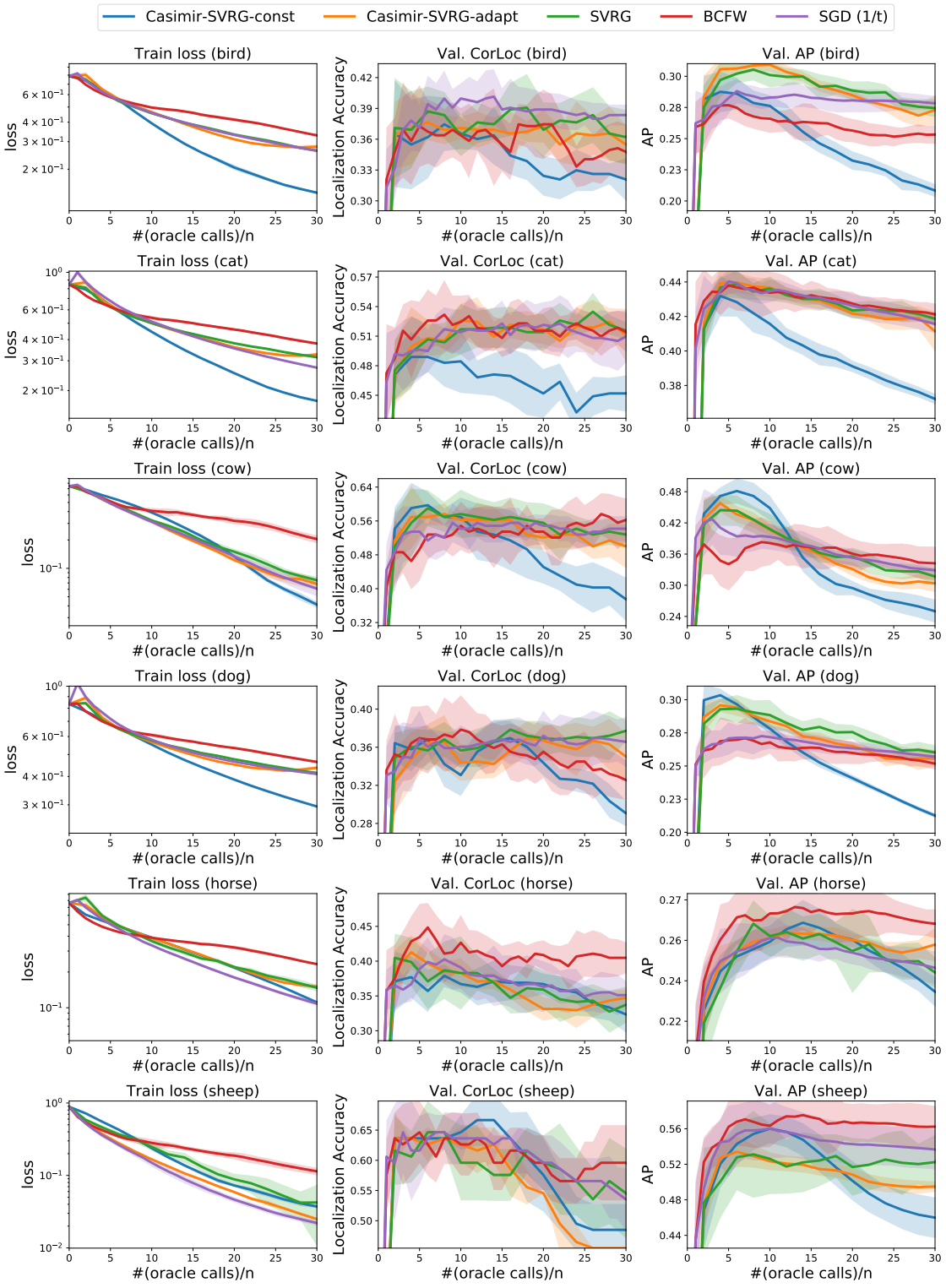
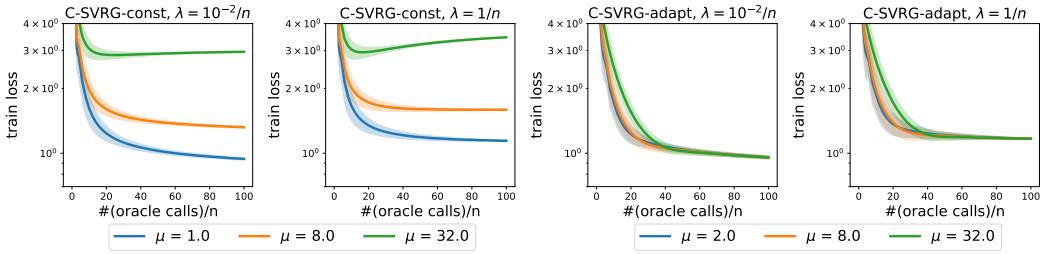
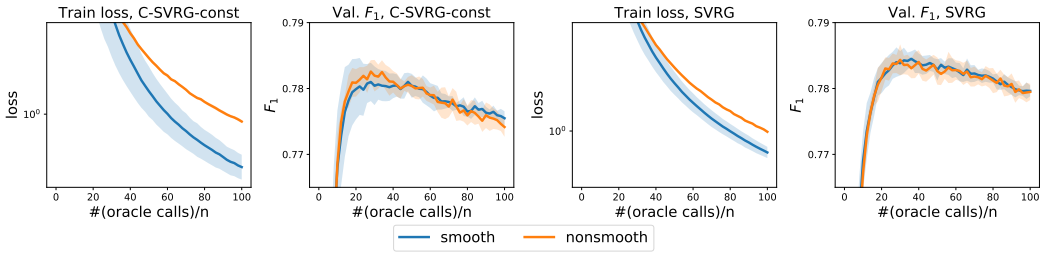


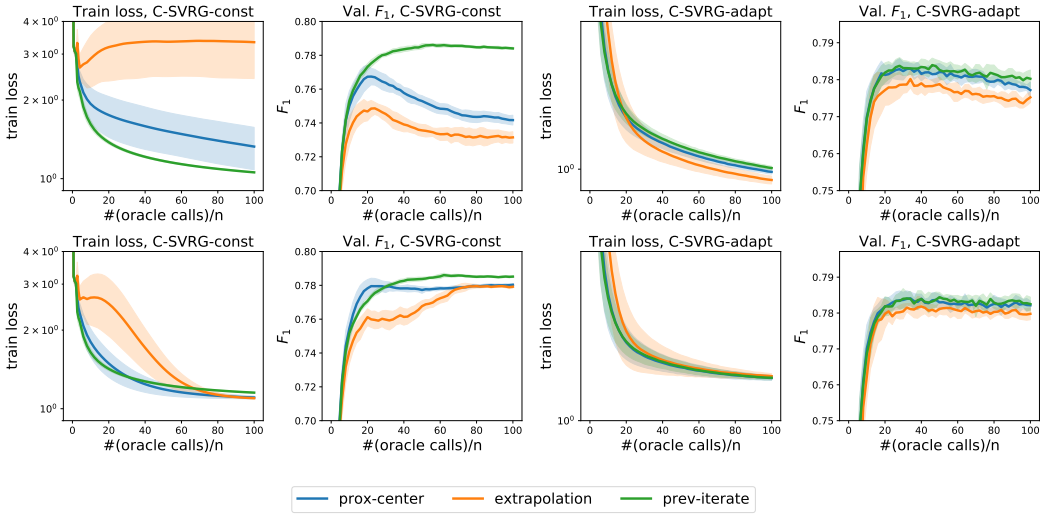
Figure 2.3: Comparison of convex optimization algorithms for the task of visual object localization on PASCAL VOC 2007 for $\lambda = 10/n$.



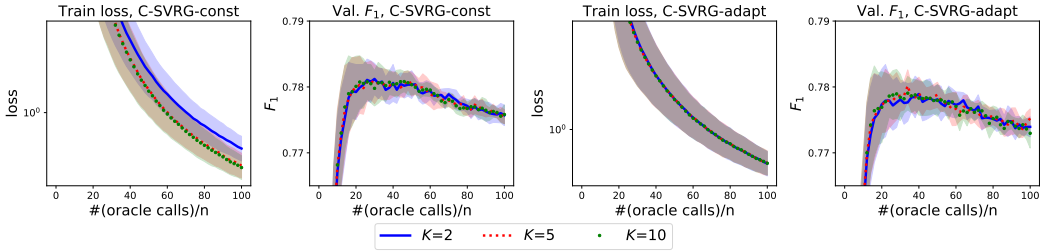
(a) Effect of level of smoothing.



(b) Effect of smoothing: use of smooth optimization with smoothing (labeled "smooth") versus the heuristic use of these algorithms without smoothing (labeled "non-smooth") for $\lambda = 0.01/n$.



(c) Effect of warm start strategies for $\lambda = 0.01/n$ (first row) and $\lambda = 1/n$ (second row).



(d) Effect of K in the top-K oracle ($\lambda = 0.01/n$).

Figure 2.4: Effect of hyperparameters for the task of Named Entity Recognition on CoNLL 2003. C-SVRG stands for Casimir-SVRG in these plots.

Chapter 3

The Trade-offs of Incremental Linearization Algorithms for Nonsmooth Composite Problems

In Chapter 2, we saw how to combine combinatorial algorithms with differentiable programming. We utilized these inference oracles to design smooth and accelerated first-order optimization algorithms for structured prediction where the objective is convex. In this chapter, we turn to the setting where the objective is nonconvex. We consider a more general compositional optimization problem that is nonsmooth and nonconvex.

The Gauss-Newton method was first developed to take advantage of the compositional structure of nonlinear least-squares problems: by linearizing the nonlinear mappings around the current iterate, we get an approximation of the objective that can be solved in analytic form to output the next iterate (Nocedal and Wright, 2006; Björck, 1996). The Gauss-Newton method was generalized to tackle generic compositional problems of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} h(\boldsymbol{g}(\boldsymbol{w})), \tag{3.1}$$

by linearizing the inner function $\boldsymbol{g}$ around the current iterate and solving the associated subproblem (Burke, 1985). For example, to solve nonlinear equations, we can consider minimizing the absolute deviations $\|\boldsymbol{g}(\boldsymbol{w})\|$ by iteratively linearizing the inner function around the current iterate and solving the resulting non-smooth convex subproblem (Nesterov, 2007).

The Gauss-Newton method and its variants such as the Levenberg-Marquardt method (Levenberg, 1944) have been applied successfully in numerous domains such as phase retrieval (Cichocki and Amari, 2002; Herring et al., 2019; Repetti et al., 2014), nonlinear control (Sideris and Bobrow, 2005; Roulet et al., 2019), non-negative matrix factorization (Huang and Fu, 2019), see (Björck, 1996) for a broader overview. The convergence guarantees for modified Gauss-Newton methods adapted to (3.1) have been thoroughly developed through the viewpoint of proximal regularization (Drusvyatskiy and Paquette, 2019; Bauschke et al., 2011). Modern machine learning problems such as deep learning possess a similar compositional structure, which makes Gauss-Newton-like algorithms potential good candidates (Drusvyatskiy and Paquette, 2019; Tran-Dinh et al., 2020; Zhang and Xiao, 2020).

However, in such problems, we are often interested in the generalization performance evaluated on unseen data. In this chapter, we investigate whether modified Gauss-Newton methods or prox-linear algorithms with incremental first-order methods in the inner loops are superior to direct stochastic subgra-

dient algorithms for nonsmooth problems with a compositional objective and a finite-sum structure. We present synthetic experiments that delineate the regimes where modified Gauss-Newton methods algorithms appear to outperform direct stochastic subgradient algorithms. We also compare these algorithms on a structured prediction problem with a convolutional neural network (end-to-end path planning) and nonlinear system solving with a non-differentiable criterion. Experimental results suggest that modified Gauss-Newton methods or prox-linear algorithms offer marginal gains in some settings, and confirm the versatility of direct stochastic subgradient algorithms to tackle complex learning problems.

3.1 Background

In this chapter, we consider optimization problems of the form

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad \left\{ F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} h(\boldsymbol{g}_i(\boldsymbol{w})) \right\} \tag{3.2}$$

with $\boldsymbol{g}_i : \mathbb{R}^d \to \mathbb{R}^m$ smooth, i.e., with Lipschitz continuous gradients, and $h : \mathbb{R}^m \to \mathbb{R}$ convex and Lipschitz-continuous. The above problem is an extension to (3.1) when considering multiple compositions that typically arise in empirical risk minimization. We are particularly interested in the settings where n is large, as is the case with modern data science applications of supervised learning.

We are given n input-output pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. In the examples we will consider, the inner function $\boldsymbol{g}_i(\boldsymbol{w}) = (\psi_i(\boldsymbol{y}; \boldsymbol{w}))_{\boldsymbol{y} \in \mathcal{Y}}$ denotes the scores of each prediction $\boldsymbol{y} \in \mathcal{Y}$ relative to the input-output pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, using a parametric model $\boldsymbol{g}(\cdot; \boldsymbol{w})$ such as a deep network. The outer function h reduces the scores for all predictions into a single scalar.

3.1.1 Examples

We instantiate the setup with specific examples.

Multi-output Regression. Here, each input $\boldsymbol{x}_i \in \mathbb{R}^p$ and each output $\boldsymbol{y}_i \in \mathbb{R}^m$ is a vector. Given a parameterized prediction function $\boldsymbol{\phi}(\cdot; \boldsymbol{w}) : \mathbb{R}^p \to \mathbb{R}^m$, the inner function $\boldsymbol{g}_i(\boldsymbol{w}) = \boldsymbol{\phi}(\boldsymbol{x}_i; \boldsymbol{w}) - \boldsymbol{y}_i$ is the residual, and h is simply a loss function on the residual. The loss function $h(\boldsymbol{u}) = \|\boldsymbol{u}\|_2^2$ would lead to the classical nonlinear least squares problems. In this chapter, we are interested in non-smooth h, which arises in robust regression problems with the mean absolute deviation loss $h(\boldsymbol{u}) = \|\boldsymbol{u}\|_1$ or the ℓ_2 loss $h(\boldsymbol{u}) = \|\boldsymbol{u}\|_2$ (without the square).

Multi-class Classification. Each input $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector while the output $\boldsymbol{y}_i \in \{1, \cdots, m\}$ is categorical. Typical loss functions include the multinomial logistic loss in the smooth case and multi-class hinge loss (Crammer and Singer, 2001) in the non-smooth case.

Structured Prediction. The multi-class classification setting can be generalized to structured prediction, i.e., the prediction of a combinatorial object such as a sequence with discriminative models, as we saw in Chapter 2. Here, $\boldsymbol{g}_i(\boldsymbol{w})$ is a stacking of the augmented score function ψ_i for all outputs, i.e., $\boldsymbol{g}_i(\boldsymbol{w}) = \big(\psi_i(\boldsymbol{y}; \boldsymbol{w})\big)_{\boldsymbol{y} \in \mathcal{Y}}$. The input $\boldsymbol{x}_i$ can be arbitrary, but the output comes from a structured space $\mathcal{Y}$. Examples of the loss function h include the conditional random fields (Lafferty et al., 2001) and the structural hinge loss (Taskar et al., 2004b; Tsochantaridis et al., 2004). In practice, efficient computation of this loss is possible using dynamic programming or other combinatorial algorithms (Viterbi, 1967). This approach has been applied for tasks in natural language processing (Rush, 2020), speech processing (Gales et al., 2012) and planning (Ratliff et al., 2006).

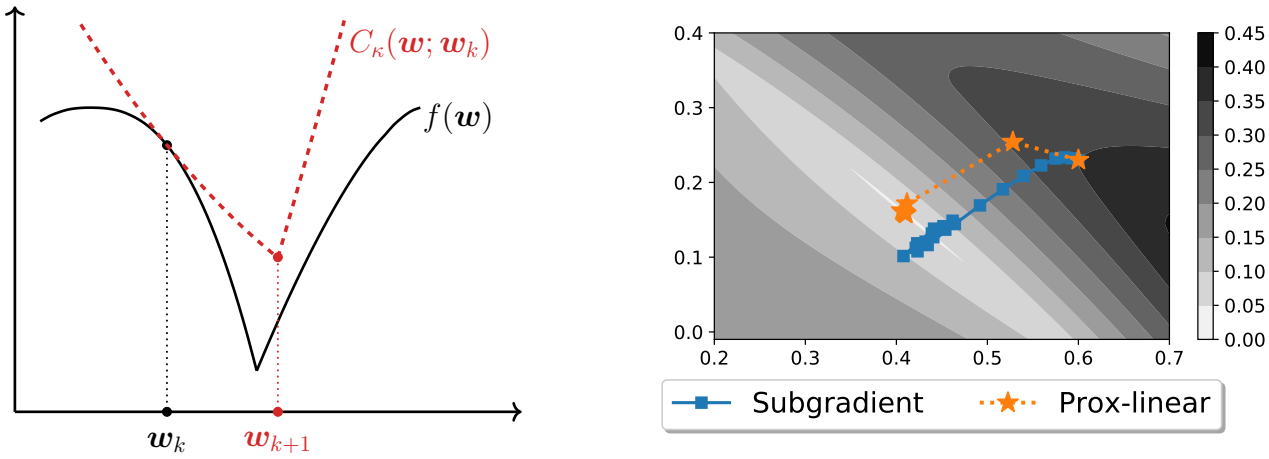


Figure 3.1: Left: The exact (resp. approximate) prox-linear method builds a convex model $C_\kappa(\boldsymbol{w}; \boldsymbol{w}_k)$ of the objective $F(\boldsymbol{w})$ around the current iterate $\boldsymbol{w}_k$. The next iterate $\boldsymbol{w}_{k+1}$ is then an exact (resp. approximate) minimizer of this model plus a proximal term. **Right**: A numerical comparison of gradient descent and the exact prox-linear method on a nonsmooth and nonconvex function $F : \mathbb{R}^2 \to \mathbb{R}$. The prox-linear method builds a more accurate model of the objective function, especially around points of nonsmoothness.

3.1.2 Optimization Algorithms

We consider two families of optimization algorithms, which are stochastic and non-smooth versions of gradient descent and the Gauss-Newton algorithm.

Stochastic Subgradient Method. The stochastic subgradient method is the non-smooth and stochastic analogue of gradient descent. With a slight abuse of nomenclature, we refer to it as *SGD*, which stands for stochastic gradient descent. In each iteration k, SGD samples an element i_k from the available n uniformly at random and takes a step in the direction of its subgradient $\boldsymbol{v}_k \in \partial(h \circ \boldsymbol{g}_{i_k})(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \gamma \boldsymbol{v}_k \,, \tag{3.3}$$

where γ is the learning rate and $\partial(h \circ \boldsymbol{g}_i)(\boldsymbol{w})$ denotes the regular (or Fréchet) subdifferential, which is defined as

$$\partial\varphi(\boldsymbol{w}) = \left\{\boldsymbol{v} \in \mathbb{R}^d \,:\, \varphi(\boldsymbol{w}') \geq \varphi(\boldsymbol{w}) + \boldsymbol{v}^\top(\boldsymbol{w}' - \boldsymbol{w}) + o(\|\boldsymbol{w}' - \boldsymbol{w}\|)\right\}.$$

The regular subdifferential generalizes the gradient for smooth functions as well as the convex subdifferential as it corresponds to the set of gradients of smooth functions that are below ψ and coincide with it at $\boldsymbol{w}$. In the case of (3.2), it takes a simple form (Rockafellar and Wets, 2009, Theorem 10.6):

$$\partial(h \circ \boldsymbol{g}_i)(\boldsymbol{w}) = \nabla \boldsymbol{g}_i(w)^\top \partial h\big(\boldsymbol{g}_i(w)\big) \,,$$

where ∂h refers to the convex subdifferential of h and $\nabla \boldsymbol{g}_i$ refers to the Jacobian of $\boldsymbol{g}_i$. The number of first-order oracle calls of $\boldsymbol{g}_i$ and h for SGD to reach an ε-stationary point is $O(1/\varepsilon^4)$ (Davis and Drusvyatskiy, 2019). In deep learning, the subgradient $\boldsymbol{v} \in \partial(h \circ \boldsymbol{g}_i(\boldsymbol{w}))$ requires the computation of the so-called vector-Jacobian product. It is readily given by reverse-mode automatic differentiation implemented in software such as PyTorch and Tensorflow.

The Prox-linear Method. The modified Gauss-Newton method (Nesterov, 2007), also known as the prox-linear method (Burke, 1985; Burke and Ferris, 1995; Drusvyatskiy and Paquette, 2019), applied to the finite-sum objective (3.2) proceeds by finding approximate solutions of a partially linearized approximation of the objective with an additional regularization term. Namely, it computes iterates of the form

$$\boldsymbol{w}_{k+1} = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ \frac{1}{n}\sum_{i=1}^{n} h(\boldsymbol{g}_i(\boldsymbol{w}_k) + \nabla \boldsymbol{g}_i(\boldsymbol{w}_k)^\top (\boldsymbol{w} - \boldsymbol{w}_k)) + \frac{\kappa}{2}\|\boldsymbol{w} - \boldsymbol{w}_k\|_2^2 \right\} \tag{3.4}$$

The subproblem within the argmin is convex. As explained in Figure 3.1 (left), the prox-linear method creates a convex model $C(\boldsymbol{w}; \boldsymbol{w}_k)$ of F around $\boldsymbol{w}_k$ by linearizing the inner function as

$$C(\boldsymbol{w}; \boldsymbol{w}_k) = \frac{1}{n}\sum_{i=1}^{n} h(\boldsymbol{g}_i(\boldsymbol{w}_k) + \nabla \boldsymbol{g}_i(\boldsymbol{w}_k)^\top (\boldsymbol{w} - \boldsymbol{w}_k)) \,.$$

The next iterate is the minimizer $\boldsymbol{w}_{k+1} = \arg\min_{\boldsymbol{w}} C_\kappa(\boldsymbol{w}; \boldsymbol{w}_k)$ where

$$C_\kappa(\boldsymbol{w}; \boldsymbol{w}_k) = C(\boldsymbol{w}; \boldsymbol{w}_k) + \frac{\kappa}{2}\|\boldsymbol{w} - \boldsymbol{w}_k\|^2$$

is the model plus a proximal term. Compare this with the subgradient method, where the model is $C'(\boldsymbol{w}; \boldsymbol{w}_k) = F(\boldsymbol{w}_k) + \boldsymbol{v}^\top(\boldsymbol{w} - \boldsymbol{w}_k)$, where $\boldsymbol{v} \in \partial F(\boldsymbol{w}_k)$; see also Figure 3.1 (right).

In practice, it might not be possible to solve the subproblem (3.4) exactly. We consider using incremental algorithms such as Casimir-SVRG of Chapter 2 in this case and use the abbreviation *PLI*.

From a computational viewpoint, this requires computing a Jacobian vector product $\nabla \boldsymbol{g}_i(\boldsymbol{w})\boldsymbol{v}$ for some vector $\boldsymbol{v}$. This is most efficiently computed via forward-mode automatic differentiation for efficient implementation, which enjoys limited support in modern software libraries such as PyTorch. In our experiments, we simulate forward-mode automatic differentiation using two calls to reverse-mode automatic differentiation.

3.2 Tradeoffs of the Prox-linear Method in Statistical Settings

The closely related family of Gauss-Newton methods and its variants are known to enjoy quadratic local convergence, provided the subproblems are solved exactly. In statistical learning problems, it is not meaningful to optimize beyond the noise level of the problem. If the noise level of the problem is large, then the local quadratic convergence may not be meaningful. In this section, we formalize this in a stylized statistical setting with the exact prox-linear algorithm.

We start with a typical quadratic local convergence result of the prox-linear method in the overparameterized regime $d > nk$. We show the quadratic convergence under the assumption that the minimal singular value $\sigma_{\min}(\nabla \boldsymbol{g}(\boldsymbol{w})^\top)$ of the transposed Jacobian of $\boldsymbol{g} = (\boldsymbol{g}_1; \cdots; \boldsymbol{g}_n)$ is strictly positive. This implies that the Jacobian $\nabla \boldsymbol{g}(\boldsymbol{w})$ is surjective; this is only possible in the overparameterized setting of $d > nk$.

Proposition 3.1. *Consider problem* (3.2) *where h is ℓ-Lipschitz, convex, and α-sharp for some $\alpha > 0$, i.e., $h(\boldsymbol{u}) - \min h \geq \alpha \operatorname{dist}(\boldsymbol{u}, U^*)$ for any $\boldsymbol{u} \in \mathbb{R}^m$ where $\operatorname{dist}(\boldsymbol{u}, U^*)$ is the Euclidean distance of $\boldsymbol{u}$ to $U^* = \arg\min_{\boldsymbol{u}\in\mathbb{R}^m} h(\boldsymbol{u}) \neq \emptyset$. Further, suppose the function $\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{g}_1(\boldsymbol{w}); \ldots; \boldsymbol{g}_n(\boldsymbol{w})) \in \mathbb{R}^{nm}$ is L-smooth and satisfies $\sigma_{\min}(\nabla \boldsymbol{g}(\boldsymbol{w})^\top) \geq \nu > 0$ for any $\boldsymbol{w} \in \mathbb{R}^d$. Then, we have the following:*

(a) The minimal value of F is the same as the minimum value of h, i.e.,

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) = \min_{\boldsymbol{u}\in\mathbb{R}^m} h(\boldsymbol{u})\,.$$

(b) The sequence $(\boldsymbol{w}_k)_{k=0}^{\infty}$ produced by the exact prox-linear algorithm (3.4) with $\kappa = L\ell$ starting from arbitrary $\boldsymbol{w}_0 \in \mathbb{R}^d$ converges globally to its minimum value $F(\boldsymbol{w}_k) \to F^* := \min F$.

(c) As soon as an iterate $\boldsymbol{w}_j$ satisfies $F(\boldsymbol{w}_j) - F^* \le (\alpha\nu)^2/(L\ell n^2)$, the subsequence $(\boldsymbol{w}_k)_{k=j}^{\infty}$ converges quadratically as

$$F(\boldsymbol{w}_{k+1}) - F^* \le \frac{L\ell n^2}{2(\alpha\nu)^2}\big(F(\boldsymbol{w}_k) - F^*\big)^2\,. \tag{3.5}$$

The proof is given in Appendix A.7.

Statistical Setting. In a statistical setting, we might not enjoy the quadratic convergence due to the noise in the problem. Suppose we are given n input-output pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ where $\boldsymbol{y}_i = \varphi(\boldsymbol{x}_i; \overline{\boldsymbol{w}}) + \boldsymbol{\xi}_i$ from a parameterized nonlinear function $\varphi(\,\cdot\,; \overline{\boldsymbol{w}}) : \mathbb{R}^p \to \mathbb{R}^m$ with parameters $\overline{\boldsymbol{w}} \in \mathbb{R}^d$, and $\boldsymbol{\xi}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_m)$ is i.i.d. Gaussian noise. We wish to recover the true signal $\overline{\boldsymbol{w}}$'s predictions, i.e. find $\hat{\boldsymbol{w}}$ such that $\varphi(\boldsymbol{x}_i; \hat{\boldsymbol{w}}) \approx \varphi(\boldsymbol{x}_i; \overline{\boldsymbol{w}})$ for each i. We instantiate (3.2) with $\boldsymbol{g}_i(\boldsymbol{w}) = \varphi(\boldsymbol{x}_i; \boldsymbol{w}) - \boldsymbol{y}_i$ and $h = \|\cdot\|_2$ in order to solve this problem. This is different from the related choice of $h = \|\cdot\|_2^2$, which corresponds to non-linear least squares regression in the fixed design setting.

We consider *early stopping* of the optimization once we reach the noise level. That is, we stop the optimization once the objective value $F(\boldsymbol{w}_k)$ in iteration k falls below $F(\overline{\boldsymbol{w}})$. In this setting, we now show that the prox-linear method can enjoy quadratic local convergence only when the noise level σ of the problem is small enough. To this end, we make a general assumption on the radius R of local quadratic convergence; Proposition 3.1 provides a lower bound on R in the sharp and overparameterized setting.

Proposition 3.2. *Fix some $\delta \in (0,1)$ and consider problem (3.2) with $\boldsymbol{g}_i$ and h as defined above with the output dimension $m \ge 4\log(2n/\delta)$. Suppose that (i) $\boldsymbol{w} \mapsto \varphi(\boldsymbol{x}_i; \boldsymbol{w})$ is L-smooth for each $i \in [n]$, and, (ii) the function φ can interpolate the data so that $\varphi(\boldsymbol{x}_i; \boldsymbol{w}^*) = \boldsymbol{y}_i$ for each $i \in [n]$ for some $\boldsymbol{w}^* \in \mathbb{R}^d$. Suppose there exists a scalar $R > 0$ and an integer j such that for all integers $k \ge j$, we have*

$$F(\boldsymbol{w}_k) - \min F \le R\,, \quad \text{and} \quad F(\boldsymbol{w}_{k+1}) - \min F \le \frac{1}{2R}\big(F(\boldsymbol{w}_k) - \min F\big)^2\,. \tag{3.6}$$

Then, we have the following with probability at least $1-\delta$:

(a) If the noise level satisfies

$$\sigma > \frac{R}{\sqrt{m}}\left(1 + \left(\frac{16}{m}\log(2n/\delta)\right)^{1/4}\right)^{-1}\,,$$

then the first iterate $\boldsymbol{w}_k$ enjoying quadratic convergence (3.6) satisfies $F(\boldsymbol{w}_k) < F(\overline{\boldsymbol{w}})$.

(b) Conversely, if the noise level satisfies

$$\sigma < \frac{R}{\sqrt{m}}\left(1 - \left(\frac{4}{m}\log(2n/\delta)\right)^{1/4}\right)^{-1}\,,$$

then the first iterate $\boldsymbol{w}_k$ enjoying quadratic convergence (3.6) satisfies $F(\boldsymbol{w}_k) > F(\overline{\boldsymbol{w}})$.

Proof. First, under the interpolation assumption, we have that $0 \leq \min F \leq F(\boldsymbol{w}^*) = 0$, so $\min F = 0$. Therefore, under this setting, quadratic convergence holds before the noise level if and only if $F(\overline{\boldsymbol{w}}) \geq R$. To complete the proof, we show below that with probability at least $1 - \delta$ that

$$\sigma\sqrt{m}\left(1 - \left(\frac{4}{m}\log(n/\delta)\right)^{1/4}\right) \leq F(\overline{\boldsymbol{w}}) \leq \sigma\sqrt{m}\left(1 + \left(\frac{16}{m}\log(n/\delta)\right)^{1/4}\right). \tag{3.7}$$

To this end, we simplify

$$F(\overline{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} \|\varphi(\boldsymbol{x}_i; \overline{\boldsymbol{w}}) - \boldsymbol{y}_i\|_2 = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{\xi}_i\|_2 \,. \tag{3.8}$$

Noting that $\|\boldsymbol{\xi}_i\|_2^2$ follows a χ^2 distribution with m degrees of freedom, a standard concentration argument shows that (see example Laurent and Massart (2000, Lemma 1))

$$\mathbb{P}\left(\sigma^2 m\left(1 - 2\sqrt{\frac{\lambda}{m}}\right) \leq \|\boldsymbol{\xi}_i\|_2^2 \leq \sigma^2 m\left(1 + 2\sqrt{\frac{\lambda}{m}} + \frac{2\lambda}{m}\right)\right) \geq 1 - 2\exp(-\lambda)$$

for any $\lambda > 0$. Plugging in $\lambda = \log(2n/\delta)$ and invoking the union bound, we have with probability at least $1 - \delta$ that

$$\sigma\sqrt{m}\left(1 - \left(\frac{4}{m}\log(n/\delta)\right)^{1/4}\right) \leq \|\xi_i\|_2 \leq \sigma\sqrt{m}\left(1 + \left(\frac{16}{m}\log(n/\delta)\right)^{1/4}\right)$$

holds simultaneously for each $i \in [n]$. Plugging this into (3.8) completes the proof. □

Proposition 3.2 shows that the potential advantages of the prox-linear method in terms of local quadratic convergence may not be relevant in some statistical problems where the noise level isw large.

3.3 The Prox-linear Method with Incremental Inner Loops

In general, we cannot solve subproblems (3.4) of the prox-linear method exactly. Provided that the outer functions h are convex, the subproblems (3.4) can be solved to any accuracy by e.g. a subgradient method. To take advantage of the finite-sum structure, we follow the recipe of Chapter 2 by smoothing the outer function by infimal convolution in order to obtain a smooth and strongly convex objective. The latter can then be solved at a linear rate by any fast incremental algorithm for strongly convex objectives such as SVRG (Johnson and Zhang, 2013), SAG (Schmidt et al., 2017), SARAH (Nguyen et al., 2017) or SPIDER (Fang et al., 2018). We refer to this as *PLI*, standing for Prox-Linear with Incremental Gradient inner loop. Theoretically, a regularization $\kappa = L\ell$ suffices to ensure the convergence of the algorithm (Drusvyatskiy and Paquette, 2019, Proposition 3.3).

Here, we directly incorporate the smoothing operation into the prox-linear analysis and present an algorithm that uses the gradient norm of the smooth objective as a criterion. We also provide a new accelerated algorithm with small total complexity.

3.3.1 Smoothing and Approximate Inner Minimization

When given a function $f = h \circ \boldsymbol{g}$ with h convex and $\boldsymbol{g}$ differentiable, the prox-linear algorithm defines a local convex approximation $c(\cdot\,; \boldsymbol{w})$ around current iterate $\boldsymbol{w} \in \mathbb{R}^d$ by linearizing the smooth map $\boldsymbol{g}$ as $c(\boldsymbol{z}; \boldsymbol{w}) := h(\boldsymbol{g}(\boldsymbol{w}) + \nabla \boldsymbol{g}(\boldsymbol{w})(\boldsymbol{z} - \boldsymbol{w}))$. With this, it builds a convex model $C(\cdot; \boldsymbol{w})$ of the finite sum F about $\boldsymbol{w}$ as

$$C(\boldsymbol{z}; \boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} h(\boldsymbol{g}^{(i)}(\boldsymbol{w}) + \nabla \boldsymbol{g}^{(i)}(\boldsymbol{w})(\boldsymbol{z} - \boldsymbol{w})) \tag{3.9}$$

The prox-linear algorithm minimizes these models with an additional proximal-term (Burke, 1985). Following Drusvyatskiy and Paquette (2019), we consider only approximate solutions of these models. In particular, we focus on having access to fast incremental order to solve the sub-problems. As before, it means smoothing the model as

$$C_{\mu\omega}(\boldsymbol{z}; \boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} h_{\mu\omega}(\boldsymbol{g}^{(i)}(\boldsymbol{w}) + \nabla \boldsymbol{g}^{(i)}(\boldsymbol{w})(\boldsymbol{z} - \boldsymbol{w})) \tag{3.10}$$

where $h_{\mu\omega}$ is a smooth surrogate of h obtained by infimal convolution smoothing as in Section 2.2. We then use fast incremental algorithms on the smooth strongly convex objective

$$C_{\mu\omega,\kappa}(\boldsymbol{z}; \boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} h_{\mu\omega}(\boldsymbol{g}^{(i)}(\boldsymbol{w}) + \nabla \boldsymbol{g}^{(i)}(\boldsymbol{w})(\boldsymbol{z} - \boldsymbol{w})) + \frac{\kappa}{2} \|\boldsymbol{z} - \boldsymbol{w}\|_2^2 \,, \tag{3.11}$$

defining next iterate as an approximate minimizer

$$\boldsymbol{w}_k \approx \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{R}^d} C_{\mu\omega,\kappa}(\boldsymbol{z}; \boldsymbol{w}_{k-1}),$$

where κ^{-1} defines the step-size of the procedure, that is the smaller κ^{-1}, the closer $\boldsymbol{w}_k$ to $\boldsymbol{w}_{k-1}$.

Concretely, the prox-linear outer loop is displayed in Algorithm 3.1. We detail its convergence guarantees in Section 3.3.2. Section 3.3.3 presents an accelerated variant that attempts to make extrapolated steps. We give the total complexity using Catalyst as the inner solver in Section 3.3.4.

Once again, similar to the convex case in chapter 2, a fixed iteration budget can be used instead of the theoretical stopping criterion.

As shown later, the convergence of the scheme is ensured for a sufficiently large regularization, i.e. for κ larger than a constant depending on the objective. If this constant is not known, one can use the *auto-adapt* procedure described by (Paquette et al., 2018), i.e. fixing the number of iterations made for each subproblem and increasing the regularization until the stopping criterion is met.

3.3.2 Convergence Analysis

We first make some assumptions needed for the convergence of the prox-linear method and interpret them for structured prediction. We then recall the measure of stationarity and overall convergence.

Assumptions. Convergence of the prox-linear is only ensured if the convex models approximate the objective up to a quadratic error, i.e., if there exists a constant $L > 0$ such that for all $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$ and $i \in [n]$,

$$|h\big(\boldsymbol{g}^{(i)}(\boldsymbol{w} + \boldsymbol{z})\big) - h\big(\boldsymbol{g}^{(i)}(\boldsymbol{w}) + \nabla \boldsymbol{g}^{(i)}(\boldsymbol{w})\boldsymbol{z}\big)| \leq \frac{L}{2} \|\boldsymbol{z}\|_2^2 \,. \tag{3.14}$$

Algorithm 3.1. (Inexact) Prox-linear algorithm: outer loop

Input: Objective F of the form (3.2), linearly convergent algorithm $\mathcal{M}$, initial point $\boldsymbol{w}_0$, regularization κ, smoothing parameters $(\mu_k)_{k\geq 1}$

1: **for** $k = 1, \cdots, K$ **do**

2: Using $\mathcal{M}$ with $\boldsymbol{w}_{k-1}$ as the starting point, find

$$\boldsymbol{w}_k \approx \arg\min_{\boldsymbol{z}\in\mathbb{R}^d} C_{\mu_k\omega,\kappa}(\boldsymbol{z}; \boldsymbol{w}_{k-1}) \tag{3.12}$$

where $C_{\mu_k\omega,\kappa}(\boldsymbol{z}; \boldsymbol{w}_{k-1})$ is defined in (3.11) such that

$$\|\nabla C_{\mu_k\omega,\kappa}(\boldsymbol{w}_k; \boldsymbol{w}_{k-1})\|_2 \leq \frac{\kappa}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2 \,. \tag{3.13}$$

3: **return** $\boldsymbol{w}_K$.

When h is G-Lipschitz and each $\boldsymbol{g}^{(i)}$ is $\widetilde{L}$-smooth, both with respect to $\|\cdot\|_2$, then this assumption holds with $L = G\widetilde{L}$ (Drusvyatskiy and Paquette, 2019). To analyze the behavior fo the prox-linear algorithm for decreasing smoothing parameters, we will need that this property holds both for the original and the smoothing function as stated in the following assumption.

Assumption 3.3. Consider $f(\boldsymbol{w}) = h \circ \boldsymbol{g}(\boldsymbol{w})$ where g is differentiable and h can be smoothed by a smoothing function ω as in Definition A.1. We assume that for all $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$, $\mu \geq 0$,

$$|h(\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z})) - h(\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})| \leq \frac{L}{2}\|\boldsymbol{z}\|_2^2 \,,$$
$$|h_{\mu\omega}(\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z})) - h_{\mu\omega}(\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})| \leq \frac{L}{2}\|\boldsymbol{z}\|_2^2 \,.$$

In the case of structured prediction, it holds when the augmented score ψ as a function of $\boldsymbol{w}$ is L-smooth, as stated below. Its proof is given in Appendix A.6.1.

Lemma 3.4. *Consider the structural hinge loss $f(\boldsymbol{w}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w}) = h \circ \boldsymbol{g}(\boldsymbol{w})$ where $h, \boldsymbol{g}$ are as defined in* (2.11)*. If the augmented score function $\boldsymbol{w} \mapsto \psi(\boldsymbol{y}; \boldsymbol{w})$ is L-smooth with respect to $\|\cdot\|_2$ for all $\boldsymbol{y} \in \mathcal{Y}$, then for all $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$*

$$|h(\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z})) - h(\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})| \leq \frac{L}{2}\|\boldsymbol{z}\|_2^2 \,,$$

The same holds if h is replaced by its smoothing $h_{\mu\omega}$.

Recall that the smoothing of h by a strongly convex function ω satisfying

$$0 \leq \omega(\boldsymbol{u}) \leq D, \qquad \text{for all } \boldsymbol{u} \in \operatorname{dom} h^* \tag{3.15}$$

enables access to smooth surrogates $F_{\mu\omega}$ and $C_{\mu\omega}(\cdot; \boldsymbol{w})$ of respectively the objective F and convex models $C(\cdot; \boldsymbol{w})$ that satisfy for any $\mu_1 \geq \mu_2 \geq 0$, and any $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$,

$$0 \leq F_{\mu_2\omega}(\boldsymbol{w}) - F_{\mu_1\omega}(\boldsymbol{w}) \leq (\mu_1 - \mu_2)D, \tag{3.16}$$
$$0 \leq C_{\mu_2\omega}(\boldsymbol{z}; \boldsymbol{w}) - C_{\mu_1\omega}(\boldsymbol{z}; \boldsymbol{w}) \leq (\mu_1 - \mu_2)D. \tag{3.17}$$

Assumption 3.3 adds the following assumptions on the models for any $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$, $\mu \geq 0$,

$$|F(\boldsymbol{w}) - C(\boldsymbol{z}; \boldsymbol{w})| \leq \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{w}\|_2^2, \tag{3.18}$$

$$|F_{\mu\omega}(\boldsymbol{w}) - C_{\mu\omega}(\boldsymbol{z}; \boldsymbol{w})| \leq \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{w}\|_2^2. \tag{3.19}$$

This control on the quality of the approximations of the smoothed convex models allows us to characterize the convergence of the method.

Convergence guarantee. While classical gradient methods use the norm of the gradient to measure convergence, the prox-linear method is better analyzed as a fixed point algorithm. The norm of the difference of iterates can indeed be related to near stationarity as demonstrated by Drusvyatskiy and Lewis (2018), whose proposition is adapted below. Its proof is provided in Appendix A.6.1.

Proposition 3.5. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. For any $\boldsymbol{w} \in \mathbb{R}^d$ and $\mu \geq 0, \kappa \geq 2L$, if $\boldsymbol{z} \in \mathbb{R}^d$ approximately minimizes the model $C_{\mu\omega,\kappa}$ as*

$$\|\nabla C_{\mu\omega,\kappa}(\boldsymbol{z}; \boldsymbol{w})\|_2 \leq \frac{\kappa}{2}\|\boldsymbol{z} - \boldsymbol{w}\|_2,$$

then there exists $\hat{\boldsymbol{z}} \in \mathbb{R}^d$ such that

$$\|\hat{\boldsymbol{z}} - \boldsymbol{z}\|_2 \leq \|\boldsymbol{z} - \boldsymbol{w}\|_2 + \sqrt{\mu D} \qquad \text{and} \qquad d(0, \partial F(\hat{\boldsymbol{z}})) \leq 5\kappa\|\boldsymbol{z} - \boldsymbol{w}\|_2 + 3\sqrt{\mu\kappa D}.$$

where $d(0, \partial F(\hat{\boldsymbol{z}})) = \inf_{\boldsymbol{z} \in \partial F(\hat{\boldsymbol{z}})} \|\boldsymbol{z}\|_2$.

The difference of iterates is thus a measure of near-stationarity as follows.

Corollary 3.6. *Given assumptions of Proposition 3.5, and using $\kappa \geq 2L$ for Algorithm 3.1, if for $k \geq 1$,*

$$5\kappa\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2 \leq \varepsilon/2$$

then there exists a point $\hat{\boldsymbol{z}} \in \mathbb{R}^d$ that is $\varepsilon/2 + 3\sqrt{\mu_k\kappa D}$ near-stationary and $\varepsilon/10\kappa + \sqrt{\mu_k D}$ close to $\boldsymbol{w}_k$.

The difference of iterates can then be controlled by an appropriate step-size and relative accuracy threshold as shown in the following theorem — the proof is given in Appendix A.6.

Theorem 3.7. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. Feed Algorithm 3.1 with a regularization $\kappa \geq 2L$ and non-increasing smoothing parameters $(\mu_k)_{k\geq 1}$, then it produces a sequence $(\boldsymbol{w}_k)_{k\geq 0}$ that satisfies*

$$\min_{k \in \{1,\ldots,K\}} \kappa^2\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq \frac{4\kappa}{K}\big(F(\boldsymbol{w}_0) - F^* + \mu_1 D\big),$$

where $F^* = \inf_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w})$.

For a target accuracy ε one can simply take a constant smoothing $\mu = \varepsilon^2/\kappa D$ to get an iterate $\varepsilon(1/\kappa + 1/\sqrt{\kappa})$-close to an ε-near stationary point after

$$O\left(\frac{\kappa(F(\boldsymbol{w}_0) - F^*)}{\varepsilon^2} + 1\right)$$

outer iterations of the prox-linear. For a decreasing accuracy, the result still holds as shown in the following corollary. Note that the analysis does not incorporate an additional log term as required by (Drusvyatskiy and Paquette, 2019, Thm. 5.2)

Corollary 3.8. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. Taking $\mu_k = \mu_0/k$, $\kappa \geq 2L$ as inputs for Algorithm 3.1 ensures that there exists an iterate $\varepsilon(1/\kappa + 1/\sqrt{\kappa})$-close to an ε-near stationary point after at most*

$$O\left(\frac{\kappa(F(\boldsymbol{w}_0) - F^* + 3\mu_0 D)}{\varepsilon^2}\right)$$

outer iterations of the prox-linear method.

Proof. After $k_\varepsilon = \mu_0\kappa D/\varepsilon^2$ iterations the smoothing approximation measured by $\sqrt{\mu_k\kappa D}$ for $k \geq k_\varepsilon$ are below the target accuracy, i.e. $\sqrt{\mu_k\kappa D} \leq \varepsilon$. Then, following the proof of Theorem 3.7, we have for $K > k_\varepsilon$,

$$\begin{aligned}
\min_{k\in\{k_\varepsilon+1,\ldots,K\}} \|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 &\leq \frac{4}{\kappa(K-k_\varepsilon)}\big(F_{\mu_{k_\varepsilon}\omega}(\boldsymbol{w}_{k_\varepsilon}) - F_{\mu_K\omega}(\boldsymbol{w}_K) + (\mu_{k_\varepsilon} - \mu_K)D\big), \\
&\leq \frac{4}{\kappa(K-k_\varepsilon)}\big(F(\boldsymbol{w}_{k_\varepsilon}) - F(\boldsymbol{w}_K) + \mu_1 D\big), \\
&\leq \frac{4}{\kappa(K-k_\varepsilon)}\big(F(\boldsymbol{w}_0) - F^* + 2\mu_0 D\big)
\end{aligned}$$

where we used from (A.46) that $F(\boldsymbol{w}_{k_\varepsilon}) \leq F(\boldsymbol{w}_0) + \mu_0 D$ and $\mu_1 = \mu_0$. Therefore for $K - k_\varepsilon \geq \frac{4\kappa}{\varepsilon^2}\big(F(\boldsymbol{w}_0) - F^* + 2\mu_0 D\big)$, there exists an iterate in $\{k_\varepsilon + 1, \ldots, K\}$ such that $\kappa\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\| \leq \varepsilon$ and its smoothing constant is below the target accuracy. Proposition 3.5 concludes the claim. □

3.3.3 The Accelerated Prox-linear Algorithm

We present a variant of the prox-linear method that attempts to make extrapolated steps in order to accelerate the convergence when the convex models lower bound the objective. In the spirit of (Paquette et al., 2018), the method presented in Algorithm 3.2 takes one classical prox-linear step on one hand and an extrapolation step to potentially accelerate the algorithm on the other hand. We then take the best of both of these as the next iterate to ensure convergence to a near-stationary point.

Convergence of the accelerated variant follows from the next proposition. Its proof is analogous to that of Theorem 2.17 from Chapter 2. The additional requirement here is to have sufficient regularization as one minimizes a model of the function and not the function itself. To ensure acceleration, the models must lower bound the objective as in the convex case.

Proposition 3.9. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. Taking $\bar{\kappa} \geq 2L$ and non-increasing smoothing parameters μ_k, the iterates produced by Algorithm 3.2 satisfy*

$$\min_{k\in\{1,\ldots,K\}} \bar{\kappa}^2\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq \frac{4\bar{\kappa}}{K}\big(F(\boldsymbol{w}_0) - F^* + \mu_1 D\big).$$

If, moreover, the smooth convex models lower bound the smooth objective, i.e. for $k \geq 1$ and $\boldsymbol{w}^* \in \arg\min_w F(w)$

$$C_{\mu_k\omega}(\boldsymbol{w}_{k-1}; \boldsymbol{z}_{k-1}) \leq F_{\mu_k\omega}(\boldsymbol{w}_{k-1}) \quad \text{and} \quad C_{\mu_k\omega}(\boldsymbol{w}^*; \boldsymbol{z}_{k-1}) \leq F_{\mu_k\omega}(\boldsymbol{w}^*), \tag{3.23}$$

then the algorithm converges for non-decreasing regularization parameters $\kappa_k \geq 2L$ as

$$F(\boldsymbol{w}_k) - F^* \leq \frac{\mathcal{A}_0^{k-1}}{\mathcal{B}_1^k}\Delta_0 + \mu_k D_\omega + \sum_{j=1}^{k} \frac{\mathcal{A}_j^{k-1}}{\mathcal{B}_j^k}\left(\mu_{j-1} - (1-\delta_j)\mu_j\right) D_\omega\,,$$

Algorithm 3.2. Accelerated prox-linear method

Input: Objective F of the form (3.2), linearly convergent algorithm $\mathcal{M}$, initial $\boldsymbol{w}_0, \alpha_0 \geq 0$, smoothing parameters μ_k, fixed and varying regularizations $\bar{\kappa}$, κ_k, relative target accuracies δ_k

1: **Initialize:** $\boldsymbol{z}_0 = \boldsymbol{w}_0$.

2: **for** $k = 1, \cdots, K$ **do**

3: Prox-linear step: Compute $\boldsymbol{\xi}_k$ using $\mathcal{M}$, such that

$$\boldsymbol{\xi}_k \approx \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{R}^d} C_{\mu_k \omega, \bar{\kappa}}(\boldsymbol{z}; \boldsymbol{w}_{k-1}) \quad \text{where} \qquad \|\nabla C_{\mu_k \omega, \kappa}(\boldsymbol{\xi}_k; \boldsymbol{w}_{k-1})\|_2 \leq \frac{\bar{\kappa}}{2} \|\boldsymbol{\xi}_k - \boldsymbol{w}_{k-1}\|_2 \,.$$

4: Accelerated step:

1. Set
$$\boldsymbol{z}_{k-1} = \alpha_{k-1} \boldsymbol{z}_{k-1} + (1 - \alpha_{k-1}) \boldsymbol{w}_{k-1}. \tag{3.20}$$

2. Compute $\boldsymbol{\zeta}_k$ using $\mathcal{M}$ such that
$$\boldsymbol{\zeta}_k \approx \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{R}^d} C_{\mu_k \omega, \kappa_k}(\boldsymbol{z}; \boldsymbol{z}_{k-1})$$
$$\text{where} \qquad \|\nabla C_{\mu_k \omega, \kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1})\|_2 \leq \frac{\kappa_k \sqrt{\delta_k}}{\sqrt{2}} \|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|_2 \tag{3.21}$$

3. Set
$$\boldsymbol{z}_k = \boldsymbol{w}_{k-1} + \frac{1}{\alpha_{k-1}} (\boldsymbol{\zeta}_k - \boldsymbol{w}_{k-1}).$$

4. Pick $\alpha_k \in (0, 1)$ satisfying
$$\frac{1 - \alpha_k}{\alpha_k^2} = \frac{\kappa_{k+1}}{\kappa_k \alpha_{k-1}^2}. \tag{3.22}$$

5: Pick best of both: Set $\boldsymbol{w}_k$ such that $F_{\mu_k \omega}(\boldsymbol{w}_k) \leq \min\{F_{\mu_k \omega}(\boldsymbol{\xi}_k), F_{\mu_k \omega}(\boldsymbol{\zeta}_k)\}$.

6: **return** $\boldsymbol{w}_K$

where $\mathcal{A}_i^j := \prod_{r=i}^j (1 - \alpha_r)$, $\mathcal{B}_i^j := \prod_{r=i}^j (1 - \delta_r)$, $\Delta_0 := F(\boldsymbol{w}_0) - F^* + \frac{\kappa_1 \alpha_0^2}{2(1-\alpha_0)} \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2$ *and* $\mu_0 := 2\mu_1$.

The Casimir algorithm from Chapter 2 (in particular, Corollaries 2.20 and 2.21) readily apply to get concrete rates when the convex models lower bound the function. The only additional requirement is to provide regularization parameters sufficiently large, i.e. $\kappa_k \geq 2L$.

Drusvyatskiy and Paquette (2019) also proposed an accelerated scheme for the prox-linear method. For unconstrained problems such as the ones we consider here, they require that the convex models $C(\cdot\,; w)$ lower bound the objective F at any point, an assumption stronger than convexity. It is not clear if our assumption is stronger or weaker than theirs. Secondly, their overall complexity depends on subproblems whose condition number grows with time. Therefore, the inner loop complexity, and hence the total complexity, grow with time. The scheme presented here circumvents this issue by taking extrapolated steps rather than a combination of short and long steps.

3.3.4 Prox-linear with Casimir-SVRG as the Inner Loop

We now analyze the total complexity of minimizing the finite sum problem (3.2) with Casimir-SVRG to approximately solve the subproblems of Algorithm 3.1. The total complexity of the accelerated scheme when the models lower bound the objective follows from the convex case.

For the algorithm to converge, the map $\boldsymbol{w} \mapsto \boldsymbol{g}^{(i)}(\boldsymbol{w}_k) + \nabla \boldsymbol{g}^{(i)}(\boldsymbol{w}_k)(\boldsymbol{w} - \boldsymbol{w}_k)$ must be Lipschitz for each i and each iterate $\boldsymbol{w}_k$. To be precise, we assume that

$$A_\omega := \max_{i \in \{1, \cdots, n\}} \sup_{\boldsymbol{w} \in \mathbb{R}^d} \|\nabla \boldsymbol{g}^{(i)}(\boldsymbol{w})\|_{2,\alpha}^2 \tag{3.24}$$

is finite, where ω, the smoothing function, is 1-strongly convex with respect to $\|\cdot\|_\alpha$. When $\boldsymbol{g}^{(i)}$ is the linear map $\boldsymbol{w} \mapsto \boldsymbol{A}^{(i)}\boldsymbol{w}$, this reduces to the assumption in Eq. (2.37). The parameter A_ω characterizes the smoothness of each subproblem and is necessary to run incremental algorithms. In practice, we can search for it on small mini-batches of the problem before running the algorithm on the whole data set, or treat it like another hyperparameter to be tuned.

We choose the smoothing parameters μ_k to decrease as $1/k$ and use the Catalyst algorithm with SVRG (Casimir-SVRG without smoothing) as the inner solver to solve each sub-problem. This results in the following rate of convergence for the prox-linear, which is proved in Appendix A.6.2.

Proposition 3.10. *Consider the setting of Corollary 3.8 and that the subproblem of Line 2 of Algorithm 3.1 is solved using Catalyst-SVRG. Then, total number of SVRG iterations N required to produce an iterate $\varepsilon(1/\kappa + 1/\sqrt{\kappa})$close to an ε-near stationary is bounded as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(\kappa \frac{\sqrt{A_\omega n}}{\varepsilon^3}(\Delta F_0 + \mu_0 D)^{3/2}\right).$$

Note that this sample complexity is smaller than the $O(1/\varepsilon^4)$ required by the stochastic subgradient method.

3.4 Experiments

We consider three different experimental setups: multi-output regression, structured prediction, and solving stochastic non-linear equations. In all cases, the hyper-parameters have been tuned by a grid search.

Synthetic Multi-output Regression. We consider a regression task of predicting output $\boldsymbol{y} \in \mathbb{R}^k$ from input $\boldsymbol{x} \in \mathbb{R}^p$, given a synthetic dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ of input-output pairs of varying size n. We use $p = 128$ and $k = 10$ throughout. We sample the each input $\boldsymbol{x}_i$ i.i.d. from a zero-mean normal distribution with covariance matrix Σ exhibiting a $1/j^2$ spectral decay. The output is generated as $\boldsymbol{y}_i = \phi^\star(\boldsymbol{x}_i; \boldsymbol{w}^\star) + \sigma\boldsymbol{\xi}_i$, where $\phi^\star(\cdot\,; \boldsymbol{w}^\star)$ is a multilayer perceptron (MLP) with one hidden layer of width 256, and standard normal weights $\boldsymbol{w}^\star$, while $\boldsymbol{\xi}_i$ is sampled from a standard Laplace distribution in $\mathbb{R}^k$ and σ is the noise scale, which we vary. We define the signal-to-noise ratio (SNR) of a problem instance as $\text{SNR} = \|\boldsymbol{w}^\star\|^2/\sigma^2 \propto 1/\sigma^2$. Finally, the loss and evaluation measure we use is the ℓ_2 loss (without the square).

We vary the number of samples n and the SNR (equivalently, σ) and compare the two methods introduced in Section 3.1.2: the stochastic subgradient method (SGD) and prox-linear with incremental gradient inner loop (PLI). For each instance, we tune the hyperparameters to achieve the smallest loss on a held-out validation dataset in 100 epochs and report the test loss. We run the experiment in two regimes: (a)

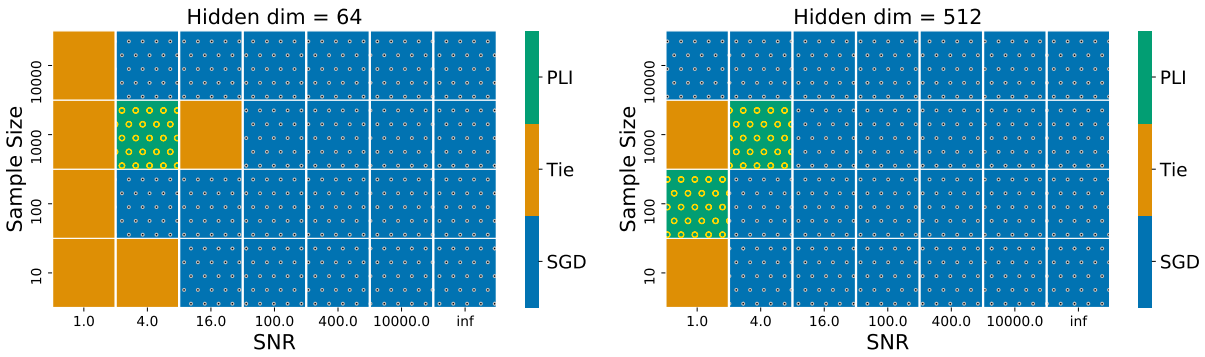


Figure 3.2: Synthetic multi-output regression: comparing the stochastic subgradient method (SGD) and the prox-linear with incremental gradient inner-loop (PLI) for various choices of the number of samples n and the signal-to-noise ratio (SNR). We highlight the method which finds the smallest test ℓ_2 loss.

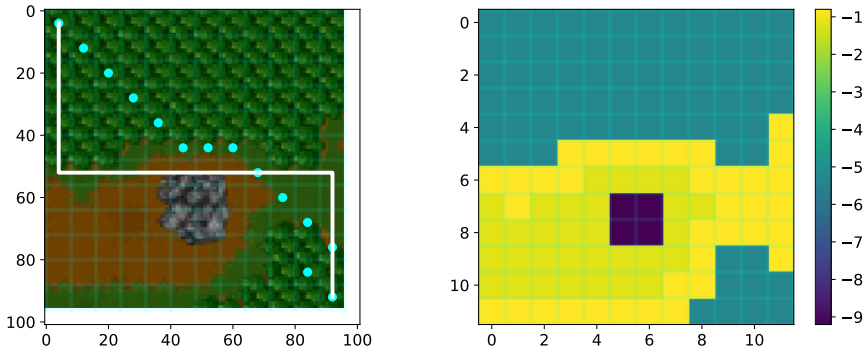


Figure 3.3: Left: example map and associated best path (in solid white lines), Right: associated reward for each cell.

under-parameterized, where the model is a MLP with 64 hidden units, and, (b) over-parameterized, where the MLP has 512 hidden units, compared to the 256 hidden units of $\phi^\star(\cdot\,; \boldsymbol{w}^\star)$.

We see in Figure 3.2 that SGD tends to outperform PLI overall, especially in high SNR regime. In the low SNR regime, PLI and SGD are mostly tied in their performance, exhibiting very similar test errors.

Path Planning as Structured Prediction. Among all monotonic paths from the top left corner to the bottom right corner of a grid, we consider finding the path that maximizes the rewards collected on each tile it passes through. Specifically, we consider images generated in the game Warcraft II (Guyomarch, 2017) as illustrated in Figure 3.3. Each tile corresponds to some terrain such as water, desert, grass, or rock with a fixed reward (grass $>$ desert $>$ water $>$ rock). As long as the rewards can directly be observed, this task can be solved by dynamic programming. In this experiment, the rewards are not directly observed; they are computed as the transformation of the raw pixels of each tile by a convolutional neural network. Our goal is to learn the reward function from a dataset of random maps with their associated optimal path.

Given a map $\boldsymbol{x}$ with associated best path $\boldsymbol{y}$, let denote by $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{y}'; \boldsymbol{w})$ the augmented score of a path

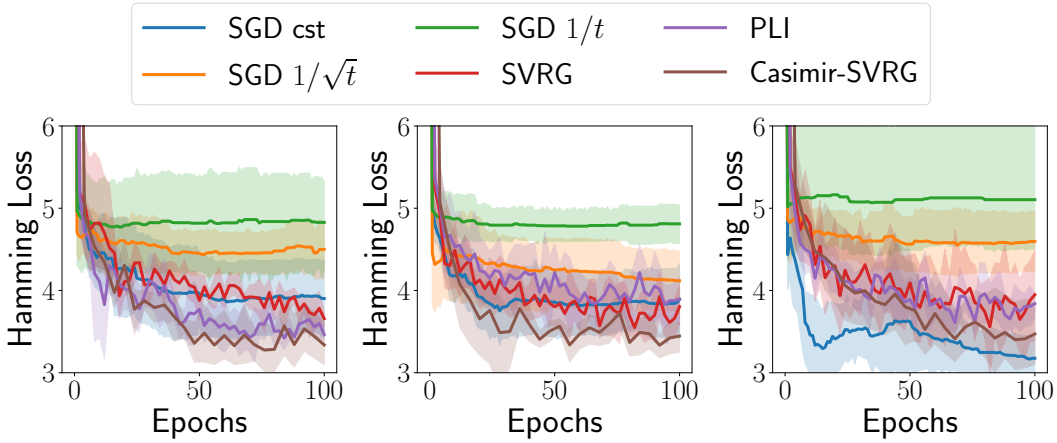


Figure 3.4: Planning as a structured prediction problem. We plot the Hamming loss, which measures how good the predicted path is to the actual shortest path on unseen grids. From left to right: $\mu = 1/n, 10^{-2}/n, 10^{-4}/n$

$\boldsymbol{y}'$ parameterized by $\boldsymbol{w}$; see Section 2.1 for a detailed description of structured prediction. Our objective is to solve the max-margin structured prediction formulation (2.7) of Chapter 2 with non-linear score functions. Concretely, the objective is

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max_{\boldsymbol{y}' \in \mathcal{Y}} \psi(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{y}'; \boldsymbol{w}) + \frac{\mu}{2} \|\boldsymbol{w}\|_2^2, \tag{3.25}$$

where $(\boldsymbol{x}_i)_{i=1}^n$ are the maps, $(\boldsymbol{y}_i)_{i=1}^n$ are their respective best paths, and $\mu \geq 0$ is some regularization parameter.

Recall from Section 2.3 that subgradients for (3.25) can be computed by estimating the highest reward path $\boldsymbol{y}'$ associated with a given sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for a feature map parameterized by the current parameters $\boldsymbol{w}$. We smooth the objective in (3.25) by the top-K oracle (cf. Definition 2.5). That is, we consider an inf-convolution of the max by a squared ℓ_2 norm, and approximated using the top-K shortest paths for the given score function.

The methods we consider are (i) stochastic subgradient methods (Davis and Drusvyatskiy, 2019), denoted SGD, with various learning rates strategies $\gamma_t = \gamma_0$, $\gamma_t = \gamma_0/\sqrt{t}$ and $\gamma_t = \gamma_0/t$, (ii) a variance reduced stochastic sub-gradient method, denoted SVRG (Johnson and Zhang, 2013), (iii) Casimir-SVRG, an accelerated algorithm on the Moreau-envelope of the objective as described in Chapter 2, (iv) a modified Gauss-Newton algorithm as described in (3.4), denoted PL. Both Casimir-SVRG and PL require to smooth the objective, i.e., using a top-K oracle, as we discussed in Chapter 2.

In Figure 3.4, we observe that SGD with constant step-size carefully tuned can perform as well as more sophisticated methods such as the modified Gauss-Newton method. Most importantly, for a small regularization parameter ($\mu = 10^{-4}/n$), SGD yields the best test Hamming loss, which in this task is the target metric.

Stochastic Nonlinear Equations. Gauss-Newton-type methods can be applied to stochastic non-linear

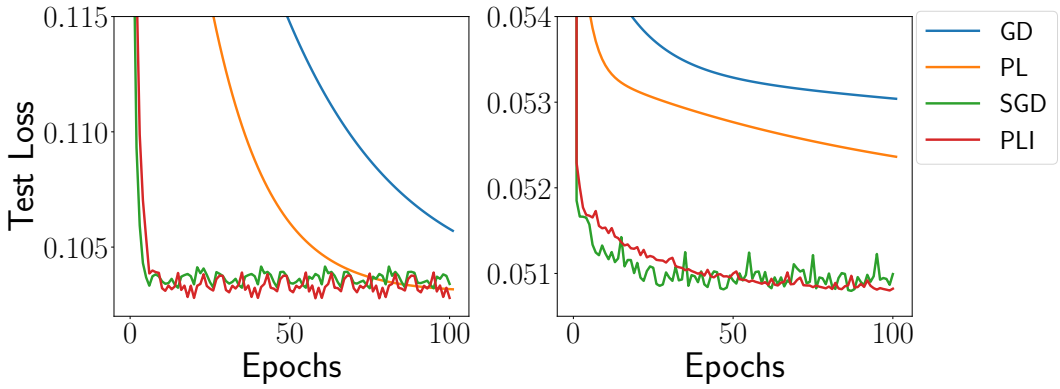


Figure 3.5: Solving stochastic nonlinear equations. We plot the "test loss", which is the objective value on a separate testing set. **Left**: ijccn1 dataset. **Right**: covtype dataset

equations of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \; h\left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_i(\boldsymbol{w})\right) \tag{3.26}$$

where h is a convex, possibly non-smooth, Lipschitz function such as $\|\cdot\|_1$ and the inner mappings are smooth and typically of the form $\boldsymbol{g}_i(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{w}) - \boldsymbol{y}_i$ (Tran-Dinh et al., 2020; Zhang and Xiao, 2020). Problem (3.26) can be interpreted as ensuring that, on average, the non-linear mapping $\boldsymbol{g}(\cdot, \boldsymbol{w})$ maps the inputs $\boldsymbol{x}_i$ to the targets $\boldsymbol{y}_i$.

Algorithms. Denoting $\boldsymbol{g}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_i(\boldsymbol{w})$, a natural baseline algorithm is to compute iterates as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \gamma \widehat{\nabla}\boldsymbol{g}(\boldsymbol{w})^\top \nabla h(\widehat{\boldsymbol{g}}(\boldsymbol{w}_k))), \tag{3.27}$$

where $\widehat{\nabla}\boldsymbol{g}(\boldsymbol{w})$ and $\widehat{\boldsymbol{g}}(\boldsymbol{w})$ are approximations of $\nabla\boldsymbol{g}(\boldsymbol{w})$ and $\boldsymbol{g}(\boldsymbol{w})$ respectively that can be approximated from a mini-batch (Wang et al., 2017); we call this "SGD". Note that the minibatch subgradient estimates can be biased since the outer function h can be non-linear. A modified Gauss-Newton or Prox-Linear method adapted to the inner finite-sum performs the iterations

$$\boldsymbol{w}_{k+1} = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} \left\{ h\left(\widehat{\boldsymbol{g}}(\boldsymbol{w}_k) + \widehat{\nabla}\boldsymbol{g}(\boldsymbol{w}_k)(\boldsymbol{w} - \boldsymbol{w}_k)\right) + \frac{M}{2}\|\boldsymbol{w} - \boldsymbol{w}_k\|_2^2 \right\} \tag{3.28}$$

where each sub-problem can be solved by incremental algorithms such as the accelerated dual proximal gradient ascent (Tran-Dinh et al., 2020; Zhang and Xiao, 2020).

Experiment. We consider the experimental setting of (Tran-Dinh et al., 2020). The objective is to solve (3.26) where h is the Huber loss, a smooth surrogate of the nonsmooth ℓ_1 norm, and inner mappings $\boldsymbol{g}$ are the concatenation of four different losses, i.e., $\boldsymbol{g}_i(\boldsymbol{w}) = (\ell_1(\boldsymbol{x}_i^\top \boldsymbol{w}, \boldsymbol{y}_i), \ldots, \ell_4(\boldsymbol{x}_i^\top \boldsymbol{w}, \boldsymbol{y}_i))$, where the formulations of the losses can be found in (Tran-Dinh et al., 2020). The samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are drawn from the datasets `ijcnn1` or `covtype` from the LIBSVM repository (Chang and Lin, 2011).

We consider (i) a gradient descent denoted GD, (ii) a modified Gauss-Newton or prox-linear method denoted PL, (iii) a baseline of the form (3.27), denoted SGD, (iv) an incremental Gauss-Newton or prox-linear method as described in (3.28), denoted PLI for consistency. In Figure 3.5, we observe that PL outperforms

GD as expected in the batch setting. However, this advantage is longer present in the incremental setting. The analogue of gradient descent, namely the SGD baseline (3.27), often performs better than its Gauss-Newton counterpart PLI.

Chapter 4

MAUVE: Measuring the Gap Between Neural Text and Human Text

As a consequence of the explosion of scale, large-scale text generation models show an ability to produce human-like text of remarkable quality and coherence in open-ended generation (Radford et al., 2019; Zellers et al., 2019; Brown et al., 2020). In this setting, a text generation model forms a distribution over natural language sequences, induced by an autoregressive neural sequence model (e.g., GPT-3 (Brown et al., 2020)) paired with a decoding algorithm (e.g., nucleus sampling (Holtzman et al., 2020)). While we cannot always compute the exact probability assigned to each sequence, we can easily sample text from this distribution. The goal of generative modeling is to obtain samples that resemble those from the "true" distribution of human-written text.

To evaluate how close a generation model's distribution is to that of human-written text, we must consider two types of errors: (I) where the model assigns high probability to sequences that do *not* resemble human-written text, and, (II) where the model distribution does not cover the human distribution, i.e., it fails to yield diverse samples. However, quantifying these aspects in a principled yet computationally tractable manner is challenging, as the text distributions are high-dimensional and discrete, accessed only through samples or expensive model evaluations (Holtzman et al., 2020; Zhang et al., 2021).

We develop MAUVE, a comparison measure for open-ended text generation. The proposed measure is efficient, interpretable, and practical for evaluating modern text generation models. It captures both types of errors (Figure 4.1) by building upon *information divergence frontiers* (Sajjadi et al., 2018; Kynkäänniemi et al., 2019; Djolonga et al., 2020), so far underexplored in natural language processing. The key idea for making the proposed measure computationally tractable, yet effective, is to reduce its measurement to computing Kullback-Leibler divergences in a quantized, low-dimensional space after embedding samples from each distribution with an external language model. From an end user's perspective, MAUVE has a simple interface: given neural text and human text, it yields a scalar measure of the gap between them.

We summarize the contributions of this chapter. First, we introduce MAUVE, a comparison measure between neural text and human text. Second, we empirically show that MAUVE is able to quantify known properties of generated text with respect to text length, model size, and decoding more correctly and with fewer restrictions than existing distributional evaluation metrics. Third, we find through a human evaluation that MAUVE better correlates with human quality judgments of text. Finally, we find that MAUVE can be highly robust to the choice of quantization, embeddings, and scaling. We open-source a pip-installable Python package to compute MAUVE.[1]

[1] Available from `https://github.com/krishnap25/mauve`.

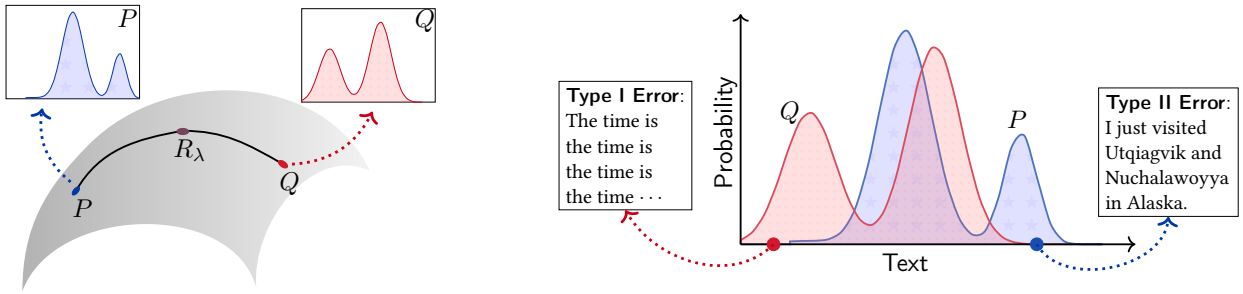


Figure 4.1: Left: MAUVE compares the machine text distribution Q to that of human text P by using the family of mixtures $R_\lambda = \lambda P + (1-\lambda)Q$ for $\lambda \in (0, 1)$. **Right**: Illustration of *Type I errors*, where Q produces degenerate, repetitive text which is unlikely under P, and, *Type II errors*, where Q cannot produce plausible human text due to truncation heuristics (Holtzman et al., 2020). MAUVE measures these errors softly, by using the mixture distribution R_λ. Varying λ in $(0, 1)$ gives a divergence curve and captures a spectrum of soft Type I and Type II errors. MAUVE summarizes the entire divergence curve in a single scalar as the area under this curve.

4.1 Background

We begin by discussing the basics of open-ended text generation. We start with neural autoregressive language models since these form the backbone of prevailing approaches to text generation.

Language Modeling. Consider sequences $\boldsymbol{x} = (x_1, \cdots, \boldsymbol{x}_{|\boldsymbol{x}|})$ of natural language text, where each x_i belongs to a finite vocabulary V (e.g., characters or words). As discussed in Chapter 1, a language model $\hat{P}(\cdot \mid \boldsymbol{x}_{1:t})$ models the conditional distribution over the next token x_{t+1} following the sequence $\boldsymbol{x}_{1:t}$. Neural language models are those language models parameterized by a neural network with parameters denoted by $\theta \in \mathbb{R}^d$, so we write it as $\hat{P}_\theta(\cdot \mid \boldsymbol{x}_{1:t})$ when we wish to make this dependence explicit. Contemporary neural language models are based on the transformer architecture (Vaswani et al., 2017), which is summarized in Figure 4.2 (left). We refer to Chapter 1 for a discussion on the history of language modeling.

The usual training objective for neural language modeling is via supervised multi-class classification of the next token. We assume that there is an underlying distribution $P(\cdot \mid \boldsymbol{x}_{1:t})$ for the next token x_{t+1} humans would write in continuation to a prefix $\boldsymbol{x}_{1:t}$. The training procedure aims to minimize the Kullback-Liebler (KL) divergence between the distributions $P(\cdot \mid \boldsymbol{x}_{1:t})$ and $\hat{P}(\cdot \mid \boldsymbol{x}_{1:t})$ assigned by humans and the language model respectively over the next token x_{t+1} in continuation to a context $\boldsymbol{x}_{1:t} \sim P_t$ coming from *human-written* text:

$$\min_\theta \; \mathbb{E}_{t \sim \text{Unif}([T-1])} \mathbb{E}_{\boldsymbol{x}_{1:t} \sim P_t} \left[\text{KL}\Big(P(\cdot \mid \boldsymbol{x}_{1:t}) \Big| \hat{P}_\theta(\cdot \mid \boldsymbol{x}_{1:t}) \Big) \right] , \tag{4.1}$$

where T is the maximum sequence length. Since neither the distribution P_t over prefixes of length t nor the distribution $P(\cdot \mid \boldsymbol{x}_{1:t})$ over the next token is known in practice, plug-in estimates of both are employed in practice.

Autoregressive models also yield an estimate of the joint probability $\hat{P}(\boldsymbol{x})$ of a sequence $\boldsymbol{x} = (x_1, \cdots, x_{|\boldsymbol{x}|})$ as

$$\hat{P}(\boldsymbol{x}) = \prod_{t=0}^{|\boldsymbol{x}|-1} \hat{P}(x_{t+1} \mid \boldsymbol{x}_{1:t}) \, .$$

Note that this parameterization is in contrast to those of Chapter 2, where we limited the length of the context akin to Hidden Markov Models (HMMs) for tractable inference.

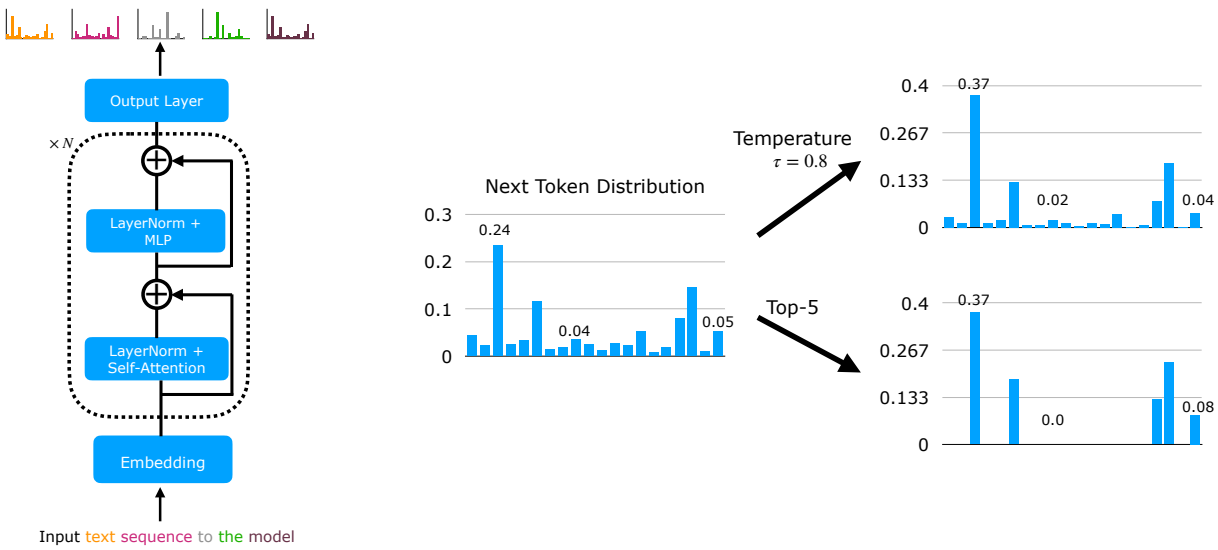


Figure 4.2: Left: Transformer architecture. **Right**: Illustration of how decoding algorithms reshape the next-token distribution.

Open-Ended Text Generation. The open-ended text generation task asks us to output text $\hat{\boldsymbol{x}}_{s+1:|\hat{\boldsymbol{x}}|}$ in continuation of a given context $\boldsymbol{x}_{1:s}$. In contrast to directed text generation tasks such as translation, summarization, and question-answering, the task here is open-ended in that the context size $s \ll |\hat{\boldsymbol{x}}|$ is typically small and does not really constrain the output space. Unlike directed text generation tasks such as translation, summarization, and question-answering, the goal here is to generate text that is coherent, fluent, creative, and engaging. Since these criteria are hard to make mathematically precise, we instead consider the surrogate goal of generating text which is *human-like*, such that generated text samples can pass for samples from the distribution P over human written text sequences.

We model a text generation system as a probability distribution $Q(\cdot \mid \boldsymbol{x}_{1:s})$ such that its generated text $\hat{\boldsymbol{x}}_{s+1:|\hat{\boldsymbol{x}}|}$ is an i.i.d. sample from Q. Given a neural autoregressive language model $\hat{P}$, we can generate open-ended text in a serial, left-to-right fashion, by sampling $\hat{x}_{s+1} \sim \hat{P}(\cdot|\boldsymbol{x}_{1:s})$, $\hat{x}_{s+2} \sim \hat{P}(\cdot|\boldsymbol{x}_{1:s}, \hat{x}_{s+1})$, etc. This is also known as *ancestral sampling*, and the induced distribution Q over sequences is

$$Q_{\text{samp}}(\boldsymbol{x}_{1:s}, \hat{\boldsymbol{x}}_{s+1:|\hat{\boldsymbol{x}}|}) = \prod_{t=1}^{s} P(x_t|\boldsymbol{x}_{1:t-1}) \prod_{t=s+1}^{|\hat{\boldsymbol{x}}|} \hat{P}(\hat{x}_t|\boldsymbol{x}_{1:s}, \hat{\boldsymbol{x}}_{s+1:t-1}) \,,$$

where we assume that the prefix $\boldsymbol{x}_{1:s} \sim P_s$ is drawn from the human distribution.

Decoding Algorithms. Assuming the language model learning has succeeded, we have that $\hat{P}(\cdot \mid \boldsymbol{x}_{1:t}) \approx P(\cdot \mid \boldsymbol{x}_{1:t})$ for prefixes $\boldsymbol{x}_{1:t} \sim P_t$ drawn from the distribution of human-written text, in the sense that the objective of (4.1) is bounded above by some $\varepsilon > 0$. However, for $\hat{\boldsymbol{x}}_{1:t}$ drawn from a distribution Q_t which is different from the human distribution P_t, the model's next-token distribution $\hat{P}(\cdot \mid \hat{\boldsymbol{x}}_{1:t})$ can be quite different from $P(\cdot \mid \hat{\boldsymbol{x}}_{1:t})$ of humans. In the iterative process of ancestral sampling, the gap between $P(\hat{\boldsymbol{x}}_{1:t})$ and $Q_{\text{samp}}(\hat{\boldsymbol{x}}_{1:t})$ keep increasing as the generation length t grows larger, so that Q_{samp} is quite far from P. This leads to *decoding algorithms* which produce samples

$$\hat{x}_{t+1} \sim Q(\cdot \mid \boldsymbol{x}_{1:s}, \hat{\boldsymbol{x}}_{s+1:t}) \,,$$

where $Q(\cdot \mid \boldsymbol{x}_{1:t})$ is a reshaping of the language model $\hat{P}(\cdot \mid \boldsymbol{x}_{1:t})$ in order to promote more conservative outputs. We now define a few popular decoding algorithms; see also Figure 4.2 (right) for an illustration.

Temperature rescaling (Ackley et al., 1985) applies to language models parameterized with a softmax function:

$$\hat{P}(x_{t+1} \mid \boldsymbol{x}_{1:t}) = \frac{\exp\big(\phi(x_{t+1}|\boldsymbol{x}_{1:t})\big)}{\sum_{x \in V} \exp\big(\phi(x|\boldsymbol{x}_{1:t})\big)}\,,$$

for some unnormalized scoring function $\phi(\cdot \mid \boldsymbol{x}_{1:t}) : V \to \mathbb{R}$. This decoding algorithm rescales the term inside the exponential with a "temperature" parameter $\tau > 0$:

$$Q_{\text{temp},\tau}(x_{t+1} \mid \boldsymbol{x}_{1:t}) = \frac{\exp\big(\frac{1}{\tau}\phi(x_{t+1}|\boldsymbol{x}_{1:t})\big)}{\sum_{x'_{t+1} \in V} \exp\big(\frac{1}{\tau}\phi(x'_{t+1}|\boldsymbol{x}_{1:t})\big)}\,.$$

When $\tau < 1$, the distribution $Q_{\text{temp},\tau}(\cdot \mid \boldsymbol{x}_{1:t})$ becomes more peaked around the most likely next tokens, making the distribution more conservative.

For an integer $K < |V|$, *top-K sampling* (Fan et al., 2018) applies the transformation

$$Q_{\text{top-}K}(x_{t+1}|\boldsymbol{x}_{1:t}) = \begin{cases} \frac{1}{Z}\,\hat{P}(x_{t+1}|\boldsymbol{x}_{1:t})\,, & \text{if } x_{t+1} \in V_{\text{top-}K}, \\ 0\,, & \text{else,} \end{cases}$$

where Z is a normalizing constant, and $V_{\text{top-}K} = \{z_{(1)}, \cdots, z_{(K)}\} \subset V$ is the set of the K highest scoring tokens satisfying

$$\hat{P}(z_{(1)}|\boldsymbol{x}_{1:t}) \geq \cdots \geq \hat{P}(z_{(K)}|\boldsymbol{x}_{1:t}) \geq \max_{z \in V \setminus V_{\text{top-}K}} \hat{P}(z|\boldsymbol{x}_{1:t})\,.$$

The extreme $K = |V|$ corresponds to ancestral sampling. The other extreme $K = 1$ is known as *greedy decoding*, which corresponds to choosing the most likely next token iteratively. Greedy decoding is often used to approximate the most likely sequence $\arg\max_{\boldsymbol{x}} P(\boldsymbol{x}|\boldsymbol{x}_{1:t})$. This is exactly the inference problem of Chapter 2, which cannot be efficiently solved with an exact algorithm in this case.

Nucleus sampling (Holtzman et al., 2020), similar to top-K sampling, returns a sparse distribution. Given a parameter $p \in (0, 1)$, it applies the transformation

$$Q_{\text{nuc},p}(x_{t+1} \mid \boldsymbol{x}_{1:t}) = \begin{cases} \frac{1}{Z}\,\hat{P}_{\text{nuc},p}(x_{t+1} \mid \boldsymbol{x}_{1:t}), & \text{if } x_{t+1} \in V_{\text{nuc},p}, \\ 0, & \text{else,} \end{cases}$$

where the top-p vocabulary $V_{\text{nuc},p}$ is defined as the smallest set $V' \subset V$ such that $\sum_{x \in V'} \hat{P}(x|\boldsymbol{x}_{1:t}) \geq p$, and Z is the normalizing constant

$$Z = \sum_{x \in V_{\text{nuc},p}} \hat{P}(x|\boldsymbol{x}_{1:t})\,.$$

4.2 MAUVE

In this section, we introduce MAUVE for measuring the divergence between machine-generated text and human text.

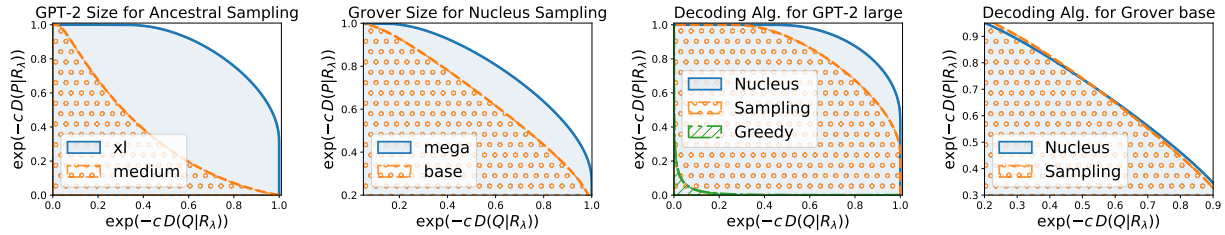


Figure 4.3: Divergence curves for different models (GPT-2 (Radford et al., 2019), Grover (Zellers et al., 2019)) and decoding algorithms (greedy decoding, ancestral and nucleus sampling). MAUVE is computed as the area of the shaded region, and larger values of MAUVE indicate that Q is closer to P. In general, MAUVE indicates that generations from larger models and nucleus sampling are closer to human text. **Rightmost**: Nucleus sampling has a slightly smaller Type I error than ancestral sampling but a higher Type II error, indicating that ancestral sampling with Grover base produces more degenerate text while nucleus sampling does not effectively cover the human text distribution.

4.2.1 Definition of MAUVE

Sources of Error in Text Generation. Our goal is to measure the gap between the model distribution Q and the target distribution P of human text. As highlighted in Figure 4.1, this gap arises from two sources of error:

(Type I) Q places high mass on some text which is unlikely under P,

(Type II) Q cannot generate text which is plausible under P.

The Type I errors are false positives, including the common failure case where a model generates text with semantic repetitions (Dinan et al., 2019; Holtzman et al., 2020; Welleck et al., 2020b) that are highly unlikely to be written by humans.[2] The Type II errors are false negatives, which can occur, for instance, because some pieces of plausible human text cannot be generated by truncation-based decoding algorithms such as nucleus sampling (Holtzman et al., 2020). The gap between P and Q is small only if both of these errors are small.

Quantifying the Errors. We formalize the Type I and II errors with the Kullback-Leibler (KL) divergences $\mathrm{KL}(Q|P)$ and $\mathrm{KL}(P|Q)$, respectively (see e.g. Han and Kobayashi, 2007, for a background). The divergence $\mathrm{KL}(Q|P)$ penalizes Q if there exists text $\boldsymbol{x}$ such that $Q(\boldsymbol{x})$ is large but $P(\boldsymbol{x})$ is small, so it quantifies the Type I error. Likewise, $\mathrm{KL}(P|Q)$ quantifies the Type II error.

Unfortunately, one or both of the KL divergences $\mathrm{KL}(P|Q)$ and $\mathrm{KL}(Q|P)$ are infinite if the supports of P and Q are not identical, which is often the case in open-ended generation. This makes the KL divergence itself unsuitable as an evaluation metric. We overcome this issue by *softly* measuring the two errors using the mixture distribution $R_\lambda = \lambda P + (1-\lambda)Q$ for some $\lambda \in (0,1)$. In particular, we define the (soft) Type I error at level λ as $\mathrm{KL}(Q|R_\lambda)$ and the (soft) Type II error as $\mathrm{KL}(P|R_\lambda)$.

Summarizing the Errors with a Divergence Curve. Since the mixture weight λ was arbitrary, we consider a family of Type I and II error values by varying λ between 0 and 1, in the same spirit as information

[2] Let text $\boldsymbol{x}$ with $P(\boldsymbol{x}) \gg 0$ be the positive class and $P(\boldsymbol{x}) \approx 0$ be the negative class. If $Q(\boldsymbol{x}) \gg 0$ for some negative $\boldsymbol{x}$, then the model incorrectly considers it a positive, so it is a *false* positive.

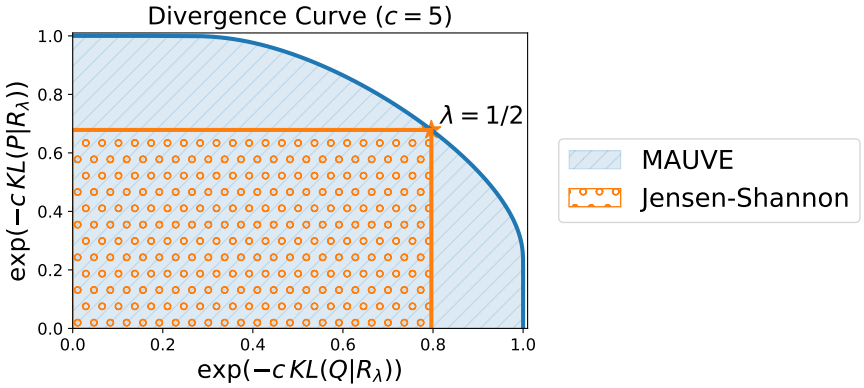


Figure 4.4: MAUVE versus the Jensen-Shannon divergence. The Jensen-Shannon divergence can be calculated from the area A of the orange region by Eq. (4.3). MAUVE, on the other hand, is the area under the entirety of the divergence curve.

divergence frontiers (Sajjadi et al., 2018; Djolonga et al., 2020). This yields a *divergence curve*,

$$\mathcal{C}(P,Q) = \left\{ \left(\exp(-c\,\mathrm{KL}(Q|R_\lambda)), \exp(-c\,\mathrm{KL}(P|R_\lambda)) \right) \,:\, R_\lambda = \lambda P + (1-\lambda)Q,\ \lambda \in (0,1) \right\}, \quad (4.2)$$

where $c > 0$ is a hyperparameter for scaling. The divergence curve formalizes and encodes information about the trade-off between Type I and II errors. Figure 4.3 illustrates the divergence curves for different models and decoding algorithms.

Our proposed measure, **MAUVE**(P,Q), is the area under the divergence curve $\mathcal{C}(P,Q)$. It provides a scalar summary of the trade-off between Type I and Type II errors. MAUVE(P,Q) lies in $(0,1]$, with a larger value meaning that Q is closer to P. Further, MAUVE$(P,Q) = 1$ if and only if $Q = P$. The area under the curve is a common summary of trade-off curves in machine learning (Cortes and Mohri, 2005; Clémençon and Vayatis, 2009; Clémençon and Vayatis, 2010; Flach, 2012).

4.2.2 Properties of MAUVE

Connections to Common Divergences. The divergence curve encodes more information than the KL divergence $\mathrm{KL}(P|Q)$, which can be obtained from the second coordinate of the curve $\mathcal{C}(P,Q)$ as $\lambda \to 0$, and the reverse KL divergence $\mathrm{KL}(Q|P)$ which can be obtained from the first coordinate of the curve $\mathcal{C}(P,Q)$ as $\lambda \to 1$. Further, the Jensen-Shannon (JS) divergence $\mathrm{JS}(P,Q) = \big(\mathrm{KL}(P|R_{1/2}) + \mathrm{KL}(Q|R_{1/2})\big)/2$, can be obtained from the two coordinates of $\mathcal{C}(P,Q)$ at $\lambda = 1/2$. Indeed, the JS divergence can be obtained as the area A of the axis-parallel rectangle between the point on the divergence curve corresponding to $\lambda = 1/2$ and the origin; cf. Figure 4.4. In particular, we have,

$$\mathrm{JS}(P,Q) = \frac{1}{2}\big(\mathrm{KL}(P|R_{1/2}) + \mathrm{KL}(Q|R_{1/2})\big) = \frac{1}{2c}\log\frac{1}{A}\,. \quad (4.3)$$

In contrast to these divergences, MAUVE summarizes *all* of the divergence curve $\mathcal{C}(P,Q)$.

Pareto Optimality of Divergence Curves. We now show that the divergence curve $\mathcal{C}(P,Q)$ encodes the Pareto frontier of $\big(\mathrm{KL}(P|R), \mathrm{KL}(Q|R)\big)$ for all distributions R, not just mixtures of the form R_λ.

Proposition 4.1. *Consider two distributions P, Q with finite support and a scaling constant $c > 0$. Let R_λ be such that $\big(e^{-c\,\mathrm{KL}(Q|R_\lambda)}, e^{-c\,\mathrm{KL}(P|R_\lambda)}\big) \in \mathcal{C}(P, Q)$. Then, R_λ is Pareto-optimal for the pair of objectives $\big(\mathrm{KL}(Q|\cdot), \mathrm{KL}(P|\cdot)\big)$. In other words, there does* not *exist any distribution R such that $\mathrm{KL}(Q|R) < \mathrm{KL}(Q|R_\lambda)$ and $\mathrm{KL}(P|R) < \mathrm{KL}(P|R_\lambda)$ simultaneously.*

Proof. Let $\mathcal{F}(P, Q)$ be the Pareto frontier of $\big(\mathrm{KL}(Q|\cdot), \mathrm{KL}(P|\cdot)\big)$. The convexity of $\mathrm{KL}(Q|\cdot), \mathrm{KL}(P|\cdot)$ allows us to compute the Pareto frontier $\mathcal{F}(P, Q)$ exactly by minimizing linear combinations of the objectives. Concretely, we have from (Miettinen, 2012, Theorems 3.4.5, 3.5.4) that

$$\mathcal{F}(P, Q) = \Big\{ \big(\mathrm{KL}(P|R^\star_\lambda), \mathrm{KL}(P|R^\star_\lambda)\big) \;:\; \lambda \in [0, 1] \Big\}, \;\text{ where, }\; R^\star_\lambda \in \arg\min_R \{\lambda\,\mathrm{KL}(Q|R) + \bar\lambda\,\mathrm{KL}(P|R)\}\,,$$

and $\bar\lambda = 1 - \lambda$. By adding and subtracting $\sum_i R_{\lambda,i} \log(R_{\lambda,i})$, we get the identity

$$\lambda\,\mathrm{KL}(P|R) + \bar\lambda\,\mathrm{KL}(Q|R) = \lambda\,\mathrm{KL}(P|R_\lambda) + \bar\lambda\,\mathrm{KL}(Q|R_\lambda) + \mathrm{KL}(R_\lambda|R)\,.$$

The first two terms of the right-hand side are independent of R and the last term is minimized at $R = R_\lambda$. Therefore, $R^\star_\lambda = R_\lambda$. □

Proposition 4.1 reveals the connection between MAUVE and divergence frontiers (Djolonga et al., 2020). In particular, the inclusive KL divergence frontier, defined as the Pareto frontier $\mathcal{F}(P, Q)$ of $\big(\mathrm{KL}(Q|\cdot), \mathrm{KL}(P|\cdot)\big)$ (see the proof of Proposition 4.1), is connected to the divergence curve $\mathcal{C}(P, Q)$ as

$$\mathcal{F}(P, Q) = \Big\{ \big(c^{-1} \log t_1^{-1}, c^{-1} \log t_2^{-1}\big) \;:\; (t_1, t_2) \in \mathcal{C}(P, Q) \Big\}\,.$$

Statistical Estimation. In practice, we do not have access to the population distribution P over human text or the complete distribution Q over machine text. These quantities must be estimated from i.i.d. samples each from P and Q, and this incurs a statistical estimation error. We bound here the statistical estimation error of a close relative of MAUVE, namely the Jensen-Shannon divergence (cf. Figure 4.4).

Theorem 4.2. *Consider two discrete distributions $P, Q \in \Delta^{k-1}$, and i.i.d. samples $Y_1, \cdots, Y_n \sim P$ and $Y_1, \cdots, Y'_n \sim Q$. Denote $\hat P_n = (1/n) \sum_{i=1}^n \delta_{Y_i}$ and $\hat Q_n = (1/n) \sum_{i=1}^n \delta_{Y'_i}$ denote the corresponding empirical distributions. We have,*

$$\mathbb{E}\,\Big|\mathrm{JS}(\hat P_n, \hat Q_n) - \mathrm{JS}(P, Q)\Big| \le \frac{\log n}{2}\big(\alpha_n(P) + \alpha_n(Q)\big) + \frac{1}{2}\big(\beta_n(P) + \beta_n(Q)\big)\,,$$

where $\alpha_n(P) = \sum_{i=1}^k \sqrt{n^{-1} P_i}$ and $\beta_n(P) = \mathbb{E}\big[\sum_{i:\hat P_{n,i}=0} P_i \log 1/P_i\big]$. We also have the distribution-free bound

$$\mathbb{E}\,\Big|\mathrm{JS}(\hat P_n, \hat Q_n) - \mathrm{JS}(P, Q)\Big| \le \frac{\log n}{2}\left(\sqrt{\frac{k}{n}} + \frac{k}{n}\right)\,.$$

The bound captures a parametric rate of convergence, i.e., $O(n^{-1/2})$, up to a logarithmic factor. In fact, this rate is not improvable in a related problem of estimating $\mathrm{KL}(P|Q)$, even with the assumption that P/Q is bounded (Bu et al., 2018).

A key component of this result is the missing mass problem, first studied by Good (1953), in the context of estimating the probability of a new observation drawn from a fixed distribution that has not previously appeared, in other words, is missing in the current sample. Their Good-Turing estimator has been widely

used in language modeling (e.g., Katz, 1987) and studied in theory (McAllester and Schapire, 2000; Orlitsky et al., 2003; Orlitsky and Suresh, 2015).

This proof technique is fairly general and can be applied for a wide range of f-divergences, cf. (Liu et al., 2021a). We only sketch the proof here and give the full proof in Appendix B.

Proof Sketch of Theorem 4.2. The proof relies on a careful analysis of the derivatives of the Jensen-Shannon divergence while accounting for the missing mass. Define the bivariate scalar function

$$\psi(p,q) = \frac{p}{2}\log\left(\frac{2p}{p+q}\right) + \frac{q}{2}\log\left(\frac{2q}{p+q}\right),$$

so that $\mathrm{JS}(P,Q) = \sum_{i=1}^{k}\psi(P_i,Q_i)$. By the triangle inequality, we have,

$$\left|\mathrm{JS}(\hat{P}_n,\hat{Q}_n) - \mathrm{JS}(P,Q)\right| \le \sum_{i=1}^{k} \underbrace{\left|\psi(\hat{P}_{n,i},\hat{Q}_{n,i}) - \psi(P_i,\hat{Q}_{n,i})\right|}_{=:\Delta_i} + \underbrace{\left|\psi(\hat{P}_i,\hat{Q}_{n,i}) - \psi(P_i,Q_i)\right|}_{=:\Delta_i'}.$$

We bound Δ_i in terms of $|\hat{P}_{n,i} - P_i|$ so that summing over all coordinates gives a bound on the total variation distance $\|\hat{P}_n - P\|_{\mathrm{TV}} = \sum_{i=1}^{k}|\hat{P}_{n,i} - P_i|$. A first order Taylor expansion gives the bound

$$\Delta_i \le \max_{\lambda\in[0,1]} |\psi_p(\lambda P_i + (1-\lambda)\hat{P}_{n,i}, Q_i)|\ |P_i - \hat{P}_{n,i}|,$$

where ψ_p denotes the partial derivative of ψ w.r.t. its first argument. Unfortunately, as $p \to 0$ for fixed $q \neq 0$, we have that $|\psi_p(p,q)| \le (1/2)\log(q/p) \to \infty$.

We use a two-pronged approach to overcome this issue. First, we take a second order Taylor expansion and carefully bound the second order remainder term to get

$$\Delta_i \le \frac{1}{2}\log\frac{1}{\max\{P_i,\hat{P}_{n,i}\}}\,|\hat{P}_{n,i} - P_i|.$$

Secondly, because $\hat{P}_n$ is an empirical measure, we can only have two possibilities: $\hat{P}_{n,i} \ge 1/n$ or $\hat{P}_{n,i} = 0$. The first case gives an additional $\log n$ dependence on the total variation distance, while the second case is the missing mass. Based on results from the missing mass literature (Berend and Kontorovich, 2012; Mcallester and Ortiz, 2003), we show that

$$\beta_n(P) = \mathbb{E}\left[\sum_{i=1}^{k}\mathbb{I}(\hat{P}_{n,i}=0)\ P_i\log\frac{1}{P_i}\right] \le \frac{k\log n}{n}.$$

Finally, we bound the total variation term with repeated applications of Jensen's inequality as

$$\mathbb{E}\|\hat{P}_n - P\|_{\mathrm{TV}} \le \sum_{i=1}^{k}\sqrt{\mathbb{E}(\hat{P}_{n,i}-P_i)^2} = \sum_{i=1}^{k}\sqrt{\frac{P_i(1-P_i)}{n}} \le \alpha_n(P) \le \sqrt{\frac{k}{n}}.$$

□

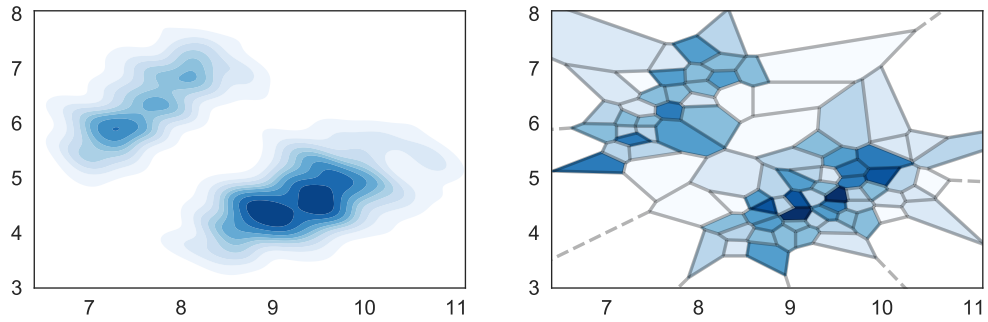


Figure 4.5: Illustration of the quantization. **Left**: A continuous two-dimensional distribution P. **Right**: A partitioning of the Euclidean plane $\mathbb{R}^2$ and the corresponding quantized distribution $\tilde{P}$.

4.2.3 Computing MAUVE for Open-Ended Text Generation

Each point on the divergence curve $\mathcal{C}(P, Q)$ consists of a coordinate

$$\mathrm{KL}(P|R_\lambda) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{R_\lambda(\boldsymbol{x})}, \tag{4.4}$$

and a similarly defined coordinate $\mathrm{KL}(Q|R_\lambda)$. We cannot compute the summation as written in Eq. (4.4), as we do not know the ground-truth probabilities $P(\cdot)$ and the support of a typical model distribution is prohibitively large since it is the space of all sequences of tokens. As a result of these two issues, MAUVE cannot be tractably computed in closed form.

We estimate MAUVE from samples $\boldsymbol{x}_i \sim P$ and $\boldsymbol{x}'_i \sim Q$ to overcome the fact that ground-truth probabilities $P(\cdot)$ are unknown. We circumvent the intractable support size by computing MAUVE in a quantized embedding space that is sensitive to important features of text.

The overall estimation procedure is as follows. First, we sample human text $\boldsymbol{x}_i \sim P$ and machine text $\boldsymbol{x}'_i \sim Q$. We then embed each text sequence using an external language model M (e.g., GPT-2 (Radford et al., 2019)) to obtain embeddings $\{M(\boldsymbol{x}_i)\}_{i=1}^N$ and $\{M(\boldsymbol{x}'_i)\}_{i=1}^{N'}$. Each embedding is now a vector $M(\boldsymbol{x}) \in \mathbb{R}^d$. Next, we jointly quantize the embedded samples (e.g. with k-means clustering (Manning and Schütze, 2001)) and count the cluster assignments to form histograms, giving low-dimensional discrete distributions that approximate each high-dimensional text distribution. In particular, the distribution P of human text is approximated by the discrete distribution $\tilde{P}$ of support size k, which is defined as,

$$\tilde{P}(j) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big(\phi(\boldsymbol{x}_i) = j\big)\,, \tag{4.5}$$

where $\phi(\boldsymbol{x}) \in \{1, \cdots, k\}$ returns the cluster id of $\boldsymbol{x}$. The model distribution Q is approximated as $\tilde{Q}$ similarly. Here, $\tilde{P}$ and $\tilde{Q}$ can be interpreted as piecewise constant approximations of P and Q, similar to a histogram; see Figure 4.5 for an illustration. Computing the divergence curve is now tractable, as each coordinate is a KL divergence between the k-element discrete distributions.

The full divergence curve (4.2) is a continuously parameterized curve for $\lambda \in (0, 1)$. For computational tractability, we take a discretization Λ of $[0, 1]$:

$$\hat{\mathcal{C}}(P, Q) = \left\{ \big(\exp(-c\,\mathrm{KL}(Q|R_\lambda)), \exp(-c\,\mathrm{KL}(P|R_\lambda))\big) \;:\; \begin{matrix} R_\lambda = \lambda P + (1-\lambda)Q, \\ \lambda \in \Lambda \end{matrix} \right\}. \tag{4.6}$$

Algorithm 4.1. Pseudocode to compute MAUVE

Input: Human text $\{\boldsymbol{x}_i^P\}_{i=1}^N$, model text $\{\boldsymbol{x}_i^Q\}_{i=1}^{N'}$, number of clusters k, embedding model M, discretization Λ of $[0, 1]$.

1: $\{M(\boldsymbol{x}_i^P)\}_{i=1}^N, \{M(\boldsymbol{x}_i^Q)\}_{i=1}^{N'} \leftarrow \texttt{embed}\left(M, \{\boldsymbol{x}_i^P\}_{i=1}^N, \{\boldsymbol{x}_i^Q\}_{i=1}^{N'}\right)$ ▷ Embed the samples

2: $C_P, C_Q = \texttt{quantize}\left(\{M(\boldsymbol{x}_i^P)\}_{i=1}^N, \{M(\boldsymbol{x}_i^Q)\}_{i=1}^{N'}\right)$ ▷ Cluster embeddings jointly

3: $\tilde{P} \leftarrow \texttt{count}(C_P)/N,\ \tilde{Q} \leftarrow \texttt{count}(C_Q)/N'$ ▷ Count cluster assignment

4: Compute $\hat{\mathcal{C}}(\tilde{P}, \tilde{Q})$ from (4.6) for $\lambda \in \Lambda$ ▷ Build the divergence curve

5: **return** MAUVE$(P, Q) = \texttt{area}\left(\hat{\mathcal{C}}(\tilde{P}, \tilde{Q})\right)$ ▷ Compute MAUVE using numerical quadrature

We take a uniform grid $\Lambda = \{1/n, 2/n, \cdots, (n-1)/n\}$ with n points. Finally, we estimate MAUVE as the area under $\hat{\mathcal{C}}(\tilde{P}, \tilde{Q})$ using numerical quadrature. We summarize the overall procedure in Algorithm 4.1.

To recap, our proposed measure **MAUVE**(P, Q) is the area under the divergence curve $\mathcal{C}(P, Q)$, providing a summary of all Type I and Type II errors through an efficient approximation designed for text generation. Next, we discuss how MAUVE compares to prior comparison measures for text (Section 4.3), then present empirical results with MAUVE (Section 4.4).

4.3 Related Work

Divergence Measures for Text. Prior measures of similarity/divergence between machine text and human text come in three broad categories: (a) reference-based, (b) statistics-based, and (c) language modeling. Table 4.1 summarizes the latter two categories, and contrasts them with MAUVE.

Reference-based measures evaluate generated text with respect to a (small set of) reference text sample(s), rather than comparing full sequence distributions. These include classical metrics for n-gram matching (Papineni et al., 2002; Lin, 2004; Banerjee and Lavie, 2005), which are designed to capture similarities in the surface form of the generated text and the human references, making them fundamentally ill-suited for open-ended generation. Moreover, it has been recently shown in (Novikova et al., 2017) that these classical metrics only weakly agree with human judgments.

More recent reference-based metrics are capable of comparisons in a high dimensional space (Shimanaka et al., 2018; Zhang et al., 2020; Sellam et al., 2020; Clark et al., 2019), thereby capturing distributional semantics beyond superficial n-gram statistics. For instance, Moverscore (Zhao et al., 2019) relies on the Word Mover's distance (Kusner et al., 2015), and is an instance of an optimal transportation distance (Villani, 2021). It computes the minimum cost of transforming the generated text to the reference text, taking into account the Euclidean distance between vector representations of n-grams, as well as their document frequencies. The paradigm of reference-based measures is useful for targeted generation tasks such as translation and summarization where matching a set of references is paramount. It is, however, unsuitable for open-ended generation where there typically are several plausible continuations for each context and creative generations are desirable.

Statistics-based measures compare the model distribution Q with respect to the human distribution P on the basis of some statistic $T(P)$ and $T(Q)$. Property-specific statistics such as the amount of repetition (Holtzman et al., 2020; Welleck et al., 2020b), verifiability (Massarelli et al., 2019), or termination (Welleck et al., 2020a) are orthogonal to MAUVE, which provides a summary of the overall gap between P and Q rather than focusing on an individual property. Another statistic is the generation perplexity (Fan et al.,

Type	Metric	Measures	Approximates
Statistics	Zipf Coefficient (Holtzman et al., 2020)	Unigram rank-frequency statistics	–
	Self-BLEU (Zhu et al., 2018)	N-gram diversity	–
	Generation Perplexity (Fan et al., 2018)	Generation quality via external model R	$\lvert\mathbb{E}_Q[\log R(\boldsymbol{x})] - \mathbb{E}_P[\log R(\boldsymbol{x})]\rvert$ (a single point inside $\mathcal{C}(P, Q)$)
Language Modeling	Perplexity	Test-set perplexity	$\mathbb{E}_P[\log Q(\boldsymbol{x})]$
	ε-perplexity (Martins et al., 2020)	Perplexity w/ Laplace smoothing	$\mathbb{E}_P[\tilde{Q}(\boldsymbol{x})]$
	Sparsemax Score (Martins et al., 2020)	LM quality (sparsemax loss (Martins and Astudillo, 2016))	$\mathbb{E}_P[\tilde{Q}(\boldsymbol{x})]$
	Token JS-Div. (Martins et al., 2020)	LM quality (JS divergence)	$\mathbb{E}_P[\tilde{Q}(\boldsymbol{x})]$
Divergence Curve	MAUVE (this work)	Quality & diversity via the divergence curve	$\mathcal{C}(P, Q)$ at all λ

Table 4.1: Summary of automatic distributional metrics for evaluating open-ended text generation. MAUVE provides a summary of all points along the divergence curve, rather than a single point. The summary is based on comparisons in a joint embedding space, rather than a statistic computed independently on each distribution. $\tilde{Q}$ informally refers to a quantity related to Q.

2018; Holtzman et al., 2020), which compares the perplexity of the model text $\boldsymbol{x} \sim Q$ with that of human text $\boldsymbol{x}' \sim P$ under an external model R. By virtue of $T(\cdot)$ being a scalar, generation perplexity cannot trade-off the Type I and Type II errors like MAUVE. In fact, the generation perplexity can be derived from *a single point* enclosed between the divergence curve and the axes.

Language modeling metrics calculate how (un)likely human text $\boldsymbol{x} \sim P$ is under the model distribution Q, for instance, using the probability $Q(\boldsymbol{x})$. These metrics are related to a single point on the divergence curve, rather than a full summary. Examples include the perplexity of the test set (which is a sample from P) under the model Q and its generalizations to handle sparse distributions (Martins et al., 2020). Unlike MAUVE, these metrics never see model text samples $\boldsymbol{x}' \sim Q$, so they cannot account for how likely the model text is under the human distribution P. Moreover, they cannot be used for decoding algorithms such as beam search which do not define a token-level distribution.

Automatic metrics have been proposed for specific domains such as generation of dialogues (Tao et al., 2018), stories (Guan and Huang, 2020), and others (Opitz and Frank, 2021). They capture task-specific properties; see the surveys (Celikyilmaz et al., 2020; Sai et al., 2020). In contrast, MAUVE compares machine and human text in a domain-agnostic manner. Other related work has proposed metrics that rely on multiple samples for quality-diversity evaluation (Caccia et al., 2020), and Bayesian approaches to compare the distribution of statistics in machine translation (Eikema and Aziz, 2020).

Non-automatic Metrics. HUSE (Hashimoto et al., 2019) aims to combine human judgments of Type I errors with Type II errors measured using perplexity under Q. Due to the costs of human evaluation, we consider HUSE, as well other metrics requiring human evaluation, such as single-pair evaluation, as complementary to MAUVE, which is an automatic comparison measure. As a separate technical caveat, it is unclear how to use HUSE for sparse Q that assigns zero probability to a subset of text, which is the case

Task Domain	Model	Finetuning	Dataset	Prompt Length	Max. Generation Length	Number of Generations
Web text	GPT-2 (all sizes)	Pretrained	Webtext	35 tokens	1024 tokens	5000
News	Grover (all sizes)	Pretrained	RealNews	varying	1024 tokens	5000
Stories	GPT-2 medium	Finetuned	WritingPrompts	50 tokens	512 tokens	5000

Table 4.2: Dataset and task summary. Note that 1024 tokens correspond to $\sim$ 750 words on average.

with state-of-the-art decoding algorithms (Holtzman et al., 2020; Martins et al., 2020).

Evaluation of Generative Models. Evaluation of generative models is an active area of research in computer vision, where generative adversarial networks (Goodfellow et al., 2014) are commonly used. However, metrics such as Inception Score (Salimans et al., 2016) are based on large-scale supervised classification tasks, and thus inappropriate for text generation. The Fréchet Distance (Heusel et al., 2017; Semeniuta et al., 2018) and its unbiased counterpart, the Kernel Inception Distance (Bińkowski et al., 2018) are both used for evaluating generative models, but unlike MAUVE, do not take into account a trade-off between different kinds of errors between the learned and a reference distribution. Sajjadi et al. (2018) and Kynkäänniemi et al. (2019) both proposed metrics based on precision-recall curves. Djolonga et al. (2020) proposed information divergence frontiers as a unified framework encompassing both these works as special cases. MAUVE extends the above line of work and is operationalized for open-ended text generation, applicable for data generated by large-scale neural language models. Complementary to this work, Liu et al. (2021a) study the theory of information divergence frontiers, proving non-asymptotic bounds on the estimation and quantization error.

4.4 Experiments

We perform three sets of experiments to validate MAUVE. Our first set of experiments (Section 4.4.1) examine how known properties of generated text with respect to generation length, decoding algorithm, and model size can be identified and quantified by MAUVE. Next, in Section 4.4.2 we demonstrate that MAUVE is robust to various embedding strategies, quantization algorithms, and hyperparameter settings. Finally, in Section 4.4.3 we find that MAUVE correlates with human judgments. The code as well as the scripts to reproduce the experiments are available online.[3]

Tasks. We consider open-ended text generation using a text completion task (Holtzman et al., 2020; Welleck et al., 2020b) in three domains: web text, news, and stories. Each domain consists of a sequence dataset split into (context, continuation) pairs. Given a context $\boldsymbol{x}_{1:k}$, the task is to generate a continuation $\hat{\boldsymbol{x}}_{k+1:T} \sim Q(\cdot \mid \boldsymbol{x}_{1:k})$, forming a completion. Each ground-truth completion $\boldsymbol{x}_{1:T}$ is considered a sample from the true distribution P, while the completion $(\boldsymbol{x}_{1:k}, \hat{\boldsymbol{x}}_{k+1:T})$ is considered a sample from Q. The datasets, context, completion lengths, and number of completions used for each domain are shown in Table 4.2.

Models. As the language model $\hat{P}(\cdot)$, we use GPT-2, a large-scale transformer (Vaswani et al., 2017) pretrained on the web text dataset (see (Radford et al., 2019)), that is representative of state-of-the-art

[3] `https://github.com/krishnap25/mauve-experiments`.

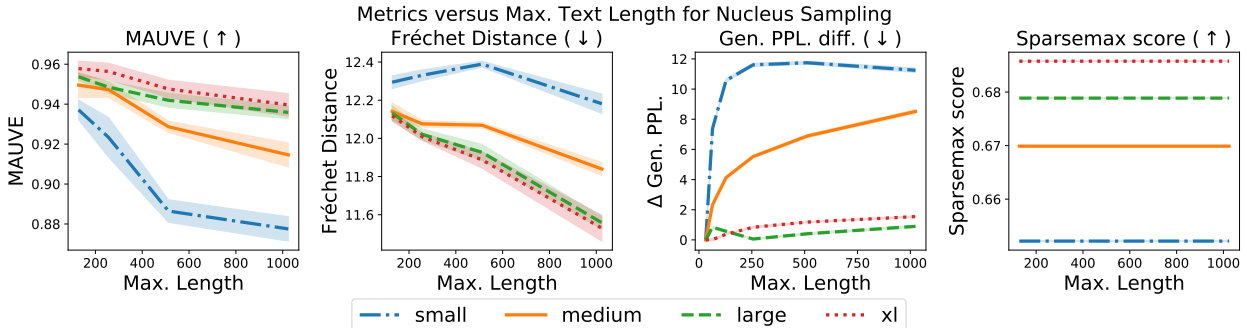


Figure 4.6: Generation quality versus maximum generation length according to MAUVE and three alternative measures (web text, GPT-2). MAUVE is the only comparison measure that identifies that generation quality decreases monotonically with increasing text length. The shaded area shows one standard deviation over generations from 5 random seeds.

autoregressive language models. As the embedding model $M(\cdot)$ we use GPT-2 Large, and compare others in Section 4.4.2.

Decoding Algorithms. We consider three common decoding algorithms: ancestral sampling, greedy decoding, and nucleus sampling. We refer to Section 4.1 for their definitions.

We also consider an adversarial sampling procedure, designed to generate low-quality text that nevertheless matches the perplexity of human text. Adversarial perplexity sampling proceeds in two phases: (1) we generate the first 15% of tokens in a sequence uniformly at random from the vocabulary, and (2) we generate the remaining tokens greedily to make the running perplexity of the generated sequence as close as possible to the perplexity of human text.

4.4.1 Quantifying Properties of Generated Text

To study MAUVE's effectiveness as a measure for comparing text distributions, we first examine how MAUVE quantifies known properties of generated text: a good measure should meet expected behavior that is known from existing research on each property. Specifically, we investigate how MAUVE behaves under changes in generation length, decoding algorithm, and model size.

MAUVE quantifies quality differences due to generation length. Although large transformer-based models can generate remarkably fluent text, it has been observed that the quality of generation deteriorates with text length: as the generation gets longer, the model starts to wander, switching to unrelated topics and becoming incoherent (Rashkin et al., 2020). As a result, an effective measure should indicate lower quality (e.g. lower MAUVE) as generation length increases.

Figure 4.6 shows MAUVE as the generation length increases, along with three alternative metrics: generation perplexity, sparsemax score, and Fréchet distance (Heusel et al., 2017; Semeniuta et al., 2018). MAUVE reflects the desired behavior, showing a decrease in quality (lower MAUVE) as generation length grows, with the trend consistent across model sizes. The other three metrics, however, show less favorable trends. Fréchet distance indicates *improving* quality as the length increases, while generation perplexity shows non-monotonic quality trends for the small and large models. Finally, language modeling metrics such as the sparsemax score (Martins et al., 2020) remain constant, since they do not depend on the samples generated.

MAUVE identifies quality differences between decoding algorithms. Recent work has identified two clear trends in open-ended text generation with standard autoregressive models: (1) using greedy decoding results in repetitive, degenerate text (Holtzman et al., 2020; Welleck et al., 2020b;a); (2) nucleus sampling (and related truncated sampling methods) yields higher quality text than ancestral sampling (Fan et al., 2018; Holtzman et al., 2020).[4] An effective measure should thus indicate the quality relationship greedy $\prec$ ancestral $\prec$ nucleus.

Table 4.3 summarizes MAUVE's quality measures of greedy decoding, ancestral sampling, and nucleus sampling, along with alternative automated metrics and a human quality score. MAUVE correctly identifies the expected quality relationship, assigning the lowest quality to greedy decoding (.016) followed by ancestral sampling (.882), and the highest quality to nucleus sampling (.940). Other commonly-used metrics fail to identify this relationship: generation perplexity rates the highly degenerate greedy-decoded text as better than ancestral sampling (11.324 vs. 19.284), while the language-modeling metrics (SP, JS, ε-PPL) rate nucleus-decoded text as equal to or worse than greedy decoding or ancestral sampling. Further, as we show in Appendix B (in particular, Section B.3), MAUVE rightly identifies degeneracy of beam search, thus quantifying the qualitative observations of Holtzman et al. (2020). Finally, generation perplexity falls victim to the adversarial decoder (Adv.), unlike MAUVE.[5]

MAUVE quantifies quality differences due to model size. Scaling the model size has been a key driver of recent advances in NLP, with larger models leading to better language modeling and higher quality generations in open-ended settings (Radford et al., 2019; Brown et al., 2020). An effective metric should capture the relationship between model size and generation quality, which we verify with human quality scores.

Table 4.4 shows MAUVE's quality measures as the model size increases, along with alternatives and human quality scores. MAUVE increases as model size increases, agreeing with the human quality measure and the expectation that larger models should have higher quality generations. The widely-used generation perplexity, however, incorrectly rates the large model's text as the best. Although the language modeling metrics (SP, JS, and ε-PPL) capture the size-quality relationship, they are constant with respect to length (Figure 4.6), and did not correctly quantify decoding algorithm quality (Table 4.3).

Table B.1 in Section B.3 shows additional results with ancestral sampling. In this case, human evaluators rate generations from the small model as better than those from the medium model. Interestingly, MAUVE also identifies this relationship, agreeing with the human ratings, in contrast to the other automatic metrics we surveyed.

Summary. MAUVE identifies properties of generated text that a good measure should capture, related to length, decoding algorithm, and model size. In contrast, commonly used language modeling and statistical measures did not capture all of these properties. Unlike these alternatives, which capture a single statistic or relate to a single point on the divergence curve, MAUVE's summary measure incorporates type I errors that quantify the degenerate text produced by greedy decoding (recall Figure 4.1), while capturing distribution-level information that describes quality changes from generation length, model size, and the nuanced distinction between ancestral and nucleus sampling.

[4] In general this relationship depends on the nucleus hyperparameter p and task. Here, we follow the same settings as Holtzman et al. (2020) and additionally include a human-assessed measure of quality.

[5] The results are consistent across model sizes and random seeds (see Section B.3).

	Adv.	Greedy	Sampling	Nucleus
Gen. PPL (↓)	**0.05**	11.3	19.3	1.54
Zipf (↓)	0.03	0.02	0.02	**0.01**
Self-BLEU (↓)	0.07	0.03	**0.02**	0.03
SP (↑)	–	0.50	**0.69**	0.69
JS (↓)	–	**0.35**	0.37	0.36
ε-PPL (↓)	–	497	**11.4**	13.7
MAUVE (↑)	0.06	0.02	0.88	**0.94**
Human (↑)	–	–	9.0	**15.7**

Table 4.3: Generation quality w.r.t different **decoding algorithms** (web text, GPT-2 xl) under various metrics, and humans. MAUVE correctly captures the relationship greedy $\prec$ ancestral $\prec$ nucleus, and rates the adversarial decoder's text as low quality. Results are consistent across model sizes and random seeds. The arrow ↑ (resp. ↓) denotes that a larger (resp. smaller) value of the metric represents a smaller gap between model text and human text, and the bold-faced/highlighted entries denote the best decoding algorithm under each metric. The human ratings (last row) are described in Section 4.4.3 (Adv./Greedy are omitted because their text is degenerate).

	Small	Medium	Large	XL
Gen. PPL (↓)	11.2	8.5	**0.9**	1.5
Zipf (↓)	0.06	**0.00**	0.02	0.01
Self-BLEU (↓)	0.05	**0.02**	0.03	0.03
SP (↑)	0.65	0.67	0.68	**0.69**
JS (↓)	0.41	0.39	0.37	**0.36**
ε-PPL (↓)	25.9	18.8	14.9	**13.7**
MAUVE (↑)	0.878	0.915	0.936	**0.940**
Human (↑)	−15.9	−3.4	12.6	**15.7**

Table 4.4: Generation quality w.r.t different **model sizes** (web text, nucleus sampling) under various metrics, as well as human evaluators. MAUVE captures the relationship between model size and generation quality, agreeing with human-evaluated quality. Results are consistent across random seeds and decoding algorithms. The arrow ↑ (resp. ↓) denotes that a larger (resp. smaller) value of the metric represents a smaller gap between model text and human text, and the bold-faced/highlighted entries denote the best model size under each metric. The human ratings (last row) are described in Section 4.4.3.

4.4.2 Approximations in MAUVE

MAUVE summarizes the divergence between two text distributions with an approximation that relies on two components: an embedding model $M(\boldsymbol{x})$ and a quantization algorithm $\mathcal{A}$ (Section 4.2, Eq. (4.5)). We study the effects of these two components.

MAUVE works with alternative embedding models. Figure 4.7 (left) shows that MAUVE with features from RoBERTa-large (Liu et al., 2019) gives qualitatively similar trends across model size and decoding as MAUVE with features from GPT-2 large. Quantitatively, the Spearman rank correlation between them across all models and decoders is 0.993. We observe that RoBERTa penalizes smaller models more than GPT-2 but rates greedy decoding higher. We leave a further study of inductive biases in the different embedding models to future work.

MAUVE is robust to quantization. We compare three different quantization algorithms:

(a) k-Means: We cluster the hidden representations using k-means, and represent them by their cluster membership to get a discrete distribution with size equal to the number of clusters.

(b) Deep Residual Mixture Models (DRMM): As a generalization of k-means, we train a deep generative model known as DRMM (Hämäläinen and Solin, 2020). We convert the soft clustering returned by DRMM into a hard clustering by assigning each point to its most likely cluster, and quantize the data using the cluster membership. We use DRMM with 3 layers and 10 components per layer for a total of 10^3 clusters, and train it for 20 epochs.

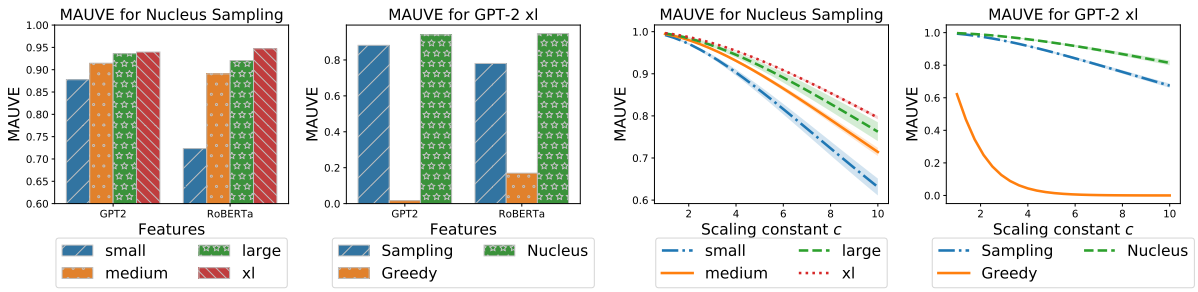


Figure 4.7: Left: MAUVE computed using GPT-2 (default) and RoBERTa (Liu et al., 2019) embeddings, across model sizes and decoding algorithms; see Table B.7 in the Appendix for further results. The Spearman rank correlation between the two is **0.993** across model sizes and decoding algorithms. **Right**: Effect of the scaling constant c on MAUVE. The choice of c does not affect the relative order of the curves but only the numerical value. We use $c = 5$ to get interpretable values with both nucleus and greedy decoding.

(c) Lattice Quantization: We learn a 4-dimensional feature representation of the vectors $M(\boldsymbol{x})$ using a deep network that maintains the neighborhood structure of the data while encouraging the features to be uniformly distributed on the unit sphere (Sablayrolles et al., 2019). We quantize the data on a uniform lattice into 744 bins.

We compare different choices of the quantization to k-means with $k = 500$, which is our default. The Spearman rank correlation between MAUVE computed with k-means for k ranging from 100 to 5000 correlates nearly perfectly with that of $k = 500$. In particular, the Spearman correlation is exactly 0.99 or 1.00. Likewise, MAUVE computed with DRMM or lattice quantization has a near-perfect Spearman correlation of at least 0.99 with k-means. While the actual numerical value of MAUVE could vary with the quantization algorithm, these results show that the *rankings induced by various variants of* MAUVE *are nearly identical.*

Practical recommendation for scaling parameter. Figure 4.7 (right) shows the effects of adjusting the scaling parameter c, which does not affect the relative order of the divergence curve but adjusts the numerical value returned by MAUVE. As a practical recommendation, we found $c = 5$ to yield interpretable values.

4.4.3 Correlation with Human Judgments

An effective metric should yield judgments that correlate highly with human judgments, assuming that human evaluators represent a gold-standard.[6] We evaluate how MAUVE's quality judgments correlate with human quality judgments. In our study, a quality judgment means choosing a particular (model, decoder) setting based on the resultant generations.

Evaluation Protocol. To obtain human judgments, we employ a pairwise setup: at each round, an annotator receives a context and continuations from two different (model, decoder) settings, and selects the continuation they found more natural using a 5-point Likert scale. Our interface for collecting annotations, as well as further details and additional results are provided in Appendix B.

[6]Concurrent work has shown that human evaluation might not always be consistent (Clark et al., 2021; Karpinska et al., 2021); however human judgments continue to be the gold standard for evaluating open-ended text generation.

Metric	Task	Gen. PPL	Zipf Coef.	REP	Distinct-4	Self-BLEU	MAUVE
Human-like/BT	Web text	0.810	0.833	−0.167	0.738	0.595	**0.952**
Interesting/BT	Web text	0.643	0.524	−0.143	0.524	0.405	**0.810**
Sensible/BT	Web text	0.738	0.690	−0.071	0.595	0.524	**0.857**
% Disc. Acc.	News	0.468	0.595	0.792	0.653	0.516	**0.956**
% Disc. Acc.	Stories	0.643	0.643	0.250	0.750	0.857	**0.893**

Table 4.5: Correlation of various similarity measures with human judgments when available, and the accuracy of a trained discriminator otherwise. "BT" denotes the Bradley-Terry score for a pairwise human evaluation (Section 4.4.3). Boldfaced/highlighted numbers indicate the highest correlation in each row. We observe that MAUVE has the highest correlation with human evaluation and discriminator accuracy.

We collect these annotations for web text generation with 8 different (model, decoder) settings plus a ninth setting for human-written continuations. Each setting is a GPT-2 model size paired with either ancestral or nucleus sampling. This gives us a total of 36 pairs of settings. Given the known difficulties with human evaluation of longer texts (Ippolito et al., 2020), we use a maximum completion length of 256 tokens. We obtain 90 preference ratings for each pair of settings, coming from a total of 214 crowd-workers from the Amazon Mechanical Turk platform. The evaluators were paid USD 0.40 per evaluation based on an estimated wage of USD 16 per hour.

We convert these pairwise preferences to a ranking by fitting a Bradley-Terry model (Marden, 1995), a parametric model used to predict the outcome of a head-to-head comparison. In particular, we obtain a score w_i for each setting i so that the log odds of humans preferring setting i to setting j in a head-to-head comparison is given by the difference $w_i - w_j$. For a given comparison measure, we compute the Spearman rank correlation between the comparison measure and the fitted Bradley-Terry coefficients w_i for each of the (model, decoder) settings. The end result is a correlation score in $[-1, 1]$, with higher values meaning that quality judgments using the comparison measure correlate with quality judgments made by human evaluators.

MAUVE correlates with human judgments. Table 4.5 shows the correlation between human judgments and five automatic evaluation metrics obtained using our evaluation protocol on the web text domain. MAUVE correlates highly with human judgments of how human-like (0.952), interesting (0.810), and sensible (0.857) the machine text is. MAUVE's correlations with human judgments are substantially higher than those for the other automated measures; for instance, the commonly-used generation perplexity has correlations that are 0.12 to 0.17 lower than MAUVE's. The results suggest that MAUVE may act as an effective, automatic surrogate for costly human judgments.

MAUVE correlates with learned discriminators. We also measure the quality of generations by how well a trained model (a discriminator) can distinguish between real and generated text (Lopez-Paz and Oquab, 2017). We report the test accuracy of a binary classifier trained to discriminate between machine and human text; a lower discrimination accuracy implies that the generation is harder to distinguish from human text. We report the accuracy of Grover mega as the discriminator for the news generations as it produced the highest discrimination accuracy (Zellers et al., 2019) while we use GPT-2 large for the story domain. As seen in Table 4.5, MAUVE correlates the highest with the discrimination accuracy (0.96 for news and 0.89 for stories) among all comparison measures. Computing the discrimination accuracy for each (model, decoder) pair requires fine-tuning a separate model, which is particularly expensive for large

models such as Grover-mega. MAUVE, on the other hand, does not require any training.

4.5 Discussion and Conclusion

We presented MAUVE, an automatic measure of the gap between neural text and human text for open-ended text generation. MAUVE measures the area under a divergence curve, formalizing and summarizing a spectrum of errors that capture phenomena present in machine and human-generated text. MAUVE correlated with human judgments and identified quality differences due to generation length, decoding algorithm, and model size, which prior metrics struggle to capture. Automated metrics have driven advances in computer vision and many other machine learning domains. MAUVE's principled foundation and strong empirical performance offers a similar path forward for open-ended text generation systems.

MAUVE introduces a paradigm of distributional comparison for text generation. It is fundamentally different from typical reference-based measures used in closed-ended text generation tasks such as translation and summarization. MAUVE is a promising alternative to reference-based measures in the settings where multiple references are available, or no references are available. The latter is particular important for most of 7000 or so recognized languages without massive datasets that power models in English. Continuing the example of translation, the Assamese language is spoken by over 15 million people worldwide but lacks any large scale corpora.[7] Adapting MAUVE to this setting while leveraging multi-lingual embedding models (e.g., Conneau et al., 2020) could be an impactful direction for future work.

Enormous language models are pretrained on uncurated data from crawling the internet and have been shown to output insensitive, hateful, racist, and otherwise biased text (e.g., Gehman et al., 2020); see also Chapter 1. While cleaning up the training corpus is challenging due to the massive scale of the data, it is more realistic to be able to obtain small samples of unbiased data. A useful extension for MAUVE is to be able to identify and flag biased text from generation models when compared against the clean data. More generally, instead of comparing MAUVE(P, Q) between a model Q and human distribution P, it could be beneficial to compute MAUVE(P', Q), where P' is an automatic modification of P based on additional desiderata such as fairness.

More generally, controllable text generation has emerged as an important field of research to deal with the unpredictability of enormous language models (see e.g., Prabhumoye et al., 2020; Hartvigsen et al., 2022). Apart from the criteria we considered in this chapter, we must also evaluate the effectiveness of the applied controls in order to assess controllable generation. A particular interesting direction for future work is to integrate gradient-based controls (e.g. Qin et al., 2022) into MAUVE for controllable generation.

The issue of misinformation, disinformation, and fake news on the internet is a major open problem. A potential solution to make social networks robust to adversarial actors propagating misinformation is by attributing authorship to news articles and empowering users to make informed decisions about the authenticity of the information they consume. Approaches based on MAUVE are promising avenues to address this problem of credit attribution.

[7] The largest monolingual corpus of the Assamese language contains roughly 32 million tokens (0.5 GB in size) (Kakwani et al., 2020), while English language models are trained on terabytes of of data; cf. Chapter 1.

Chapter 5

RFA: Robust Aggregation for Federated Learning

In this chapter and the next, we turn to the revolution of federated learning. Recall that federated learning is a key paradigm for machine learning and analytics on mobile, wearable and edge devices (McMahan et al., 2017; Kairouz et al., 2021b). It has found widespread applications ranging from mobile apps deployed on millions of devices (Yang et al., 2018; Ammad-ud din et al., 2019), to sensitive healthcare applications (Pantelopoulos and Bourbakis, 2009; Huang et al., 2019).

As we discussed in Chapter 1, federated learning consists of a number of devices with privacy-sensitive data collaboratively optimizing a machine learning model under the orchestration of a central server, while keeping the data fully decentralized and private. Recent work has looked beyond supervised learning to domains such as data analytics but also semi-, self- and un-supervised learning, transfer learning, meta learning, and reinforcement learning (Kairouz et al., 2021b; Ren et al., 2019; Lin et al., 2020; Zhuang et al., 2021).

We study a question relevant in all these areas: robustness to corrupted updates. Federated learning relies on the aggregation of updates contributed by participating devices, where the aggregation is privacy-preserving. Sensitivity to corrupted updates, caused either by adversaries intending to attack the system or due to failures in low-cost hardware, is a vulnerability of the usual approach. The standard arithmetic mean aggregation in federated learning is not robust to corruptions, in the sense that even a single corrupted update in a round is sufficient to degrade the global model for all devices. In one dimension, the median is an attractive aggregate for its robustness to outliers. We adopt this approach to federated learning by considering a classical multidimensional generalization of the median, known variously as the geometric or spatial or L_1 median (Maronna et al., 2006).

Our robust approach preserves the privacy of the device updates by iteratively invoking the secure multi-party computation primitives used in typical non-robust federated learning (Bonawitz et al., 2017; Bell et al., 2020). A device's updates are information-theoretically protected in that they are computationally indistinguishable from random noise and the sensitivity of the final aggregate to the contribution of each device is bounded. Our approach is scalable since the underlying secure aggregation algorithms are implemented in production systems across millions of mobile users across the planet (Bonawitz et al., 2019). The approach is communication-efficient, requiring a modest $3\times$ the communication cost of the non-robust setting to compute the non-linear aggregate in a privacy-preserving manner.

Contributions. This chapter makes the following original contributions:

(a) *Robust Aggregation:* We design a novel robust aggregation oracle based on the classical geometric median. We analyze the convergence of the resulting federated learning algorithm, RFA, for least-squares estimation and show that the proposed method is robust to update corruption in up to half the devices in federated learning with bounded heterogeneity. We also describe an extension of the framework to handle arbitrary heterogeneity via personalization.

(b) *Algorithmic Implementation:* We show how to implement this robust aggregation oracle in a practical and privacy-preserving manner. This relies on an alternating minimization algorithm which empirically exhibits rapid convergence. This algorithm can be interpreted as a numerically stable version of the classical algorithm of Weiszfeld (Weiszfeld, 1937), thus shedding new light on it.

(c) *Numerical Simulations:* We demonstrate the effectiveness of our framework for data corruption and parameter update corruption, on federated learning tasks from computer vision and natural language processing, with linear models as well as convolutional and recurrent neural networks. In particular, our results show that the proposed RFA algorithm (i) outperforms the standard FedAvg (McMahan et al., 2017), in high corruption and (ii) nearly matches the performance of the FedAvg in low corruption, both at *1-3 times the communication cost.* Moreover, the proposed algorithm is agnostic to the actual level of corruption in the problem instance.

We open source an implementation of the proposed approach in TensorFlow Federated (rfa, 2019b). The Python code and scripts used to reproduce experimental results are publicly available online (rfa, 2019a).

Overview. We give an overview of federated learning in Section 5.1. Section 5.2 describes the problem formulation and tradeoffs of robustness. Section 5.3 proposes a robust aggregation oracle and presents a convergence analysis of the resulting robust federated learning algorithm. Finally, Section 5.4 gives comprehensive numerical simulations demonstrating the robustness of the proposed federated learning algorithm compared to standard baselines.

5.1 Background: Federated Learning

Federated learning consists of client devices which collaboratively train a machine learning model under the orchestration of a central server (McMahan et al., 2017; Kairouz et al., 2021b). The data remains local to the client devices while the job of the server is to orchestrate the training.

Federated learning comes in two flavors. *Cross-device federated learning* refers to the setting with a massive number of clients (in the millions or billions) such as mobile phones and other smart devices; this is already run at a global scale by major service providers, such as the keyboard for Android mobile phones (Hard et al., 2018). Here, the amount of data and compute available per client device is limited, the clients are unreliable, and only a small fraction of the clients might be available for training. *Cross-silo federated learning* refers to the setting with a small number of clients (2 to 100) such as hospitals, which have huge amounts of data and compute power, and are always available for training. Across both these scenarios, a special emphasis is placed on user privacy and data heterogeneity. The methods proposed in this dissertation are broadly applicable in both scenarios, but we will tailor the discussion to the cross-device setting.

We consider the supervised learning setting to make the setup precise. Suppose that we have n client devices, where each device i has a distribution D_i over some data space such that the data on the client is sampled i.i.d. from D_i. Let the vector $w \in \mathbb{R}^d$ denote the parameters of a (supervised) learning model and let $f(w; z)$ denote the loss of model w on input-output pair z, such as the mean-squared-error loss. Then, the objective function of device i is $F_i(w) = \mathbb{E}_{z \sim D_i}[f(w; z)]$.

Next, we turn to the two components of the "learning" aspect, i.e., the overall objective function and the optimization algorithm.

Objective Function. In federated learning, we aim to find a model $w^\star$ by minimizing some objective function and then deploy it on each of the devices, including those not seen during training. Ideally, we would want $w^\star$ to achieve good performance on *each* test client device *simultaneously*. However, this is a difficult multiobjective optimization problem which might not even admit a simultaneous minimizer. The usual approach is to construct an objective function of the form

$$\min_{w\in\mathbb{R}^d} h\big(F_1(w),\cdots,F_n(w)\big) \tag{5.1}$$

where $h:\mathbb{R}^n\to\mathbb{R}$ combines the losses on each of the n devices into a single objective. The dominant approach (McMahan et al., 2017) corresponds to $h(f_1,\cdots,f_n)=(1/n)\sum_{i=1}^n f_i$, so that the objective is simply the average of the per-device objectives. More generally, we assign device i a weight by $\alpha_i>0$ to arrive at the objective

$$\min_{w\in\mathbb{R}^d}\left[F(w):=\sum_{i=1}^n \alpha_i\, F_i(w)\right]. \tag{5.2}$$

In practice, the weight α_i is chosen proportional to the amount of data on device i. For instance, in an empirical risk minimization setting, each D_i is the uniform distribution over a finite set $\{z_{i,1},\cdots,z_{i,N_i}\}$ of size N_i. It is common practice to choose $\alpha_i=N_i/N$ where $N=\sum_{i=1}^n N_i$ so that the objective $F(w)=(1/N)\sum_{i=1}^n\sum_{j=1}^{N_i} f(w;z_{i,j})$ is simply the unweighted average over all samples from all n devices.

Federated Learning Algorithms. Typical federated learning algorithms run in synchronized rounds of communication between the server and the devices with some local computation on the devices based on their local data, and aggregation of these updates to update the server model. The de facto standard training algorithm is FedAvg (McMahan et al., 2017), which runs as follows.

(a) The server samples a set S_t of m clients from $[n]$ and broadcasts the current model $w^{(t)}$ to these clients.

(b) Staring from $w_{i,0}^{(t)}=w^{(t)}$, each client $i\in S_t$ makes τ local gradient or stochastic gradient descent steps for $k=0,\cdots,\tau-1$ with a learning rate γ:

$$w_{i,k+1}^{(t)}=w_{i,k}^{(t)}-\gamma\nabla F_i(w_{i,k}^{(t)})\,. \tag{5.3}$$

(c) Each device $i\in S_t$ sends to the server a vector $w_i^{(t+1)}$ which is simply the final iterate, i.e., $w_i^{(t+1)}=w_{i,\tau}^{(t)}$. The server updates its global model using the weighted average

$$w^{(t+1)}=\frac{\sum_{i\in S_t}\alpha_i w_i^{(t+1)}}{\sum_{i\in S_t}\alpha_i}\,. \tag{5.4}$$

The only local computation on client devices takes place in step (b), where the server model is updated on the local data available on-device with stochastic gradient steps. The server-to-client communication takes place via model broadcast in step (a), while the client-to-server communication takes place in step (c) via aggregation.

5.1.1 Key Factors of Federated Learning

The design of a federated learning objective and optimization algorithm are determined by the following factors (Kairouz et al., 2021b; Li et al., 2020a; Gafni et al., 2022): communication efficiency, privacy, robustness, and statistical heterogeneity.

Communication Efficiency. Besides the computation cost, the communication cost is an important parameter in distributed optimization. While communication is relatively fast in the datacenter, that is not the case of federated learning. The repeated exchange of massive models between the server and client devices over resource-limited wireless networks makes communication over the network more of a bottleneck in federated learning than local computation on the devices. Therefore, training algorithms should be able to trade-off more local computation for lower communication, similar to step (b) of FedAvg above. While the exact benefits (or lack thereof) of local steps is an active area of research, local steps have been found empirically to reduce the amount of communication required for a moderately accurate solution (McMahan et al., 2017; Wang et al., 2021).

Accordingly, we set aside the local computation cost for a first order approximation, and compare algorithms in terms of their total communication cost (Kairouz et al., 2021b). Since typical federated learning algorithms proceed in synchronized rounds of communication, we measure the complexity of the algorithms in terms of the number of communication rounds.

Privacy. While the privacy-sensitive data $z \sim D_i$ is kept local to the device, the model updates $w_i^{(t+1)}$ might also leak privacy. To add a further layer of privacy protection, the server is not allowed to inspect individual updates $w_i^{(t+1)}$ in the aggregation step (c); it can only access the aggregate $w^{(t+1)}$.

We make this precise through the notion of a *secure average oracle*. Given m devices with each device i containing $w_i \in \mathbb{R}^d$ and a scalar $\beta_i > 0$, a secure average oracle computes the average $\sum_{i=1}^m \beta_i w_i / \sum_{i=1}^m \beta_i$ at a total communication of $O(md + m \log m)$ bits such that no w_i or β_i are revealed to either the server or any other device.

In practice, a secure average oracle is implemented using cryptographic protocols based on secure multi-party computation (Bonawitz et al., 2017; Bell et al., 2020). These require a communication overhead of $O(m \log m)$ in addition to $O(md)$ cost of sending the m vectors. We postpone a detailed description to Section 5.1.2, and summarize it here.

First, the vector $\beta_i w_i$ is dimension-wise discretized on the ring $\mathbb{Z}_M^d$ of integers modulo M in d-dimensions. Then, a noisy version $\tilde{w}_i$ is sent to the server, where the noise is designed to satisfy:

- correctness up to discretization, by ensuring $\sum_{i=1}^m \tilde{w}_i \mod M = \sum_{i=1}^m \beta_i w_i \mod M$ with probability 1, and,
- privacy preservation from honest-but-curious devices and server in the information theoretic sense, by ensuring that $\tilde{w}_i$ is computationally indistinguishable from $\zeta_i \sim \text{Uniform}(\mathbb{Z}_M^d)$, irrespective of w_i and β_i.

As a result, we get the correct average (up to discretization) while not revealing any further information about a w_i or β_i to the server or other devices, beyond what can be inferred from the average. Hence, no further information about the underlying data distribution D_i is revealed either. Unless stated otherwise, we assume for simplicity that the secure average oracle returns the exact update, i.e., we ignore the effects of discretization on the integer ring and modular wraparound. This assumption is reasonable for a large enough value of M. Finally, we note that a secure average oracle can also be used as a primitive designing a differentially private aggregation mechanism (Kairouz et al., 2021a).

Robustness. We would like a federated learning algorithm to be robust to corrupted updates contributed by malicious devices or hardware/software failures. FedAvg uses an arithmetic mean to aggregate the device updates in (5.4), which is known to not be robust (Huber, 1964). This can be made precise by the notion of a breakdown point (Donoho and Huber, 1983), which is the smallest fraction of the points which need to be changed to cause the aggregate to take on arbitrary values. The breakdown point of the mean is 0, since only one point needs to changed to arbitrarily change the aggregate (Maronna et al., 2006). This means in federated learning that a single corrupted update, either due to an adversarial attack or a failure, can arbitrarily change the resulting aggregate in each round. We will give examples of adversarial corruptions in Section 5.2. The focus of this chapter is to improve the robustness of FedAvg.

Statistical Heterogeneity. The training objective (5.2) fits a model to minimize the loss on the average training distribution $D_\alpha := \sum_{i=1}^n \alpha_i D_i$, since the equality $F(w) = \mathbb{E}_{z\sim D_\alpha}\left[f(w;z)\right]$ holds. Since the data is generated by a diverse set of users, the distribution D of a new test device might be quite different from that of the training distribution D_α; in other words, federated learning suffers from a *train-test mismatch*. In particular, the resulting model w can sacrifice performance on "difficult" clients in order to perform well on average. Therefore, a model which works well *on average* over all devices might not work well on *each individual* device whose distribution D might be quite different from D_α. We will address the this train-test mismatch in Chapter 6.

5.1.2 Secure Aggregation

Secure Aggregation refers to a class of algorithms where a group of mutually distrustful users $\{1, \cdots, n\}$ each hold a private value $x_i \in \mathbb{R}^d$ and collaborate to compute an aggregate value, such as the sum $s = \sum_{i=1}^n x_i$. Such algorithms are more broadly studied by cryptographers under the name "Secure Multi-Party Computation" (Evans et al., 2018). The key *security* requirement of secure aggregation is that the computation must proceed without revealing to one another any information about their private value except what is learnable from the aggregate value s itself. We focus here on the sum s, since other related quantities such as the weighted and unweighted averages can be implemented from the sum. Secure aggregation is used as a communication primitive in federated learning (Bonawitz et al., 2017).

We now define a *secure summation oracle*. Recall that $\mathbb{Z}$ denotes the set of integers and $\mathbb{Z}_M$ denotes the ring of integers modulo $M \in \mathbb{Z}$.

Definition 5.1. For a fixed integer $M \in \mathbb{Z}_+$ and a dimension $d \in \mathbb{Z}_+$, the secure summation oracle is a map $\mathcal{S} : (\mathbb{Z}_M^d)^n \to \mathbb{Z}_M$ which returns $\mathcal{S}(x_1, \cdots, x_n) = (\sum_{i=1}^n x_i) \mod M$. The secure summation oracle reveals no further information about each x_i to a privacy adversary.

Implementation of a Secure Summation Oracle. We assume that each pair of users have access to a secure communication channel. In the context of federated learning, these messages are encrypted and routed via the coordinating server. Consider the following scheme, illustrated in Figure 5.1:

- Each pair i, j of users communicate over their secure channel to sample (dependent) random variables $\xi_{i,j}, \xi_{j,i}$ such that each of their marginal distributions is uniform over $\mathbb{Z}_M^d$ but $\xi_{i,j} + \xi_{j,i} \equiv 0 \mod M$.
- Each client sends y_i to the server, which is computed as

$$y_i = \left(x_i + \sum_{j \neq i} \xi_{i,j} \right) \mod M\,. \tag{5.5}$$

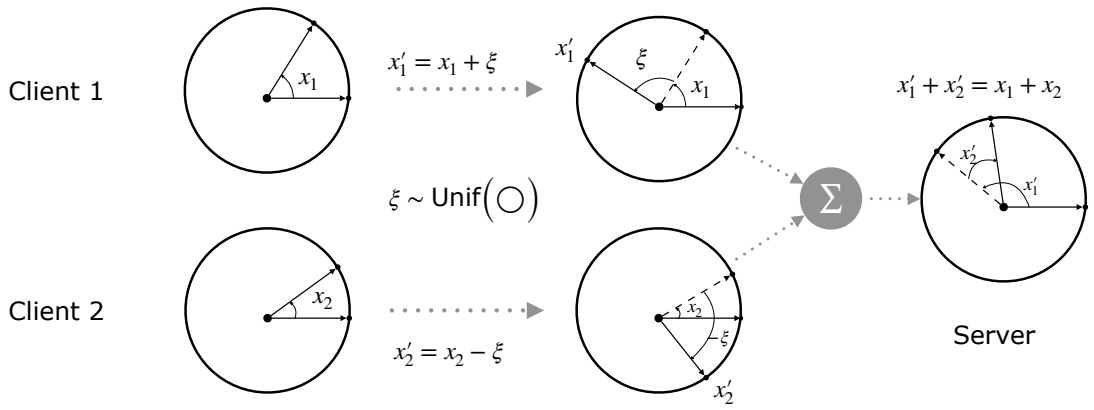


Figure 5.1: Illustration of secure summation of two integers $x_1, x_2 \in \mathbb{Z}_M$. The modular ring $\mathbb{Z}_M$ is denoted by $\bigcirc$ and visualized as a circle, and elements of the ring are represented by angular offsets because of the wraparound property of modular arithmetic. Secure summation is implemented by adding pairwise canceling noise $\xi \sim \text{Unif}(\bigcirc)$ such that the transmitted vectors $x_1' = x_1 + \xi$ and $x_2' = x_2 - \xi$ appear to be uniformly distributed on the modular ring, while their summation is correct, i.e., $x_1' + x_2' = x_1 + x_2$.

- The server computes and returns the estimate

$$\hat{s} = \left(\sum_{i=1}^{n} y_i \right) \mod M \,. \tag{5.6}$$

This procedure returns the correct sum and is privacy-perserving in that the communicated quantity appears uniformly random to the server.

Property 5.2. *Consider the quantities $\{y_i\}_{i=1}^n$ and $\hat{s}$ from* (5.5) *and* (5.6)*. We have the following:*
(a) The output $\hat{s}$ of the protocol satisfies $\hat{s} = (\sum_{i=1}^n x_i) \mod M$.
(b) For any $x_i \in \mathbb{Z}_M^d$, we have that the random variable y_i is uniformly distributed over $\mathbb{Z}_M^d$.

Proof. The first property follows because the pairwise noise cancels out: $\xi_{i,j} + \xi_{j,i} \equiv 0 \mod M$. The second property follows because an offset does not modify the uniform distribution over a finite ring. Indeed, if $U, U' \sim \text{Uniform}(\mathbb{Z}_M)$ then both $(U + c) \mod M$ and $(U + U') \mod M$ are uniformly distributed on $\mathbb{Z}_M$ for any $c \in \mathbb{Z}_M$. □

A number of improvements are necessary to make an implementation of a secure summation oracle scalable and robust to industrial use-cases, such as gracefully handling clients who dropout of the protocol. We refer to Bonawitz et al. (2017) for a thorough account.

From Secure Summation over Rings of Integers to Secure Averaging over Reals. Secure summation over reals $\mathcal{S}' : (\mathbb{R}^d)^n \to \mathbb{R}^d$ involves scaling, clipping and quantization to a ring of integers of size M followed by secure summation.

The hyperparameters are those of the clipping and quantization. Concretely, these are the clipping threshold $r > 0$, and the modulo base M, which is also the size of the ring of integers.

- Clip each coordinate to $[-r, r]$. Mathematically, we apply the map $C_r : \mathbb{R} \to [-r, r]$ coordinate-wise given by

$$C_r(u) = \begin{cases} u, & \text{if } u \in [-r, r], \\ \text{sign}(u)\, r, & \text{else.} \end{cases} \tag{5.7}$$

- We linearly map the interval $[-r, r]$ to $[0, (M-1)/n]$. This ensures no modular wraparound after quantization. Concretely, the map and inverse map are given by

$$\phi(u) = \left(\frac{M-1}{2rn}\right) u + \frac{M-1}{2n}, \quad \text{and,} \quad \phi^{-1}(v) = \left(\frac{2rn}{M-1}\right) v - r\,. \tag{5.8}$$

- Quantization: This involves rounding real numbers to their nearest smallest integers as $u \mapsto \lfloor u \rfloor$.
- Invert maps: We now invert the scaling operation, i.e., we apply ϕ^{-1}.

We extend each of the above operations from $u \in \mathbb{R}$ to $x \in \mathbb{R}^d$ by applying them coordinate-wise on x. All together, this gives us the following implementation of the secure summation oracle for reals:

$$\mathcal{S}'(x_1, \cdots, x_n) = \phi^{-1}\left(\mathcal{S}\left(\lfloor \phi \circ C_r(x_1) \rfloor, \cdots, \lfloor \phi \circ C_r(x_n) \rfloor\right)\right). \tag{5.9}$$

This clipping scheme satisfies the same privacy properties as in the integer rings. Moreover, the error incurred in the sum can be bounded as follows.

Proposition 5.3. *Fix parameters $r \in \mathbb{R}_+$ and $M \in \mathbb{Z}_+$ and consider the secure summation oracle $\mathcal{S}'$ defined in* (5.9) *and consider inputs $x_1, \cdots, x_n \in \mathbb{R}^d$. Denote the j^{th} component of x_i by $x_{i,j}$. Let $N_c = \sum_{i=1}^n |\{j : x_{i,j} \le r\}|$ be the number of coordinates clipped by the map C_r across all n points. We have the squared error bound on the output of $\mathcal{S}'$:*

$$\left\| \sum_{i=1}^n x_i - \mathcal{S}'(x_1, \cdots, x_n) \right\|^2 \le 3nd \left(\frac{2rn}{M-1}\right)^2 + N_c \left(\max_{i,j} x_{i,j} - r\right)^2.$$

Proof. Since ϕ and ϕ^{-1} are linear maps on reals, we have,

$$\begin{aligned} \left\| \sum_{i=1}^n x_i - \mathcal{S}'(x_1, \cdots, x_n) \right\|^2 &= \left\| \sum_{i=1}^n x_i - \phi^{-1}\left(\sum_{i=1}^n \lfloor \phi \circ C_r(x_i) \rfloor\right) \right\|^2 \\ &= \left\| \sum_{i=1}^n x_i - \sum_{i=1}^n \phi^{-1}\left(\lfloor \phi \circ C_r(x_i) \rfloor\right) \right\|^2 \\ &\le \gamma^2 \sum_{i=1}^n \sum_{j=1}^d \left(\phi^{-1}(\phi(x_{i,j})) - \phi^{-1}\left(\lfloor \phi \circ C_r(x_{i,j}) \rfloor\right)\right)^2 \\ &= \gamma^2 \left(\frac{2rn}{M-1}\right)^2 \sum_{i=1}^n \sum_{j=1}^d \left(\phi(x_{i,j}) - \lfloor \phi \circ C_r(x_{i,j}) \rfloor\right)^2 \end{aligned}$$

When $x_{i,j}$ is not clipped by C_r, the term inside the square is at most 1. If not, we have,

$$\begin{aligned}\big(\phi(x_{i,j}) - \lfloor \phi \circ C_r(x_{i,j}) \rfloor\big)^2 &\le 2\big(1 + \phi(x_{i,j}) - \phi \circ C_r(x_{i,j})\big)^2 \\ &= 2 + 2(\phi(x_{i,j}) - \phi(r))^2 \\ &\le 2 + 2\left(\frac{M-1}{2rn}\right)^2 \left(\max_{i,j} x_{i,j} - r\right)^2 .\end{aligned}$$

Plugging this in completes the proof. □

Example of Secure Aggregation: FedAvg. We consider how secure aggregation can concretely be used for FedAvg (McMahan et al., 2017). Each round of FedAvg includes the aggregation of per-client updates to update the server model in (5.4). The client-to-server communication required for this step can be performed using secure aggregation to avoid the privacy risks of directly sending the per-client updates to the server.

Suppose we wish to aggregate updates $w_1, \cdots, w_m \in \mathbb{R}^d$ from m selected clients with equal weights $\alpha_1 = \cdots = \alpha_m = 1/m$. Assume that the updates satisfy $\max_i \|w_i\|_\infty \le 1$ and we choose $r = 1$ in (5.7) to avoid clipping. The error in the estimation of the mean $(1/m)\sum_{i=1}^m w_i$ to update the server model is, from Proposition 5.3,

$$\left\| \frac{1}{m}\sum_{i=1}^m w_i - \frac{1}{m}\mathcal{S}'(w_1, \cdots, w_m) \right\|^2 \le \frac{12md}{(M-1)^2} .$$

Larger M reduces the error but increases the communication cost, which scales as $O(md \log_2 M)$. This comes from the total number of bits of all coordinates of the noisy quantized updates $y_i \in \mathbb{Z}_M^d$ from (5.5) for each $i = 1, \cdots, m$.

Let us instantiate this bound concretely. In typical federated learning scenarios, the number of clients per round m is around 10^3, while the model dimensionality d is of the order 10^7 to 10^8 (i.e., 10 to 100 million model parameters). Considering 32-bit floating point numbers, if we choose $M \approx 2^{32} \approx 10^{10}$, the squared error is at most 10^{-8}.

5.2 Problem Setup: Federated Learning with Corruptions

In this work, we aim to address the lack of robustness of FedAvg. We formally describe our corruption model and discuss the trade-offs introduced by requiring robustness to corrupted updates.

A popular robust aggregation of scalars is the median rather than the mean. We investigate a multidimensional analogue of the median while respecting the other two factors: communication efficiency and privacy. While the non-robust mean aggregation can be computed with secure multi-party computation via the secure average oracle, it is unclear if a robust aggregate can also satisfy this requirement. We discuss this as well as other tradeoffs involving robustness in the next section.

We start with the corruption model used in this work. We allow a subset $\mathcal{C} \subset [n]$ of *corrupted devices* to, unbeknownst to the server, send arbitrary vectors $w_i^{(t+1)} \in \mathbb{R}^d$ rather than the updated model $w_{i,\tau}^{(t)}$ from local data as expected by the server. Formally, we have,

$$w_i^{(t+1)} = \begin{cases} w_{i,\tau}^{(t)}, & \text{if } i \notin \mathcal{C}, \\ H_i\left(w^{(t)}, \{(w_{j,\tau}^{(t)}, D_j)\}_{j \in S_t}\right) & \text{if } i \in \mathcal{C}, \end{cases} \tag{5.10}$$

Table 5.1: Example corruptions and the capability of an adversary they require, as measured along the following axes: **Data write**, where a device $i \in \mathcal{C}$ can replace its local distribution D_i by any arbitrary distribution $\tilde{D}_i$; **Model read**, where a device $i \in \mathcal{C}$ can read the server model $w^{(t)}$ and replace its local distribution D_i by an adaptive distribution $\tilde{D}_i^{(t)}$ depending on $w^{(t)}$; **Model write**, where a device $i \in \mathcal{C}$ can return an arbitrary vector to the server for aggregation as in (5.10), and, **Aggregation**, where a device $i \in \mathcal{C}$ can behave arbitrarily during the computation of an iterative secure aggregate. The last column indicates whether the proposed RFA algorithm is robust to each type of corruption.

Corruption Type	Data write	Model read	Model write	Aggregation	RFA applicable?
Non-adversarial	-	-	-	-	✓
Static data poisoning	Yes	-	-	-	✓
Adaptive data poisoning	Yes	Yes	-	-	✓
Update poisoning	Yes	Yes	Yes	-	✓
Byzantine	Yes	Yes	Yes	Yes	N/A

where H_i is an arbitrary $\mathbb{R}^d$-valued function which is allowed to depend on the global model $w^{(t)}$, the uncorrupted updates $w_{j,\tau}^{(t)}$ as well as the data distributions D_j of each device $j \in S_t$.

This encompasses situations where the corrupted devices are individually or collectively trying to "attack" the global model, that is, reduce its predictive power over uncorrupted data. We define the *corruption level* ρ as the total fraction of the weight of the corrupted devices:

$$\rho = \frac{\sum_{i \in \mathcal{C}} \alpha_i}{\sum_{i=1}^{n} \alpha_i} \,. \tag{5.11}$$

Since the corrupted devices can only harm the global model through the updates they contribute in the aggregation step, we aim to robustify the aggregation in federated learning. However, it turns out that robustness is not directly compatible with the two other desiderata of federated learning, namely communication efficiency and privacy.

The Tension Between Robustness, Communication, and Privacy. We first argue that any federated learning algorithm can only have two out of the three of robustness, communication, and privacy under the existing techniques of secure multi-party computation. The standard approach of FedAvg is communication-efficient and privacy-preserving but not robust, as we discussed earlier. In fact, any aggregation scheme $A(w_1, \cdots, w_m)$ which is a linear function of $w_1, \cdots, w_m$ is similarly non-robust. Therefore, any robust aggregate A must be a non-linear function of the vectors it aggregates.

The approach of sending the updates to the server at a communication cost of $O(md)$ and utilizing one of the many robust aggregates studied in the literature (e.g. Chen et al., 2017; Yin et al., 2018; Alistarh et al., 2018) has robustness and communication efficiency but not privacy. If we try to make it privacy-preserving, however, we lose communication efficiency. Indeed, the secure multi-party computation primitives based on secret sharing, upon which privacy-preservation is built, are communication efficient only for linear functions of the inputs (Evans et al., 2018). The additional $O(m \log m)$ overhead of secure averaging for linear functions becomes $\Omega(md \log m)$ for general non-linear functions required for robustness; this makes it impractical for large-scale systems (Bonawitz et al., 2017). Therefore, one cannot have both communication efficiency and privacy preservation along with robustness.

In this work, we strike a compromise between robustness, communication, and privacy. We will approximate a non-linear robust aggregate as an *iterative secure aggregate*, i.e., as a sequence of weighted averages, computed with a secure average oracle with weights being adaptively updated.

Definition 5.4. A function $A : (\mathbb{R}^d)^m \to \mathbb{R}^d$ is said to be an iterative secure aggregate of $w_1, \cdots, w_m$ with R communication rounds and initial iterate $v^{(0)}$ if for $r = 0, \cdots, R-1$, there exist weights $\beta_1^{(r)}, \cdots, \beta_m^{(r)}$ such that

(i) $\beta_i^{(r)}$ depends only on $v^{(r)}$ and w_i,

(ii) $v^{(r+1)} = \sum_{i=1}^m \beta_i^{(r)} w_i / \sum_{i=1}^m \beta_i^{(r)}$, and,

(iii) $A(w_1, \cdots w_m) = v^{(R)}$.

Further, the iterative secure aggregate is said to be s-privacy preserving for some $s \in (0, 1)$ if

(iv) $\beta_i^{(r)} / \sum_{j=1}^m \beta_j^{(r)} \leq s$ for all $i \in [m]$ and $r \in [R]$.

If we have an iterative secure aggregate with R communication rounds which is also robust, we gain robustness at a R-fold increase in communication cost. Condition (iv) ensures privacy preservation because it reveals only weighted averages with weights at most s, so a user's update is only available after being mixed with those from a large cohort of devices.

The Tension Between Robustness and Heterogeneity. Heterogeneity is a key property of federated learning. The distribution D_i of device i can be quite different from the distribution D_j of some other device j, reflecting the heterogeneous data generated by a diverse set of users.

To analyze the effect of heterogeneity on robustness, consider the simplified scenario of robust mean estimation in Huber's contamination model (Huber, 1964). Here, we wish to estimate the mean $\mu \in \mathbb{R}^d$ given samples $w_1, \cdots, w_m \sim (1-\rho)\mathcal{N}(\mu, \sigma^2 I) + \rho Q$, where Q denotes some outlier distribution that ρ-fraction of the points (designated as outliers) are drawn from. Any aggregate $\bar{w}$ must satisfy the lower bound $\|\bar{w} - \mu\|^2 \geq \Omega\big(\sigma^2 \max\{\rho^2, d/m\}\big)$ with constant probability (Chen et al., 2018b, Theorem 2.2). In the federated learning setting, more heterogeneity corresponds to a greater variance σ^2 among the inlier points, implying a larger error in mean estimation. This suggests a tension between robustness and heterogeneity, where increasing heterogeneity makes robust mean estimation harder in terms of ℓ_2 error.

In this work, we strike a compromise between robustness and heterogeneity by considering a family $\mathcal{D}$ of allowed data distributions such that any device i with $D_i \notin \mathcal{D}$ will be regarded as a corrupted device, i.e., $i \in \mathcal{C}$. We will be able to guarantee convergence up to the degree of heterogeneity in $\mathcal{D}$; we call this $\mathrm{width}(\mathcal{D})$ and make it precise in Section 5.3. In the i.i.d. case, $\mathcal{D}$ is a singleton and $\mathrm{width}(\mathcal{D}) = 0$.

Examples. Next, we consider some examples of update corruption — see (Kairouz et al., 2021b) for a comprehensive treatment. Corrupted updates could be non-adversarial in nature, such as sensor malfunctions or hardware bugs in unreliable and heterogeneous devices (e.g., mobile phones) that are outside the control of the orchestrating server. On the other hand, we could also have adversarial corruptions of the following types:

(a) *Static data poisoning:* The corrupted devices $\mathcal{C}$ are allowed to modify their training data prior to the start of the training and the data is fixed thereafter. Formally, the objective function of device $i \in \mathcal{C}$ is now $\tilde{F}_i(w) = \mathbb{E}_{z \sim \tilde{D}_i}[f(w; z)]$ where $\tilde{D}_i$ has been modified from the original D_i. These devices then participate in the local updates (5.3) with $\nabla \tilde{F}_i$ rather than ∇F_i. We consider device i to contribute corrupted updates only if $\tilde{D}_i \notin \mathcal{D}$ (for instance, $\mathcal{D}$ is the set of natural RGB images).

(b) *Adaptive data poisoning:* The corrupted devices $\mathcal{C}$ are allowed to modify their training data in each round of training depending on the current model $w^{(t)}$. Concretely, the objective function of device

Algorithm 5.1. The RFA Algorithm

Input: Initial iterate $w^{(0)}$, number of communication rounds T, number of clients per round m, number of local updates τ, local step size γ, approximation threshold ε

for $t = 0, 1, \cdots, T-1$ **do**
  Sample m clients from $[n]$ without replacement in S_t
  for each selected client $i \in S_t$ in parallel **do**
    Initialize $w_{i,0}^{(t)} = w^{(t)}$
    for $k = 0, \cdots, \tau - 1$ **do**
      Sample data $z_{i,k}^{(t)} \sim D_i$
      Update $w_{i,k+1}^{(t)} = w_{i,k}^{(t)} - \gamma \nabla f(w_{i,k}^{(t)}; z_{i,k}^{(t)})$
    Set $w_i^{(t+1)} = w_{i,\tau}^{(t)}$
  $w^{(t+1)} = \mathrm{GM}\big((w_i^{(t+1)})_{i \in S_t}, (\alpha_i)_{i \in S_t}, \varepsilon\big)$ (Algo. 5.2)
return w_T

$i \in \mathcal{C}$ in round t is $\tilde{F}_i^{(t)}(w) = \mathbb{E}_{z \sim \tilde{D}_i^{(t)}}[f(w; z)]$ where $\tilde{D}_i^{(t)}$ has been modified from the original D_i using knowledge of $w^{(t)}$. As previously, these devices then participate in the local updates (5.3) with $\nabla \tilde{F}_i^{(t)}$ rather than ∇F_i in round t.

(c) *Update Poisoning:* The corrupted devices can send an arbitrary vector to the server for aggregation, as described by (5.10) in its full generality. This setting subsumes all previous examples as special cases.

The corruption model in (5.10) precludes the *Byzantine setting* (e.g., Kairouz et al., 2021b, Sec. 5.1), which refers to the worst-case model where a corrupted client device $i \in \mathcal{C}$ can behave arbitrarily, such as for instance, changing the weights $\beta_i^{(r)}$ or the vector w_i between each of the rounds of the iterative secure aggregate, as defined in Definition 5.4. It is provably impossible to design a Byzantine-robust iterative secure aggregate in this sense. The examples listed above highlight the importance of robustness to the corruption model under consideration.

Table 5.1 compares the various corruptions in terms of the capability of an adversary required to induce the corruption.

5.3 Robust Aggregation and the RFA Algorithm

In this section, we design a robust aggregation oracle and analyze the convergence of the resulting federated algorithm.

Robust Aggregation with the Geometric Median. The geometric median (GM) of $w_1, \cdots, w_m \in \mathbb{R}^d$ with weights $\alpha_1, \cdots, \alpha_m > 0$ is the minimizer of

$$g(v) := \sum_{i=1}^{m} \alpha_i \|v - w_i\|, \tag{5.12}$$

where $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm. As a robust aggregation oracle, we use an ε-approximate minimizer $\widehat{v}$ of g which satisfies $g(\widehat{v}) - \min_z g(v) \leq \varepsilon$. We denoted it by $\widehat{v} = \mathrm{GM}\left((w_i)_{i=1}^m, (\alpha_i)_{i=1}^m, \varepsilon\right)$. Further,

Algorithm 5.2. The Smoothed Weiszfeld Algorithm

Input: $w_1, \cdots, w_m \in \mathbb{R}^d$ with w_i on device i, $\alpha_1, \cdots, \alpha_m > 0$, $\nu > 0$, budget R, $v^{(0)} \in \mathbb{R}^d$, secure average oracle $\mathcal{A}$

1: **for** $r = 0, 1, \cdots, R-1$ **do**
2: Server broadcasts $v^{(r)}$ to devices $1, \cdots, m$
3: Device i computes $\beta_i^{(r)} = \alpha_i / (\nu \vee \|v^{(r)} - w_i\|)$
4: $v^{(r+1)} \leftarrow \left(\sum_{i=1}^m \beta_i^{(r)} w_i\right) / \sum_{i=1}^m \beta_i^{(r)}$ using $\mathcal{A}$

return $v^{(R)}$

when $\alpha_i = 1/m$, we write $\mathrm{GM}\left((w_i)_{i=1}^m, \varepsilon\right)$. We assume that $w_1, \cdots w_m$ are non-collinear, which is reasonable in the federated setting. Then, g admits a unique minimizer $v^\star$. Further, we assume $\sum_i \alpha_i = 1$ w.l.o.g.[1]

The breakdown point is a popular notion of robustness (Donoho and Huber, 1983). It is the smallest fraction of points that must be changed to get an aggregate to equal an arbitrary point, as we define below.

Definition 5.5 (Breakdown Point)**.** Consider a collection $W = \{w_1, \cdots, w_m\} \subset \mathbb{R}^d$ of m points in $\mathbb{R}^d$. The breakdown point of an estimator $\psi : (\mathbb{R}^d)^m \to \mathbb{R}^d$ is defined as

$$\mathrm{BP}(\psi, W) := \min_{k \in [m]} \left\{ \frac{k}{m} \,:\, \sup\left\{ \|\psi(W) - \psi(W')\| \,:\, W' \subset \mathbb{R}^d,\, |W'| = m,\, |W' \cap W| = m - k \right\} = \infty \right\}.$$

The GM is robust in the sense that it has an optimal breakdown point of 1/2, as we formalize below. In other words, to get the geometric median to equal an arbitrary point, at least half the points (in total weight) must be modified. The result is originally due to Lopuhaa and Rousseeuw (1991, Thm. 2.2).

Property 5.6. *Consider a collection of points $W = \{w_1, \cdots, w_m\} \subset \mathbb{R}^d$ and denote $v^* = \mathrm{GM}(W)$ as its geometric median. For all $v \in \mathbb{R}^d$ and $S \subset [m]$ with $|S| = k < m/2$, we have,*

$$\|v - v^*\| \le 2 \left(\frac{n-k}{n-2k} \right) \max_{i \notin S} \|v - w_i\|.$$

In other words, the breakdown point of the geometric median is $\mathrm{BP}(\mathrm{GM}, W) = 1/2$.

Robust Federated Aggregation (RFA). The RFA algorithm is obtained by replacing the mean aggregation of FedAvg with this GM-based robust aggregation oracle – the full algorithm is given in Algorithm 5.1. Similar to FedAvg, RFA also trades off some communication for local computation by running multiple local steps in line 5. The communication efficiency and privacy preservation of RFA follow from computing the GM as an iterative secure aggregate, which we turn to next. Note that RFA is agnostic to the *actual* level of corruption in the problem and the aggregation is robust regardless of the convexity of the local objectives F_i.

Geometric Median as an Iterative Secure Aggregate. While the GM is a natural robust aggregation oracle, the key challenge in the federated setting is to implement it as an iterative secure aggregate. Our approach, given in Algorithm 5.2, iteratively computes a new weight $\beta_i^{(r)} \propto 1/\|v^{(r)} - w_i\|$, up to a tolerance

[1] One could apply the results to $\widetilde{g}(v) := g(v) / \sum_{i=1}^m \alpha_i$.

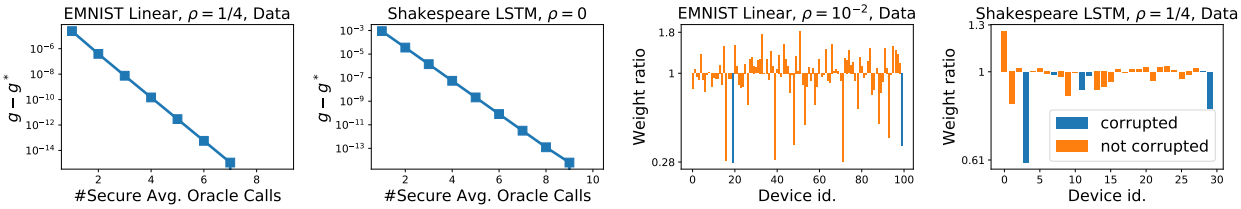


Figure 5.2: Left two: Convergence of the smoothed Weiszfeld algorithm. **Right two**: Visualization of the re-weighting β_i/α_i, where β_i is the weight of w_i in $\mathrm{GM}((w_i),(\alpha_i)) = \sum_i \beta_i w_i$. See Appendix C.3 for details.

$\nu > 0$, whose role is to prevent division by zero. This endows the algorithm with greater stability. We call it the smoothed Weiszfeld algorithm as it is a variation of Weiszfeld's classical algorithm (Weiszfeld, 1937). The smoothed Weiszfeld algorithm satisfies the following convergence guarantee. Its full proof may be found in (Pillutla et al., 2022a, Appendix III).

Proposition 5.7. *The iterate $v^{(R)}$ of Algorithm 5.2 with input $v^{(0)} \in \mathrm{conv}\{w_1, \cdots, w_m\}$ and $\nu > 0$ satisfies*

$$g(v^{(R)}) - g(v^\star) \leq \frac{2\|v^{(0)} - v^\star\|^2}{\overline{\nu} R} + \frac{\nu}{2},$$

where $v^\star = \arg\min g$ and $\overline{\nu} = \min_{r\in[R], i\in[m]} \nu \vee \|v^{(r-1)} - w_i\| \geq \nu$. Furthermore, if $0 < \nu \leq \min_{i=1,\cdots,m} \|v^\star - w_i\|$, then it holds that $g(v^{(R)}) - g(v^\star) \leq 2\|v^{(0)} - v^\star\|^2/\overline{\nu} R$.

For a ε-approximate GM, we set $\nu = O(\varepsilon)$ to get a $O(1/\varepsilon^2)$ rate. However, if the GM $v^\star$ is not too close to any w_i, then the same algorithm automatically enjoys a faster $O(1/\varepsilon)$ rate. The algorithm enjoys plausibly an even faster convergence rate *locally*, and we leave this for future work.

The proof relies on constructing a jointly convex surrogate $G : \mathbb{R}^d \times \mathbb{R}^m_{++} \to \mathbb{R}$ defined using $\eta = (\eta_1, \cdots, \eta_m) \in \mathbb{R}^m$ as

$$G(v, \eta) := \frac{1}{2} \sum_{k=1}^m \alpha_k \left(\frac{\|v - w_k\|^2}{\eta_k} + \eta_k \right).$$

Instead of minimizing $g(v)$ directly using the equality $g(v) = \inf_{\eta>0} G(v, \eta)$, we impose the constraint $\eta_i \geq \nu$ instead to avoid division by small numbers. The following alternating minimization leads to Algorithm 5.2:

$$\eta^{(r)} = \arg\min_{\eta \geq \nu} G(v^{(r)}, \eta), \text{ and, } v^{(r+1)} = \arg\min_{v \in \mathbb{R}^d} G(v, \eta^{(r)}).$$

Numerically, we find in Figure 5.2 that Algorithm 5.2 is rapidly convergent, giving a **high quality solution in 3 iterations**. This ensures that the approximate GM as an iterative secure aggregate provides robustness at a modest $3\times$ increase in communication cost over regular mean aggregation in FedAvg.

Privacy Preservation. While we can compute the geometric median as an iterate secure aggregate, privacy preservation also requires that the effective weights $\beta_i^{(r)} / \sum_j \beta_j^{(r)}$ are bounded away from 1 for each i. We show this holds for m large.

Proposition 5.8. *Consider $\beta^{(r)}, v^{(r)}$ produced by Algorithm 5.2 when given $w_1, \cdots w_m \in \mathbb{R}^d$ with weights $\alpha_i = 1/m$ for each i as inputs. Denote $B = \max_{i,j} \|w_i - w_j\|$ and $\bar{\nu}$ as in Proposition 5.7. Then, we have for all $i \in [m]$ and $r \in [R]$ that*

$$\frac{\beta_i^{(r)}}{\sum_{j=1}^m \beta_j^{(r)}} \leq \frac{B}{B + (m-1)\bar{\nu}} \,.$$

Proof. Since $v^{(r)} \in \mathrm{conv}\{w_1, \cdots, w_m\}$, we have $\bar{\nu} \leq \|v^{(r)} - w_i\| \leq B$. Hence, $\alpha_i/B \leq \beta_i^{(r)} \leq \alpha_i/\bar{\nu}$ for each i and r and the proof follows. □

5.3.1 Convergence Analysis of RFA

We now present a convergence analysis of RFA under two simplifying assumptions. First, we focus on least-squares fitting of additive models, as it allows us to leverage sharp analyses of SGD (Bach and Moulines, 2013; Jain et al., 2017a;b) and focus on the effect of the aggregation. Second, we assume w.l.o.g. that each device is weighted by $\alpha_i = 1/n$ to avoid technicalities of random sums $\sum_{i \in S_t} \alpha_i$. This assumption can be lifted with standard reductions; see Remark 5.10.

Setup. We are interested in the supervised learning setting where $z_i \equiv (x_i, y_i) \sim D_i$ is an input-output pair. We assume that the output y_i satisfies $\mathbb{E}[y_i] = 0$ and $\mathbb{E}[y_i^2] < \infty$. Denote the marginal distribution of input x_i as $D_{X,i}$. The goal is to estimate the regression function $\overline{x} \mapsto \mathbb{E}[y_i | x_i = \overline{x}]$ from a training sequence of independent copies of $(x_i, y_i) \sim D_i$ in each device. The corresponding objective is the square loss minimization

$$F(w) = \frac{1}{n} \sum_{i=1}^n F_i(w) \,, \quad \text{where} \tag{5.13}$$

$$F_i(w) = \frac{1}{2} \mathbb{E}_{(x,y) \sim D_i} \left(y - w^\top \phi(x) \right)^2 \text{ for all } i \in [n] \,. \tag{5.14}$$

Here, $\phi(x) = (\phi_1(x), \ldots, \phi_d(x)) \in \mathbb{R}^d$ where $\phi_1, \ldots, \phi_d$ are a fixed basis of measurable, centered functions. The basis functions may be nonlinear, thus encompassing random feature approximations of kernel feature maps and pre-trained deep network feature representations.

We state our results under the following assumptions: (a) the feature maps are bounded as $\|\phi(x)\| \leq R$ with probability one under $D_{X,i}$ for each device i; (b) each F_i is μ-strongly convex; (c) the additive model is well-specified on each device: for each device i, there exists $w_i^\star \in \mathbb{R}^d$ such that $y_i = \phi(x_i)^\top w_i^\star + \zeta_i$ where $\zeta_i \sim \mathcal{N}(0, \sigma^2)$. The second assumption is equivalent to requiring that $H_i = \nabla^2 F_i(w) = \mathbb{E}_{x \sim D_{X,i}}[\phi(x)\phi(x)^\top]$, the covariance of x on device i, has eigenvalues no smaller than μ.

Quantifying Heterogeneity. We quantify the heterogeneity in the data distributions D_i across devices in terms of the heterogeneity of marginals $D_{X,i}$ and of the conditional expectation $\mathbb{E}[y_i | x_i = x] = \phi(x)^\top w_i^\star$. Let $H = \nabla^2 F(w) = (1/n) \sum_{i=1}^n H_i$ be the covariance of x under the mixture distribution across devices, where H_i is the covariance of x_i in device i. We measure the dissimilarities $\Omega_X, \Omega_{Y|X}$ of the marginal and the conditionals respectively as

$$\Omega_X = \max_{i \in [n]} \lambda_{\max}(H^{-1/2} H_i H^{-1/2}) \,, \tag{5.15}$$

$$\Omega_{Y|X} = \max_{i,j \in [n]} \|w_i^\star - w_j^\star\| \,, \tag{5.16}$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue. Note that $\Omega_X \geq 1$ and it is equal to 1 iff each $H_i = H$. It measures the spectral misalignment between each H_i and H. The second condition is related to the Wasserstein-2 distance (Panaretos and Zemel, 2020) between the conditionals $D_{Y|X,i}$ as $W_2(D_{Y|X,i}, D_{Y|X,j}) \leq R\Omega_{Y|X}$. We define the degree of heterogeneity between the various $D_i = D_{X,i} \otimes D_{Y|X,i}$ as $\mathrm{width}(\mathcal{D}) = \Omega_X \Omega_{Y|X} =: \Omega$. That is, if the conditionals are the same ($\Omega_{Y|X} = 0$), we can tolerate arbitrary heterogeneity in the marginals $D_{X,i}$.

Convergence. We now analyze RFA where the local SGD updates are equipped with "tail-averaging" (Jain et al., 2017b) so that $w_i^{(t+1)} = (2/\tau) \sum_{k=\tau/2}^{\tau} w_{i,k}^{(t)}$ is averaged over the latter half of the trajectory of iterates instead of line 8 of Algorithm 5.1. We show that this variant of RFA converges up to the dissimilarity level $\Omega = \Omega_X \Omega_{Y|X}$ when the corruption level $\rho < 1/2$.

Theorem 5.9. *Consider F defined in* (5.13) *and suppose the corruption level satisfies $\rho < 1/2$. Consider Algorithm 5.1 run for T outer iterations with a learning rate $\gamma = 1/(2R^2)$, and the local updates are run for τ_t steps in outer iteration t with tail averaging. Fix $\delta > 0$ and $\theta \in (\rho, 1/2)$, and set the number of devices per iteration, m as*

$$m \geq \frac{\log(T/\delta)}{2(\theta - \rho)^2}\,. \tag{5.17}$$

Define $C_\theta := (1 - 2\theta)^{-2}$, $w^\star = \arg\min F$, $F^\star = F(w^\star)$, $\kappa := R^2/\mu$ and $\Delta_0 := \|w^{(0)} - w^\star\|^2$. Let $\tau \geq 4\kappa \log(128 C_\theta \kappa)$. We have that the event $\mathcal{E} = \bigcap_{t=0}^{T-1} \{|S_t \cap \mathcal{C}| \leq \theta m\}$ holds with probability at least $1 - \delta$. Further, if $\tau_t = 2^t \tau$ for each iteration t, then the output $w^{(T)}$ of Algorithm 5.1 satisfies,

$$\mathbb{E}\left[\|w^{(T)}) - w^\star\|^2 \,\middle|\, \mathcal{E}\right] \leq \frac{\Delta_0}{2^T} + C C_\theta \left(\frac{d\sigma^2 T}{\mu\tau 2^T} + \frac{\varepsilon^2}{m^2} + \Omega^2\right)$$

where C is a universal constant. If $\tau_t = \tau$ instead, then, the noise term above reads $d\sigma^2/\mu\tau$.

Theorem 5.9 shows near-linear convergence $O(T/2^T)$ up to two error terms in the case that ρ is bounded away from $1/2$ (so that θ and C_θ can be taken to be constants). The increasing local computation $\tau_t = 2^t \tau$ required by this rate is feasible since local computation is assumed to be cheaper than communication.

The first error term is ε^2/m^2 due to approximation ε in the GM, which can be made arbitrarily small by increasing the number m of devices sampled per round. The second error term Ω^2 is due to heterogeneity. Indeed, exact convergence as $T \to \infty$ is not possible in the presence of corruption: lower bounds for robust mean estimation (e.g. Chen et al., 2018b, Theorem 2.2) imply that $\|w^{(T)} - w^\star\|^2 \geq C\rho^2 \Omega_{Y|X}^2$ w.p. at least $1/2$. Consistent with our theory, we find in real heterogeneous datasets in Section 5.4 that RFA can lead to *marginally* worse performance than FedAvg in the corruption-free regime ($\rho = 0$). Finally, while we focus on the setting of least squares, our results can be extended to the general convex case.

Remark 5.10. *For unequal weights, we can perform the reduction $\tilde{F}_i(w) = n\alpha_i F_i(w)$, so the theory applies under the substitution $(R^2, \sigma^2, \mu, \Omega_X) \mapsto (c_1\, R^2, c_1\, \sigma^2, c_2\, \mu, (c_1/c_2)\Omega_X)$, where $c_1 = n \max_i \alpha_i$ and $c_2 = n \min_i \alpha_i$.*

We use the following convergence result of SGD (Jain et al., 2017a, Theorem 1), (Jain et al., 2017b, Corollary 2).

Theorem 5.11 (Jain et al. (2017b;a)). *Consider a F_k from* (5.13). *Then, defining $\kappa := R^2/\mu$, the output $\overline{v}_\tau$ of τ steps of tail-averaged SGD starting from $v_0 \in \mathbb{R}^d$ using learning rate $(2R^2)^{-1}$ satisfies*

$$\mathbb{E}\|\overline{v}_\tau - w^\star\|^2 \leq 2\kappa \exp\left(-\frac{\tau}{4\kappa}\right)\|v_0 - w^\star\|^2 + \frac{8d\sigma^2}{\mu\tau}\,.$$

Proof of Theorem 5.9. Define the event $\mathcal{E}_t = \{|S_t \cap \mathcal{C}| \leq \theta m\}$ so that $\mathcal{E} = \bigcap_{t=0}^{T-1} \mathcal{E}_t$. Hoeffding's inequality gives $\mathbb{P}(\overline{\mathcal{E}_t}) \leq \delta/T$ for each t so that $\mathbb{P}(\overline{\mathcal{E}}) \leq \delta$ using the union bound. Below, let $\mathcal{F}_t$ denote the sigma algebra generated by $w^{(t)}$.

Consider the local updates on an uncorrupted device $i \in S_t \setminus \mathcal{C}$, starting from $w^{(t)}$. Theorem 5.11 gives, upon using $\tau_t \geq \tau \geq 4\kappa \log(128 C_\theta \kappa)$,

$$\mathbb{E}\left[\|w_i^{(t+1)} - w_i^\star\|^2 \,\middle|\, \mathcal{E}, \mathcal{F}_t\right] \leq \frac{1}{64 C_\theta}\|w^{(t)} - w_i^\star\|^2 + \frac{8d\sigma^2}{\mu\tau_t}\,.$$

Note that $w^\star = (1/n)\sum_{j=1}^n H^{-1}H_j w_j^\star$, so that

$$\|w^\star - w_i^\star\| \leq \frac{1}{n}\sum_{j=1}^n \|H^{-1}H_j(w_j^\star - w_i^\star)\| \leq \Omega\,.$$

Using $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we get,

$$\begin{aligned}
\mathbb{E}\Big[\|w_i^{(t+1)} - w^\star\|^2 \,\Big|\, \mathcal{E}, \mathcal{F}_t\Big] &\leq 2\mathbb{E}\left[\|w_i^{(t+1)} - w_i^\star\|^2 \,\middle|\, \mathcal{E}, \mathcal{F}_t\right] + 2\Omega^2 \\
&\leq \frac{1}{32 C_\theta}\|w^{(t)} - w_i^\star\|^2 + \frac{16 d\sigma^2}{\mu\tau_t} + 2\Omega^2 \\
&\leq \frac{q}{16 C_\theta}\|w^{(t)} - w^\star\|^2 + \frac{16 d\sigma^2}{\mu\tau_t} + 4\Omega^2\,.
\end{aligned}$$

We now apply the robustness property of the GM ((Lopuhaa and Rousseeuw, 1991, Thm. 2.2) or (Wu et al., 2020, Lem. 3)) to get,

$$\mathbb{E}\left[\|w^{(t+1)} - w^\star\|^2 \,\middle|\, \mathcal{E}, \mathcal{F}_t\right] \leq \frac{1}{2}\|w^{(t)} - w^\star\|^2 + \frac{128 C_\theta d\sigma^2}{\mu\tau_t} + \Gamma\,,$$

where $\Gamma = 2C_\theta(\varepsilon^2/m^2 + 16\Omega^2)$. Taking an expectation conditioned on $\mathcal{E}$ and unrolling this inequality gives

$$\mathbb{E}\left[\|w^{(T)} - w^\star\|^2 \,\middle|\, \mathcal{E}\right] \leq \frac{\Delta_0}{2^T} + \frac{128 C_\theta d\sigma^2}{\mu}\sum_{t=1}^T \frac{1}{2^{T-t}\tau_t} + 2\Gamma\,.$$

When $\tau_t = 2^t\tau$, the series sums to $2^{-(T-1)}T/\tau$, while for $\tau_t = \tau$, the series is upper bounded by $2/\tau$. □

We now consider RFA in connection with the three factors mentioned in Section 5.1.

(i) **Communication Efficiency**: Similar to FedAvg, RFA performs multiple local updates for each aggregation round, to save on the total communication. However, owing to the trade-off between communication, privacy, and robustness, RFA requires a modest $3\times$ more communication for robustness per aggregation. In the next section, we present a heuristic to reduce this communication cost to one secure average oracle call per aggregation.

Algorithm 5.3. One-step Smoothed Weiszfeld Algorithm

Input: Same as Algorithm 5.2

1: Device i sets $\beta_i = \alpha_i / (\nu \vee \|w_i\|)$
2: **return** $\left(\sum_{i=1}^m \beta_i w_i\right) / \sum_{i=1}^m \beta_i$ using $\mathcal{A}$

Algorithm 5.4. RFA with Personalization

Replace lines 4 to 8 of Algorithm 5.1 with the following:

1: Set $u_{i,0}^{(t)} = u_i^{(t)}$ and $w_{i,0}^{(t)} = w^{(t)}$
2: **for** $k = 0, \cdots, \tau - 1$ **do**
3: $\quad u_{i,k+1}^{(t)} = u_{i,k}^{(t)} - \gamma \nabla f(w^{(t)} + u_{i,k}^{(t)}; z_{i,k}^{(t)})$ with $z_{i,k}^{(t)} \sim D_i$
4: **for** $k = 0, \cdots, \tau - 1$ **do**
5: $\quad w_{i,k+1}^{(t)} = w_{i,k}^{(t)} - \gamma \nabla f(w_{i,k}^{(t)} + u_{i,\tau}^{(t)}; \tilde{z}_{i,k}^{(t)})$ with $\tilde{z}_{i,k}^{(t)} \sim D_i$
6: Set $w_i^{(t+1)} = w_{i,\tau}^{(t)}$ and $u_i^{(t+1)} = u_{i,\tau}^{(t)}$

(ii) **Privacy Preservation**: Algorithm 5.2 computes the aggregation as an iterative secure aggregate. This means that the server only learns the intermediate parameters after being averaged over all the devices, with effective weights bounded away from 1 (Proposition 5.8). The noisy parameter vectors sent by individual devices are uniformly uninformative in information theoretic sense with the use of secure multi-party computation.

(iii) **Robustness**: The geometric median has a breakdown point of 1/2 (Lopuhaa and Rousseeuw, 1991, Theorem 2.2), which is the highest possible (Lopuhaa and Rousseeuw, 1991, Theorem 2.1). In the federated learning context, this means that convergence is still guaranteed by Theorem 5.9 when up to half the points in terms of total weight are corrupted. RFA is resistant to both data or update poisoning while being privacy-preserving. On the other hand, FedAvg has a breakdown point of 0, where a single corruption in each round can cause the model to become arbitrarily bad.

5.3.2 Extensions to RFA

We now discuss two extensions to RFA to reduce the communication cost (without sacrificing privacy) and better accommodate statistical heterogeneity in the data with model personalization.

One-step RFA: Reducing the Communication Cost. Recall that RFA results in a 3-5$\times$ increase in the communication cost over FedAvg. Here, we give a heuristic variant of RFA in an extremely communication-constrained setting, where it is infeasible to run multiple iterations of Algorithm 5.2. We simply run Algorithm 5.2 with $v^{(0)} = 0$ and a communication budget of $R = 1$; see Algorithm 5.3 for details. We find in Section 5.4.3 that one-step RFA retains most of the robustness of RFA.

Personalized RFA: Offsetting Heterogeneity. We now show RFA can be extended to better handle heterogeneity in the devices with the use of personalization. The key idea is that predictions are made on device i by summing the shared parameters w maintained by the server with personalized parameters $U = \{u_1, \cdots, u_n\}$ maintained individually on-device. In particular, the optimization problem we are

Table 5.2: Dataset description and statistics.

Dataset	Task	#Classes	#Train	#Test	#Devices	#Train per Device		
						Median	Max	Min
EMNIST	Image Classification	62	$204K$	$23K$	1000	160	418	92
Shakespeare	Character-level Language Modeling	53	$2.2M$	$0.25M$	628	1170	70600	90
Sent140	Sentiment Analysis	2	$57K$	$15K$	877	55	479	40

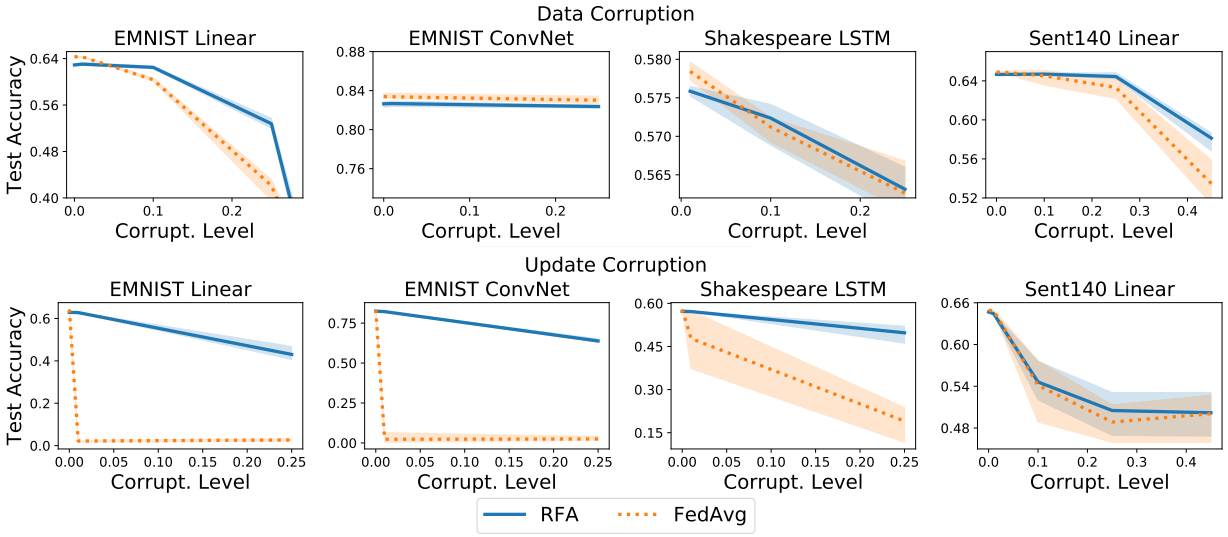


Figure 5.3: Comparison of robustness of RFA and FedAvg under data corruption (**top**) and update corruption (**bottom**). The left three plots for update corruption show omniscient corruption while the rightmost one shows Gaussian corruption. The shaded area denotes minimum and maximum over 5 random seeds.

interested in solving is

$$\min_{w,U} \left[F(w, U) := \sum_{i=1}^{n} \alpha_k \, \mathbb{E}_{z \sim D_i} \left[f(w + u_i; z) \right] \right].$$

We outline the algorithm in Algorithm 5.4. We train the shared and personalized parameters on each other's residuals, following the residual learning scheme of (Agarwal et al., 2020). Each selected device first updates its personalized parameters u_i while keeping the shared parameters w fixed. Next, the updates to the shared parameter are computed on the residual of the personalized parameters. The updates to the shared parameter are aggregated with the geometric median, identical to RFA. Experiments in Section 5.4.3 show that personalization is effective in combating heterogeneity.

5.4 Experiments

We now conduct simulations to compare RFA with other federated learning algorithms. The simulations were run using TensorFlow and the data was preprocessed using LEAF (Caldas et al., 2018). We first describe the experimental setup in Section 5.4.1, then study the robustness and convergence of RFA in

Section 5.4.2. We study the effect of the extensions of RFA in Section 5.4.3. The full details from this section and more simulation results are given in Appendix C.1. The code and scripts to reproduce these experiments can be found online (rfa, 2019a).

5.4.1 Setup

We consider three machine learning tasks. The datasets are described in Table 5.2. As described in Section 5.1, we take the weight α_i of device i to be proportional to the number of datapoints N_i on the device.

(a) *Character Recognition*: We use the EMNIST dataset (Cohen et al., 2017), where the input x is a 28×28 grayscale image of a handwritten character and the output y is its identification (0-9, a-z, A-Z). Each device is a writer of the handwritten character x. We use two models — a linear model $\varphi(x; w) = w^\top x$ and a convolutional neural network (ConvNet). We use as objective $f(w; (x, y)) = \ell(y, \varphi(x; w))$, where ℓ is the multinomial logistic loss ℓ. We evaluate performance using the classification accuracy.

(b) *Character-Level Language Modeling*: We learn a character-level language model over the Complete Works of Shakespeare (Shakespeare). We formulate it as a multiclass classification problem, where the input x is a window of 20 characters and the output y is the next (i.e., 21st) character. Each device is a role from a play (e.g., Brutus from The Tragedy of Julius Caesar). We use a long-short term memory model (LSTM) (Hochreiter and Schmidhuber, 1997) together with the multinomial logistic loss. The performance is evaluated with the classification accuracy of next-character prediction.

(c) *Sentiment Analysis*: We use the Sent140 dataset (Go et al., 2009) where the input x is a tweet and the output $y = \pm 1$ is its sentiment. Each device is a distinct Twitter user. We use a linear model using an average of the GloVe embeddings (Pennington et al., 2014) of the words of the tweet. It is trained with the binary logistic loss and evaluated with the classification accuracy.

Corruption Models. We consider the following corruption models for corrupted devices $\mathcal{C}$, cf. Section 5.2:

(a) *Data Poisoning*: The distribution D_i on a device $k \in \mathcal{C}$ is replaced by some fixed $\tilde{D}_i$. For EMNIST, we take the negative of an image so that $\tilde{D}_i(x, y) = D_i(1 - x, y)$. For the Shakespeare dataset, we reverse the text so that $\tilde{D}_i(c_1, \cdots c_{20}, c_{21}) = D_i(c_{21}, \cdots c_2, c_1)$. In both these cases, the labels are unchanged. For the Sent140 dataset, we flip the label while keeping x unchanged.

(b) *Update poisoning with Gaussian corruption*: Each corrupted device $i \in \mathcal{C}$ returns $w_i^{(t+1)} = w_{i,\tau}^{(t)} + \zeta_i^{(t)}$, where $\zeta_i^{(t)} \sim \mathcal{N}(0, \sigma^2 I)$, where σ^2 is the variance across the components of $w_{i,\tau}^{(t)} - w^{(t)}$.[2]

(c) *Update poisoning with omniscient corruption*: The parameters $w_i^{(t+1)}$ returned by devices $i \in \mathcal{C}$ are modified so that the weighted arithmetic mean $\sum_{i \in S_t} \alpha_i w_i^{(t+1)}$ over the selected devices S_t is set to $-\sum_{i \in S_t} \alpha_i w_{i,\tau}^{(t)}$, the negative of what it would have been without the corruption. This is designed to hurt the weighted arithmetic mean aggregation.

Hyperparameters. The hyperparameters are chosen similar to the defaults of (McMahan et al., 2017). A learning rate schedule was tuned on a validation set for FedAvg with no corruption. The same schedule was used for RFA. The aggregation in RFA is implemented using the smoothed Weiszfeld algorithm with a budget of $R = 3$ calls to the secure average oracle, thanks to its rapid empirical convergence (cf. Figure 5.2), and $\nu = 10^{-6}$ for numerical stability. Each simulation was repeated 5 times and the shaded area denotes the minimum and maximum over these runs. We give further details and sensitivity analyses in Chapter C.

[2]Model updates $w_i^{(t)} - w^{(t)}$ are aggregated, not the models $w_i^{(t)}$ directly (Kairouz et al., 2021b).

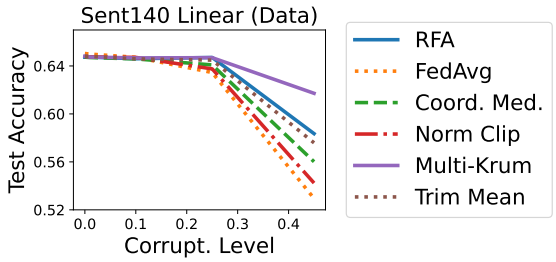


Figure 5.4: Comparison of RFA with other robust aggregation algorithms on Sent140 with data corruption.

5.4.2 Robustness and Convergence of RFA

First, we compare the robustness of RFA as opposed to vanilla FedAvg to different types of corruption across different datasets in Figure 5.3. We make the following observations.

RFA gives improved robustness to linear models with data corruption. For instance, consider the EMNIST linear model at $\rho = 1/4$. RFA achieves 52.8% accuracy, over 10% better than FedAvg at 41.2%.

RFA performs similarly to FedAvg in deep nets with data corruption. RFA and FedAvg are within one standard deviation of each other for the Shakespeare LSTM model, and nearly equal for the EMNIST ConvNet model. We note that the behavior of the training of a neural network when the data is corrupted is not well-understood in general (e.g., Zhang et al., 2017).

RFA gives improved robustness to omniscient corruptions for all models. For the omniscient corruption, the test accuracy of the FedAvg is close to 0% for the EMNIST linear model and ConvNet, while RFA still achieves over 40% at $\rho = 1/4$ for the former and well over 60% for the latter. A similar trend holds for the Shakespeare LSTM model.

RFA almost matches FedAvg in the absence of corruption. Recall from Section 5.2 that robustness comes at the cost of heterogeneity; this is also reflected in the theory of Section 5.3. Empirically, we find that the performance hit of RFA due to heterogeneity is quite small: 1.4% for the EMNIST linear model (64.3% vs. 62.9%), under 0.4% for the Shakespeare LSTM, and 0.3% for Sent140 (65.0% vs. 64.7%). Further, we demonstrate in Section C.2 that, consistent with the theory, this gap completely vanishes in the i.i.d. case.

RFA is competitive with other robust aggregation schemes while being privacy-preserving. We now compare RFA with: (a) coordinate-wise median (Yin et al., 2018) and ℓ_2 norm clipping (Sun et al., 2019) which are agnostic to the actual corruption level ρ like RFA, and, (b) trimmed mean (Yin et al., 2018) and multi-Krum (Blanchard et al., 2017), that require exact knowledge of the level of corruption ρ in the problem. We find that RFA is more robust than the two agnostic algorithms coordinate-wise median and norm clipping. Perhaps surprisingly, RFA is also more robust than the trimmed mean which uses perfect knowledge of the corruption level ρ. We note that multi-Krum is more robust than RFA. That being said, RFA has the advantage that it is fully agnostic to the actual corruption level ρ and is privacy-preserving, while the other robust approaches are not.

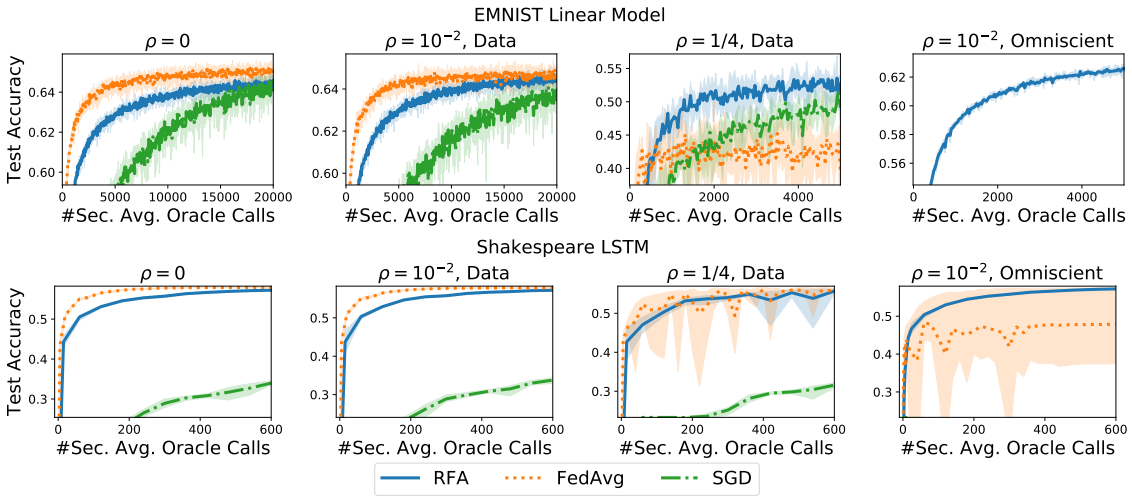


Figure 5.5: Comparison of methods plotted against the number of calls to a secure average oracle for different corruption settings. For the case of omniscient corruption, FedAvg and SGD are not shown in the plot if they diverge. The shaded area denotes the maximum and minimum over 5 random seeds.

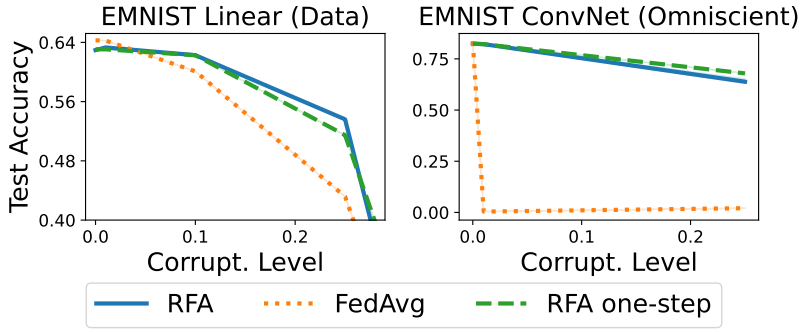


Figure 5.6: Robustness of one-step RFA.

Summary: robustness of RFA. Overall, we find that RFA is no worse than FedAvg in the presence of corruption and is often better while being almost as good in the absence of corruption. Furthermore, RFA degrades more gracefully as the corruption level increases.

RFA requires only $3\times$ the communication of FedAvg. Next, we plot in Figure 5.5 the performance versus the number of rounds of communication as measured by the number of calls to the secure average oracle. We note that in the low corruption regime of $\rho = 0$ or $\rho = 10^{-2}$ under data corruption, RFA requires $3\times$ the number of calls to the secure average oracle to reach the same performance. However, it matches the performance of FedAvg when measured in terms of the number of outer iterations, with the additional communication cost coming from multiple Weiszfeld iterations for computation of the average.

RFA exhibits more stable convergence under corruption. We also see from Figure 5.5 ($\rho = 1/4$, Data) that the variability of accuracy across random runs, denoted here by the shaded region, is much smaller for RFA. Indeed, by being robust to the corrupted updates sent by random sampling of corrupted clients, RFA exhibits a more stable convergence across iterations.

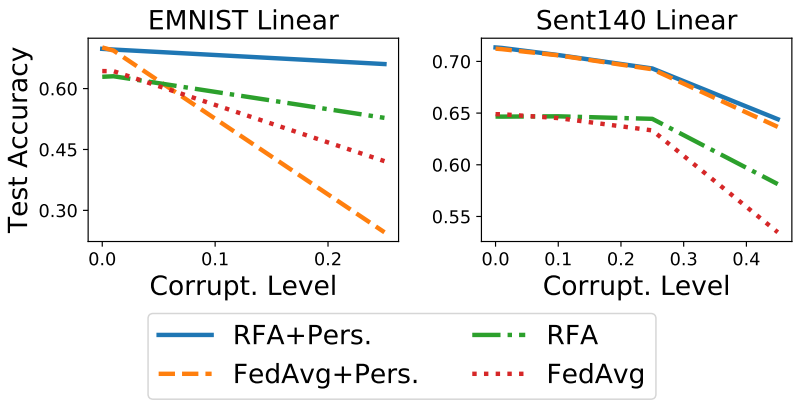


Figure 5.7: Effect of personalization on the robustness of RFA and FedAvg under data corruption.

5.4.3 Extensions of RFA

We now study the proposed extensions: one-step RFA and personalization.

One-step RFA gives most of the robustness with no extra communication. From Figure 5.6, we observe that for one-step RFA is quite close in performance to RFA across different levels of corruption for both data corruption on an EMNIST linear model and omniscient corruption on an EMNIST ConvNet. For instance, in the former, one-step RFA gets 51.4% in accuracy, which is 10% better than FedAvg while being almost as good as full RFA (52.8%) at $\rho = 0.25$. Moreover, for the latter, we find that one-step RFA (67.9%) actually achieves higher test accuracy than full RFA (63.0%) at $\rho = 0.25$.

Personalization helps RFA offset effects of heterogeneity. Figure 5.7 plots the effect of RFA with personalization. First, we observe that personalization leads to an improvement with no corruption for both FedAvg and RFA. For the EMNIST linear model, we get 70.1% and 69.9% respectively from 64.3% and 62.9%. Second, we observe that RFA exhibits greater robustness to corruption with personalization. At $\rho = 1/4$ with the EMNIST linear model, RFA with personalization gives 66.4% (a reduction of 3.4%) while no personalization gives 52.8% (a reduction of 10.1%). The results for Sent140 are similar, with the exception that FedAvg with personalization is nearly identical to RFA with personalization.

5.5 Related Work

We now survey some related work.

Federated Learning was introduced in (McMahan et al., 2017) as a distributed optimization approach to handling on-device machine learning, with secure multi-party averaging algorithms given in (Bonawitz et al., 2017; Balle et al., 2020). Extensions were proposed in (Smith et al., 2017b; Mohri et al., 2019; Li et al., 2020c; Karimireddy et al., 2020; Dinh et al., 2020; Fallah et al., 2020; Laguel et al., 2021a; Avdiukhin and Kasiviswanathan, 2021; Reddi et al., 2021); see also the recent surveys (Li et al., 2020a; Kairouz et al., 2021b). We address robustness to corrupted updates, which is broadly applicable in these settings.

Distributed optimization has a long history (Bertsekas and Tsitsiklis, 1989). Recent work includes primal-dual frameworks (Smith et al., 2018; Ma et al., 2017) and variants suited to decentralized (He et al., 2018), and asynchronous (Leblond et al., 2018) settings.

From the lens of *learning in networks* (Sayed, 2014), federated learning comprises a star network where agents (i.e., devices) with private data are connected to a server with no data, which orchestrates cooperative learning. Further, for privacy, model updates from individual agents cannot be shared directly, but must be aggregated securely.

Robust estimation was pioneered by Huber (Huber, 1964; 2011). Robust median-of-means were introduced in (Nemirovski and Yudin, 1983), with follow-ups in (Minsker, 2015; Hsu and Sabato, 2016; Lugosi and Mendelson, 2019a; Lecué and Lerasle, 2020; Lugosi and Mendelson, 2019b). Robust mean estimation, in particular, received much attention (Diakonikolas et al., 2016; Minsker, 2018; Cheng et al., 2019). Robust estimation in networks was considered in (Al-Sayed et al., 2017; Yu et al., 2019; Chen et al., 2019). These works consider the statistics of robust estimation in the i.i.d. case, while we focus on distributed optimization with privacy preservation.

Byzantine robustness, resilience to arbitrary behavior of some devices (Lamport et al., 1982), was studied in distributed optimization with gradient aggregation (Blanchard et al., 2017; Chen et al., 2017; 2018a; Yin et al., 2018; Alistarh et al., 2018; Cao and Lai, 2019). Byzantine robustness of federated learning is a priori not possible without additional assumptions because the secure multi-party computation protocols require faithful participation of the devices. Thus, we consider a more nuanced and less adversarial corruption model Section 5.2; here, devices participate faithfully in the aggregation loop. Further, it is unclear how to securely implement the nonlinear aggregation algorithms of these works. Lastly, the use of, e.g., secure enclaves (Subramanyan et al., 2017) in conjunction with our approach could guarantee Byzantine robustness in federated learning. We aggregate *model parameters* in a robust manner, which is more suited to the federated setting. We note that (Li et al., 2019) also aggregate model parameters rather than gradients by framing the problem in terms of consensus optimization. However, their algorithm requires devices to be always available and participate in multiple rounds, which is not practical in the federated setting (Kairouz et al., 2021b).

Weiszfeld's algorithm (Weiszfeld, 1937) to *compute the geometric median*, has received much attention (Kuhn, 1973; Vardi and Zhang, 2001; Beck and Sabach, 2015). The Weiszfeld algorithm is also known to exhibit asymptotic linear convergence (Katz, 1974). However, unlike these variants, ours is numerically stable. A theoretical proposal of a near-linear time algorithm for the geometric median was recently explored in (Cohen et al., 2016).

Frameworks to guarantee *privacy* of user data include differential privacy (Dwork et al., 2006b; Kairouz et al., 2021a) and homomorphic encryption (Gentry, 2010). These directions are orthogonal to ours, and could be used in conjunction. See (Bonawitz et al., 2017; Li et al., 2020a; Kairouz et al., 2021b) for a broader discussion.

5.6 Discussion and Conclusion

We presented a robust aggregation approach, based on the geometric median and the smoothed Weiszfeld algorithm to efficiently compute it, to make federated learning more robust to settings where a fraction of the devices may be sending corrupted updates to the orchestrating server. The robust aggregation oracle preserves the privacy of participating devices, operating with calls to secure multi-party computation primitives enjoying privacy preservation theoretical guarantees. RFA is available in several variants, including a fast one with a single step of robust aggregation and one adjusting to heterogeneity with on-device personalization. All variants are readily scalable while preserving privacy, building off secure multi-party computation primitives already used at a planetary scale.

The bound we proved for RFA in Theorem 5.9 is not the best possible. Lower bounds for robust mean

estimation (e.g. Chen et al., 2018b, Theorem 2.2) imply that the error $\|w^{(T)} - w^\star\|^2 \gtrsim \rho^2$ scales as the square of corruption level ρ. However, Theorem 5.9 has a non-zero additive error term independent of ρ. Since it is impractical for an adversary to control even a constant fraction of clients in a production system at a global scale, it is desirable to prove tight bounds under the low corruption regime of $\rho \to 0$. Another interesting open problem lies in designing robust aggregation approaches that obtain tight rates in theory and satisfy the key requirements of federated learning (Section 5.1.1).

While we focused on corrupted updates that aim to degrade the quality of the global model for all clients. There is another family of corruptions, known as backdoor attacks, whose goal is to corrupt the performance of the trained model on specific sub-tasks (Bhagoji et al., 2019; Bagdasaryan et al., 2020). Backdoor attacks essentially leverage the fact that most deep networks are overparameterized to overcome robust aggregation. In fact, recent evidence suggests that it is unlikely to have a federated learning method that is certifiably robust to backdoors (Wang et al., 2020a). An open problem in this space is to identify a subset of backdoor attacks for which one can develop certifiably robust federated learning methods, and to ensure the developed method satisfies the key requirements of federated learning (Section 5.1.1).

Recent works have studied personalization in federated learning for its ability to better handle heterogeneity (Agarwal et al., 2020; Dinh et al., 2020; Pillutla et al., 2022b). In this chapter, we have combined robustness with personalization. Recent work suggests that personalization without explicit robust aggregation can also retain robustness properties (Li et al., 2021b). Precisely characterizing the robustness benefits of personalization is an interesting open problem.

Chapter 6

$\triangle$-FL: A Superquantile Optimization Approach for Federated Learning with Heterogeneous Devices

We previously studied the robustness of federated learning to corrupted updates in Chapter 5. In this chapter, we turn to another key feature of federated learning, namely statistical heterogeneity. The client data distributions are *not* identically distributed (Kairouz et al., 2021b; Li et al., 2020a). In typical cross-device federated learning scenarios, each client corresponds to a user. The diversity in the data they generate reflects the diversity in their unique personal, cultural, regional, and geographical characteristics.

This data heterogeneity in federated learning manifests itself as a train-test distributional shift. Indeed, the usual approach minimizes the prediction error of the model on average over the population of clients available for training (McMahan et al., 2017), while at test time, the same model is deployed on individual clients. This approach can be liable to fail on clients whose data distribution is far from most of the population or who may have less data than most of the population. It is highly desirable, therefore, to have a federated learning method that can robustly deliver good predictive performance across a wide variety of natural distribution shifts posed by individual clients.

We present in this chapter a robust approach to federated learning that guarantees a minimum level of predictive performance to all clients even in situations where the population is heterogeneous. The approach we develop addresses these issues by minimizing a learning objective based on the notion of a superquantile (Rockafellar and Uryasev, 2002; Rockafellar et al., 2008), a risk measure that captures the tail behavior of a random variable.

Training models with a learning objective involving the superquantile raises challenges. The superquantile is a non-smooth functional with sophisticated properties. Furthermore, the superquantile function can be seen as a kind of nonlinear expectation that we would like to blend well with averaging mechanisms. We show how to address the former by leveraging the dual formulation and the latter by leveraging the tail-domain viewpoint. As a result, we can obtain an algorithm that can be implemented in a similar way to FedAvg (McMahan et al., 2017) yet offers important benefits to heterogeneous populations.

The approach we propose, Δ-FL, allows one to control higher percentiles of the distribution of errors over the heterogeneous population of clients. We show in the experiments that our approach is more efficient than a direct approach simply seeking to minimize the worst error over the population of clients. Compared to FedAvg, Δ-FL delivers improved prediction to data-poor or non-conforming clients. We present finite time theoretical convergence guarantees for the Δ-FL algorithm when used to train additive

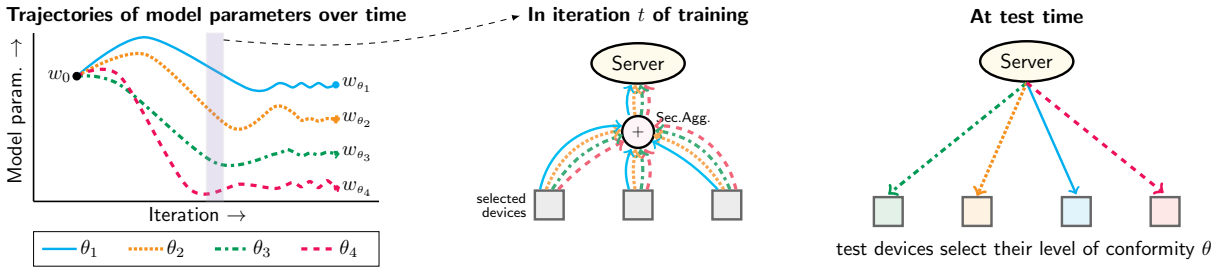


Figure 6.1: Schematic summary of the Δ-FL framework. **Left**: The server maintains multiple models w_{θ_j}, one for each level of conformity θ_j. **Middle**: During training, selected clients participate in training *each* model w_{θ_j}. Individual updates are securely aggregated to update the server model. **Right**: Each test user is allowed to select their level of conformity θ, and are served the corresponding model w_θ.

models or deep networks and show how to implement it in a way that is compatible and modular with secure aggregation and distributed differential privacy.

Contributions. We make the following concrete contributions in this chapter.

- *The Δ-FL Framework*: We introduce the Δ-FL framework, summarized in Figure 6.1, which seeks to guarantee a minimal level of predictive performance on nonconforming clients. The framework relies on a nonsmooth superquantile-based objective to minimize the tail statistics of the prediction errors on the client data distributions. The objective is parameterized by the conformity level, which is a scalar summary of how closely a client conforms to the population.
- *Optimization Algorithm, Convergence, and Privacy Analysis*: To optimize the Δ-FL objective, we present a federated optimization algorithm that interleaves differentially private client reweighting steps with federated averaging steps. We establish bounds on its rate of convergence in the convex and nonconvex cases. Further, we provided an analysis of the differential privacy of the proposed algorithm.
- *Numerical Experiments*: We perform numerical experiments using neural networks and linear models, on tasks including image classification, and sentiment analysis based on public datasets. The experiments demonstrate the superior performance of Δ-FL over state-of-the-art baselines on the upper quantiles of the error on test clients, with particular improvements on data-poor clients, while being competitive on the mean error.

Outline. We start with some background in Section 6.1. Section 6.2 describes the general setup, recalls the FedAvg algorithm for federated learning, and formalizes the notions of conformity and heterogeneity. Section 6.3 presents a federated optimization algorithm for Δ-FL. We analyze its convergence in the convex and non-convex cases, as well as its differential privacy properties in Section 6.4. We discuss an extension to other risk measures and relations to fair allocation in Section 6.5, and review some related work in Section 6.6. Section 6.7 presents experimental results, comparing the proposed approach to existing ones, on benchmark datasets for federated learning. Detailed proofs and additional details are deferred to Chapter D. The code and the scripts to reproduce results are made publicly available at `https://github.com/krishnap25/simplicial-fl`.

Notation. The norm $\|\cdot\|$ denote the Euclidean norm $\|\cdot\|_2$ in $\mathbb{R}^d$. We use $\Delta^{n-1} = \left\{\pi \in \mathbb{R}^n_+ \,:\, \sum_{i=1}^n \pi_i = 1\right\}$

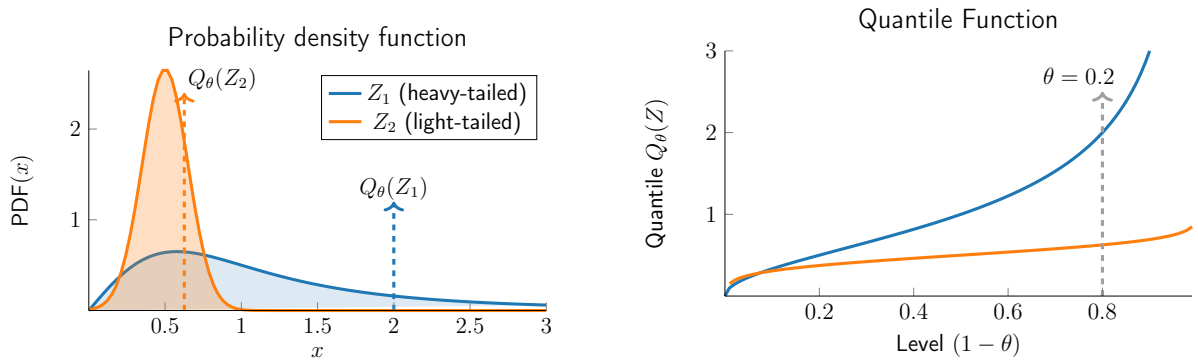


Figure 6.2: Left: The probability density functions of a heavy-tailed log-logistic random variable Z_1, and a light-tailed Gaussian random variable Z_2. While both have a mean of 0.5, the heavy-tailed distribution has larger quantiles. The dotted lines show the 80^{th} percentile (i.e., $\theta = 0.2$) for each distribution. **Right**: The corresponding quantile functions.

to denote the probability simplex in $\mathbb{R}^n$.

6.1 Background

Central to our developments in this chapter is the superquantile (Rockafellar and Uryasev, 2000). It is a measure of the tail of a random variable and has also been known variously as the conditional value at risk (CVaR), tail value at risk, mean excess loss, and mean shortfall. We review the motivation, definition, and basic properties of the superquantile here. For further details and proofs of these properties, we refer to (Laguel et al., 2021b).

Superquantiles and Sensitivity to Extreme Outcomes. Consider real-valued random variables Z_1 and Z_2 that denote an outcome, such that larger values denote more extreme outcomes. The loss of a model on a randomly sampled datapoint is such an example, where large values represent poor predictions. The mean of a random variable does not capture extreme outcomes. For instance, Z_1 and Z_2 can have the same mean but exhibit vastly different tail behaviors, as shown in Figure 6.2 (left). In a number of applications, such as the learning setting, it might be preferable to be *risk averse*, i.e., prefer to avoid extremely large loss values over the potential for a smaller-than-average loss. Indeed, extreme losses could potentially lead to catastrophic failures of learning systems, as we discussed in Chapter 1.

A risk-averse scalar summary of a random variable would quantify its "worse-than-average-case" behavior, i.e., give more importance to the behavior of the right tail over the left tail. The upper quantiles of a random variable measure its tail behavior and extreme outcomes. Formally, for $\theta \in (0, 1)$, the $(1-\theta)$-quantile of Z, denoted by $Q_\theta(Z)$, is the inverse of the cumulative distribution function of Z:

$$Q_\theta(Z) \leq t \iff \mathbb{P}(Z \leq t) \geq 1 - \theta\,, \tag{6.1}$$

for all $t \in \mathbb{R}$. When $\theta = 1/2$, this corresponds to the median value of the random variable Z. For $\theta > 1/2$, it quantifies the right tail of the random variable Z — see Figure 6.2 (right) for an example.

Unfortunately, the quantile functions are difficult to work with, as they are piecewise constant for discrete random variables. A better-behaved measure of the tail behavior of a random variable is its *su-*

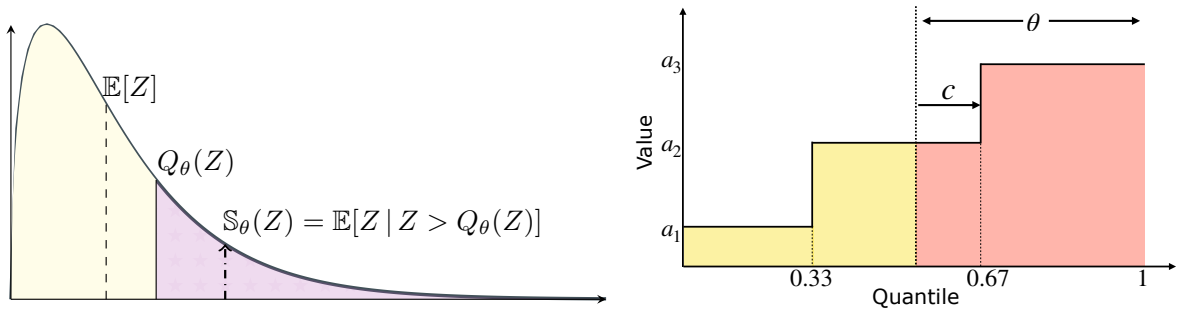


Figure 6.3: Left: $(1-\theta)$-quantile $Q_\theta(Z)$ and superquantile $\mathbb{S}_\theta(Z)$ of a continuous r.v. Z. **Right**: The quantile function of the empirical measure $Z_3 = (1/3)(\delta_{a_1} + \delta_{a_2} + \delta_{a_3})$ over reals $a_1 < a_2 < a_3$. The figure also depicts the distance c to the next discontinuity in the quantile function, as defined in (6.4).

perquantile, which is defined as an average of all quantiles larger than the $(1-\theta)$ quantile. Formally, for any $\theta \in (0, 1]$, the $(1-\theta)$-superquantile of Z is given by

$$\mathbb{S}_\theta(Z) = \frac{1}{\theta} \int_0^\theta Q_{\theta'}(Z) \, \mathrm{d}\theta' \,. \tag{6.2}$$

When $\theta = 1$, the superquantile simply reduces to the expected value: $\mathbb{S}_1(Z) = \mathbb{E}[Z]$. On the other hand, $\lim_{\theta \to 0} \mathbb{S}_\theta(Z)$ is the essential supremum of Z, i.e., Z is almost surely smaller than this quantity. If Z is a continuous random variable (i.e., its cumulative distribution function $t \mapsto \mathbb{P}(Z \leq t)$ is continuous), the superquantile is simply its tail expectation $\mathbb{S}_\theta(Z) = \mathbb{E}[Z \mid Z > Q_\theta(Z)]$; this is illustrated in Figure 6.3 (left). The superquantile is thus a natural measure of the worse-than-average-case behavior of a random variable.

More generally, the following variational form is applicable to both discrete and continuous random variables (Rockafellar and Uryasev, 2000)

$$\mathbb{S}_\theta(Z) = \min_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{\theta} \mathbb{E}\left[\max\{0, Z - \eta\}\right] \right\} .$$

In this expression, the quantile $Q_\theta(Z)$ is obtained as the left end-point of the argmin over η.

Learning with Superquantiles. We now formalize the learning setting alluded to previously. Consider a a loss function f such that $f(w; z)$ gives the loss of w on data z, and a data distribution p over the data. Given samples $z_1, \cdots, z_n \sim p$, the usual approach aims to minimize the population loss $F(w) = \mathbb{E}_{z \sim p}\left[f(w; z)\right]$. Given a new sample $z \sim p$, a minimizer of the population loss F guarantees small loss on average, but it is not risk averse. To protect against extreme outcomes such as catastrophic failures, one might choose to minimize the superquantile of the losses instead: $F_\theta(w) = [\mathbb{S}_\theta]_{z \sim p}\left[f(w; z)\right]$. The goal of this chapter is to instantiate and operationalize this approach to federated learning.

In practice, one does not have access to the data generating distribution p. Rather, we only have access to samples $z_1, \cdots, z_n$ from p. In the risk-neutral setting, the empirical risk minimization approach uses the empirical distribution $p_n = (1/n) \sum_{i=1}^n \delta_{z_i}$ over the samples; here, δ_a denotes a point mass at a. Likewise, we consider the empirical counterpart to the superquantile F_θ as $[\mathbb{S}_\theta]_{z \sim p_n}\left[f(w; z)\right]$.

Superquantile of Empirical Measures. We now review some of the properties of the superquantile of empirical measures. Let Z_n denote the uniform distribution over a finite set $\{a_1, \cdots, a_n\} \subset \mathbb{R}$ of size n. In this case, (6.1) can be written as

$$\mathbb{S}_\theta(a_1, \cdots, a_n) := \mathbb{S}_\theta(Z_n) = \frac{1}{n\theta} \sum_{i \in I_>} a_i + \frac{c}{\theta} Q_\theta(Z_n)\,, \quad \text{where} \quad I_> = \{i \,:\, a_i > Q_\theta(Z_n)\}\,. \tag{6.3}$$

This expression involves the distance c from θ to the next discontinuity point of the quantile function $\theta' \mapsto Q_{\theta'}(Z_n)$

$$c = \theta - \mathbb{P}(Z_n > Q_\theta(Z_n)) = \theta - \frac{|I_>|}{n}\,. \tag{6.4}$$

See Figure 6.3 (right) for an illustration. This gives a recipe for computing the superquantile similar to the tail mean for continuous random variables — we first compute the corresponding quantile in $O(n)$ time, followed by averaging values greater than the quantile as per (6.3).

The empirical superquantile $(a_1, \cdots, a_n) \mapsto \mathbb{S}_\theta(a_1, \cdots, a_n)$ is a closed, convex function with full domain $\mathbb{R}^n$ for each $\theta \in (0, 1]$. Its (convex) subdifferential is given by the expression

$$\partial\, \mathbb{S}_\theta(a_1, \cdots, a_n) = \left(\frac{1}{n\theta} \sum_{i \in I_>} e_i \right) + \frac{c}{\theta} \operatorname{conv}\{e_i \,:\, i \in I_=\}\,,$$

where e_i is the i^{th} canonical basis vector, $I_= = \{i \,:\, a_i = Q_\theta(Z_n)\}$ is the set of indices i such that a_i equals the $(1-\theta)$-quantile, $I_>$ is as given in (6.3) and c in (6.4) and Figure 6.3 (right). From this expression, we see that the empirical superquantile is differentiable if $I_=$ is a singleton.

6.2 Problem Setup

We work with the same federated learning setup as in Chapter 5. We review the notation here and refer to Section 5.1 for details.

Let the vector $w \in \mathbb{R}^d$ denote the d model parameters. We assume that each client has a distribution q over some data space such that the data on the client is sampled i.i.d. from q. The loss incurred by the model $w \in \mathbb{R}^d$ on this client is $F(w; q) := \mathbb{E}_{z \sim q}[f(w; z)]$, For a given distribution q, smaller values of $F(\cdot; q)$ denote a better fit of the model to the data. There are n clients available for training. We number these clients as $1, \ldots, n$ and denote the distribution on training client i by q_i. We denote the loss on client i by $F_i(w) := F(w; q_i)$.

The goal of federated learning is to train a model w so that it achieves good performance when deployed on *each* test client, including those which are unseen during training. The usual learning objective in federated learning, which we refer to as the *vanilla FL* objective, is

$$\min_{w \in \mathbb{R}^d} \left[F(w) := \frac{1}{n} \sum_{i=1}^{n} F_i(w) + \frac{\lambda}{2} \|w\|_2^2 \right]\,. \tag{6.5}$$

This is a special case of (6.5) with equal weights $\alpha_i = 1/n$ with added regularization with parameter λ. Owing to the statistical heterogeneity in federated learning, the distribution p of a specific test client could be different from the average distribution $(1/n) \sum_{i=1}^n q_i$ that the model is trained on. However, the vanilla FL objective places a limit on how well statistical heterogeneity can be addressed. By minimizing

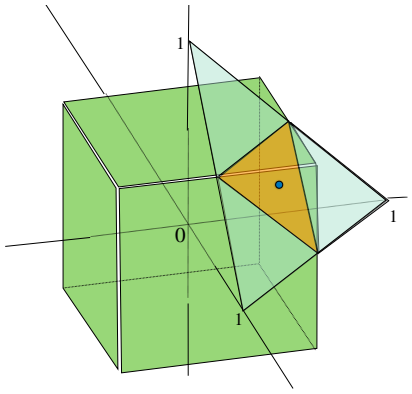


Figure 6.4: The set of mixture weights $\pi = (\pi_1, \pi_2, \pi_3)$ of conformity $\mathrm{conf}(p_\pi) \geq \theta$ is given by the intersection of the box constraints $0 \leq \pi_i \leq (3\theta)^{-1}$ for $i = 1, 2, 3$, with the simplex constraint $\pi_1 + \pi_2 + \pi_3 = 1$.

the average training loss, the resulting model w can sacrifice performance on "difficult" clients in order to perform well on average. In other words, it is not guaranteed to perform well on *individual* test clients, whose distribution p might be quite different from the average training distribution $(1/n)\sum_{i=1}^n q_i$. Our goal in this chapter is to design an objective function, to better handle statistical heterogeneity and the associated train-test mismatch. We also design a federated optimization algorithm similar to FedAvg to optimize it.

Conformity and Heterogeneity. In this chapter, we consider test clients whose distribution p can be written as a mixture $p_\pi := \sum_{i=1}^n \pi_i q_i$ of the training distribution q_i of the clients with weights $\pi \in \Delta^{n-1}$. Here, Δ^{n-1} denotes the probability simplex in $\mathbb{R}^n$. The test distribution p_π is different from the average training distribution $p_{\text{train}} = (1/n)\sum_{i=1}^n q_i$ if the mixture weights π are different from $1/n$ at training time.

We now define *conformity* of a mixture p_π to the training distribution p_{train}, as a measure of the degree of similarity between p_π and p_{train}.

Definition 6.1. The *conformity* $\mathrm{conf}(p_\pi) \in [n^{-1}, 1]$ of a mixture distribution p_π with weights π is defined as $\left(n \max_{i \in [n]} \pi_i\right)^{-1}$. The conformity of a client refers to the conformity of its data distribution.

When π coincides with the uniform distribution over $[n]$, we have that $\mathrm{conf}(p_\pi) = 1$, and this is the largest possible value of $\mathrm{conf}(p_\pi)$. On the other extreme, suppose that $\pi_i = 1$ for some i, so that π is very different from the uniform distribution. Here, $\mathrm{conf}(p_\pi) = 1/n$, which is the smallest value it takes. In other words, the conformity measures how similar the mixture weights π of p_π are to the original weights $(1/n, \cdots, 1/n)$.

More generally, a mixture distribution p_π with $\mathrm{conf}(p_\pi) \geq \theta$ must satisfy $\pi_i \leq 1/(\theta n)$ for each i. In other words, the set of all mixture weights $\{\pi \in \Delta^{n-1} : \mathrm{conf}(p_\pi) \geq \theta\}$ lie in an axis-parallel box around $(1/n, \cdots, 1/n)$, as shown in Figure 6.4. We do not directly impose a lower bound on π_i because it is not realistic to assume that the distribution on a test client must necessarily contain a component of every training distribution q_i.

Assuming that the training clients are a representative sample of the population of clients, every client's distribution can be well-approximated by a mixture p_π for some $\pi \in \Delta^{n-1}$. The conformity of a client is *a scalar summary of how close it is to the population.* A test client with conformity $\theta \approx 1$ closely conforms to

the population. Then, a model trained on the population p_{train} is expected to have high predictive power. In contrast, a test client with $\theta \approx 0$ would be vastly different from the population p_{train}, and the predictive power of a model trained on p_{train} could be poor. The inverse of the conformity $1/\text{conf}(p_\pi)$ is a measure of how much p_π is shifted relative to p_{train}.

There is a trade-off between fitting the population and supporting non-conforming test clients. The conformity θ presents a natural way to encapsulate this tradeoff in a scalar parameter. That is, given a conformity $\theta \in (0, 1)$, we choose to only support test distributions p_π with $\text{conf}(p_\pi) \geq \theta$.

6.3 Handling Heterogeneity with Δ-FL

In this section, we introduce the Δ-FL framework in Section 6.3.1, and propose an algorithm to optimize in the federated setting in Section 6.3.2.

6.3.1 The Δ-FL Framework

The Δ-FL framework aims to address the train-test distributional mismatch by supplying each test client with a model appropriate to its conformity. Given a discretization $\{\theta_1, \ldots, \theta_r\}$ of $(0, 1]$, Δ-FL maintains r models, one for each conformity level θ_j. The local data is not allowed to leave a client due to privacy restrictions; hence, the conformity of a test client cannot be measured. Instead, we allow each test client to tune their conformity. See the schematic in Figure 6.1 for an illustration.

To train a model for a conformity level θ, we aim to do well on *all* distributions p_π with $\text{conf}(p_\pi) \geq \theta$:

$$\begin{aligned} &\min_{w \in \mathbb{R}^d} \left[F_\theta(w) := \max_{\pi \in \mathcal{P}_\theta} F(w; p_\pi) + \frac{\lambda}{2}\|w\|^2 \right], \\ &\text{where,} \quad \mathcal{P}_\theta := \left\{ \pi \in \Delta^{n-1} \,:\, \text{conf}(p_\pi) \geq \theta \right\}. \end{aligned} \tag{6.6}$$

In contrast, the vanilla FL objective optimizes $F(w; p_{\text{train}})$, which is defined on the basis of the training distribution p_{train}. We observe that Δ-FL is designed to be robust on all test clients with conformity of at least θ.

Connection to the Superquantile. The objective function of (6.6) brings the notion of superquantile into play. In particular, we show that the Δ-FL objective is the superquantile of a discrete random variable with the per-client losses.

Property 6.2. *Let $Z(w)$ be a discrete random variable which takes the value $F_i(w)$ with probability $1/n$ for $i = 1, \ldots, n$. Then, we have that $F_\theta(w) = \mathbb{S}_\theta(Z(w)) + (\lambda/2)\|w\|^2$.*

Proof. The proof follows from the following equality, which holds due to linear programming duality:

$$\max_{\pi \in \mathcal{P}_\theta} \sum_{i=1}^{n} \pi_i x_i = \min_{(\eta, \mu) \in M} \left\{ \eta + \frac{1}{\theta} \sum_{i=1}^{n} \frac{\mu_i}{N} \right\}$$

with $M = \{(\eta, \mu) \in \mathbb{R} \times \mathbb{R}^N_+ \,:\, \mu_i \geq x_i - \eta \text{ for } i \in [n]\}$. □

Algorithm 6.1. The Δ-FL Algorithm

Input:

Input: Initial iterate $w^{(0)}$, number of communication rounds T, number of clients per round m, number of local updates τ, local step size γ

1: **for** $t = 0, 1, \cdots, T-1$ **do**
2: Sample m clients from $[n]$ without replacement in S
3: Estimate the $(1-\theta)$-quantile of $F_i(w^{(t)})$ for $i \in S$ with distributed differential privacy (Algorithm 6.2); call this $Q^{(t)}$
4: **for** each selected client $i \in S$ in parallel **do**
5: Set $\tilde{\pi}_i^{(t)} = \mathbb{I}\left(F_i(w^{(t)}) \geq Q^{(t)}\right)$
6: Initialize $w_{k,0}^{(t)} = w^{(t)}$
7: **for** $k = 0, \cdots, \tau - 1$ **do**
8: $w_{i,k+1}^{(t)} = (1-\gamma\lambda) w_{i,k}^{(t)} - \gamma \nabla F_i(w_{i,k}^{(t)})$
9: $w^{(t+1)} = \sum_{i \in S} \tilde{\pi}_i^{(t)} w_{i,\tau}^{(t)} / \sum_{i \in S} \tilde{\pi}_i^{(t)}$
10: **return** w_T

6.3.2 Federated Optimization for Δ-FL

We now propose a federated optimization algorithm for the Δ-FL objective (6.6). While there could be many approaches to optimizing (6.6), we consider algorithms similar to FedAvg for their ability to avoid communication bottlenecks and preserve the privacy of user data.

The objective function (6.6) is effectively the average loss of the clients in the tail, as visualized in Figure 6.3. Therefore, a natural algorithm to minimize it first evaluates the loss on all the clients, and only performs gradient updates on those clients in the tail above the $(1-\theta)$-quantile. However, a practical algorithm cannot assume that all the clients are available at a given point in time. Therefore, we perform the same operation on a subsample of clients.

Concretely, the optimization algorithm for the Δ-FL objective (6.6) is given in Algorithm 6.1. It has the following four steps:

(a) *Model Broadcast* (line 2): The server samples a set S of m clients from $[n]$ and sends the current model $w^{(t)}$.

(b) *Quantile Computation and Reweighting* (lines 3 and 5): Selected clients $i \in S$ and the server collaborate to estimate the $(1-\theta)$-quantile of the losses $F_i(w^{(t)})$ with differential privacy. The clients then update their weights to be zero if their loss is smaller than the estimated quantile and leave them unchanged otherwise. This ensures that model updates are only aggregated from the tail clients; cf. Figure 6.3.

(c) *Local Updates* (loop of line 7): Staring from $w_{k,0}^{(t)} = w^{(t)}$, each client $i \in S$ makes τ local gradient or stochastic gradient descent steps with a learning rate γ.

(d) *Update Aggregation* (line 9): The models from the selected clients are sent to the server and aggregated to update the server model, with weights from line 5).

Compared to FedAvg, Δ-FL has the additional step of computing the quantile and new weights $\tilde{\pi}_i^{(t)}$ for each selected client $i \in S$ in lines 3 and 5. Let us consider Δ-FL in relation to the key aspects of federated learning which we introduced in Section 5.1.1.

(1) *Communication Bottleneck*: Identical to FedAvg, Δ-FL algorithm performs multiple computation rounds per communication round.

(2) *Statistical Heterogeneity*: The Δ-FL objective (6.6) is designed to better handle the statistical hetero-

Algorithm 6.2. Quantile Computation with Distributed Differential Privacy

Input: Ring size M, set S of clients where each client i has a scalar $\ell_i \in [0, B]$, target quantile $1-\theta \in (0,1)$, discretization $l_0, l_1, \cdots, l_b$ of $[0, B]$, variance proxy σ^2, scaling factor $c \in \mathbb{Z}_+$

1: Each client i computes local histogram $x_i = \big(\mathbb{I}(l_{j-1} \le \ell_i < l_j)\big)_{j=1}^{b}$
2: Each client i samples $\xi_i \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma^2 I_b)$ and sets $\tilde{x}_i = (cx_i + \xi_i) \mod M$
3: Compute $s = (\sum_{i \in S} \tilde{x}_i) \mod M$ using a secure summation oracle
4: Set histogram $\hat{h} = s/c$
5: **return** Quantile estimate $Q_\theta(\hat{h})$ of the histogram $\hat{h}$ from Eq. (6.7).

geneity by minimizing the worst-case over all test distributions with conformity at least θ, while the vanilla FL objective cannot handle non-conforming clients.

(3) *Privacy*: Identical to FedAvg, Δ-FL does not require any data transfer and the aggregation of line 9 can be securely performed using secure multiparty communication. The extra step of quantile computation is also performed with distributed differential privacy, as we describe next.

Quantile Estimation with Distributed Differential Privacy. The naïve way to compute the quantile of the per-client losses in line 3 of Algorithm 6.1 is to have the clients send their losses to the server. To avoid the privacy risk of leakage of information about the clients to the server, we compute the quantile with distributed differential privacy (Kairouz et al., 2021a) using the discrete Gaussian mechanism (Canonne et al., 2020). The key idea behind differential privacy (Dwork et al., 2006a; 2016) is to ensure that the addition or removal of the data from one client does not lead to a substantial change in the output of an algorithm. A large change in the output would give a privacy adversary enough signal to learn about the client which was added or removed.

Distributed differential privacy simulates a trusted central aggregator by using a secure summation oracle (Bonawitz et al., 2017), which enables the computation of summations $\sum_{i \in S} v_i$ where $v_i \in \mathbb{R}^d$ is a privacy-sensitive vector residing with client i. Practical implementations of such algorithms are based on cryptographic techniques such as secure multiparty computation (Evans et al., 2018), which requires each component of the vectors v_i to be discretized to the ring $\mathbb{Z}_M$ of integers modulo M.[1] As defined in Definition 5.1 (Chapter 5), a secure summation oracle $\mathcal{S} : (\mathbb{Z}_M^d)^m \to \mathbb{Z}_M$ returns the sum $\mathcal{S}\big((v_i)_{i \in S}\big) = \big(\sum_{i \in S} v_i\big) \mod M$ without revealing any further information to a privacy adversary.

Our algorithm is given in Algorithm 6.2. We assume that the losses are bounded as $F_i(w) \in [0, B]$ and that we are given b bin edges $0 \le l_0 < l_1 < \cdots < l_b = B$. Each client i first computes a local histogram $x_i \in \{0,1\}^b$ of b bins as the indicator vector

$$x_i = \Big(\mathbb{I}\big(F_i(w) \in [l_{j-1}, l_j)\big) \Big)_{j=1}^{b}.$$

Each client then adds random discrete Gaussian[2] noise $\xi_i \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma^2 I_b)$ with scale parameter σ^2, and finally sums them up using a secure summation oracle. At the end of all these steps, the server has a vector $\hat{h} \in \mathbb{R}^b$ which approximates the true histogram $h = \sum_{i \in S} x_i$ of per-client losses. While $\hat{h}$ might not be a valid histogram due to negative values, we continue to call it a "histogram" with a slight abuse of terminology.

[1] For ease of handling negative integers, we perform modular arithmetic over the ring $\{-M/2+1, \cdots, -1, 0, 1, 2, \cdots, M/2\}$ rather than $\{0, 1, \cdots, M-1\}$.

[2] See Section D.2 for a formal definition.

Algorithm 6.3. The Δ-FL Algorithm with Exact Reweighting

Input: Same as Algorithm 6.1

1: **for** $t = 0, 1, \cdots, T-1$ **do**
2: Sample m clients from $[n]$ without replacement in S
3: Compute $\pi^{(t)} = \arg\max_{\pi \in \mathcal{P}_{\theta,S}} \sum_{i \in S} \pi_i F_i(w^{(t)})$
4: **for** each selected client $i \in S$ in parallel **do**
5: Initialize $w_{i,0}^{(t)} = w^{(t)}$
6: **for** $k = 0, \cdots, \tau - 1$ **do**
7: $w_{i,k+1}^{(t)} = (1 - \gamma\lambda) w_{i,k}^{(t)} - \gamma \nabla F_i(w_{i,k}^{(t)})$
8: $w^{(t+1)} = \sum_{i \in S} \pi_i^{(t)} w_{i,\tau}^{(t)}$
9: **return** w_T

The final step is to define and return an appropriate notion of a $(1-\theta)$-quantile of $\hat{h}$. A non-negative histogram $h \in \mathbb{R}_+^b$ can be viewed as a random variable $Z(h)$ with mass function $\mathbb{P}\big(Z(h) = l_{j-1}\big) \propto h_j$ for $j = 1, \cdots, b$. The $(1-\theta)$-quantile of the non-negative histogram h is simply the quantile function of this induced $Z(h)$:

$$Q_\theta(h) := Q_\theta(Z(h)) = \min_{j \in [b]} \left\{ l_{j-1} \,:\, h_{j+1} + \cdots + h_b \le \theta(h_1 + \cdots + h_b) \right\}.$$

Similarly, for approximate histograms $\hat{h}$ that allow negative values, we return the bin edge l_j such that the mass $h_{j+1} + \cdots + h_b$ to the right of this edge is as close to θ as possible:

$$Q_\theta(\hat{h}) := l_{j_\theta^*(\hat{h})} \quad \text{where} \quad j_\theta^*(\hat{h}) = \arg\min_{j \in [b]} \left| (h_{j+1} + \cdots + h_b) - \theta(h_1 + \cdots + h_b) \right|. \tag{6.7}$$

6.4 Theoretical Analysis

In this section, we analyze the convergence analysis of Δ-FL (Section 6.4.1) and study the differential privacy properties of the quantile computation (Section 6.4.2).

6.4.1 Convergence Analysis

We study the convergence of Algorithm 6.1 with respect to the objective (6.6) in two cases: (i) the general non-convex case, and, (ii) when each $F_i(w)$ is convex.

Assumptions. We make some assumptions on the per-client losses F_i, which are assumed to hold throughout this section. For each client $i \in [n]$, the objective F_i is

(a) B-bounded, i.e., $0 \le F_i(w) \le B$ for all $w \in \mathbb{R}^d$,
(b) G-Lipschitz, i.e., $|F_i(w) - F_i(w')| \le G\|w - w'\|$ for all $w, w' \in \mathbb{R}^d$, and,
(c) L-smooth, i.e., F_i is continuously differentiable and its gradient ∇F_i is L-Lipschitz.

Equivalent Algorithm. Algorithm 6.1 is not amenable to theoretical analysis as it is stated because the quantile function of discrete random variables computed in line 3 is piecewise constant and discontinuous.

To overcome this obstacle, we introduce a near-equivalent algorithm in Algorithm 6.3, which replaces the reweighting step of Algorithm 6.1 (lines 3 and 5) with the ideal reweighting suggested by the objective (6.6).

Let us start with the case of $S = [n]$. Ideally, we wish the weights $\pi^{(t)}$ to achieve the maximum over π in the objective (6.6). It then follows by the chain rule (Rockafellar and Wets, 2009, Thm. 10.6) that $\sum_{i=1}^n \pi_i^{(t)} \nabla F_i(w^{(t)}) + \lambda w^{(t)} \in \partial F_\theta(w^{(t)})$, where ∂F_θ is the regular subdifferential of F_θ. This allows us to derive convergence guarantees.

Algorithm 6.3 extends this intuition to the setting where only a subsample $S \subset [n]$ of clients are available in each round. We define the counterpart of the constraint set $\mathcal{P}_\theta$ from (6.6) defined on a subset $S \subset [n]$ of m clients as:

$$\mathcal{P}_{\theta,S} = \left\{ \pi \in \Delta^{|S|-1} \,:\, \pi_i \leq \frac{1}{\theta m}, \text{ for } k \in S \right\}, \tag{6.8}$$

where we denote $(\pi_i)_{i \in S} \in \mathbb{R}^{|S|}$ by π with slight abuse of notation. With this notation, Algorithm 6.3 computes the new weights of the clients as

$$\pi^{(t)} = \operatorname*{arg\,max}_{\pi \in \mathcal{P}_{\theta,S}} \sum_{i \in S} \pi_i F_i(w^{(t)}) \,.$$

We now analyze how close Algorithm 6.3 is to Algorithm 6.1. Let $Z(w)$ be a discrete random variable which takes the value $F_i(w)$ with probability $1/n$ for $i = 1, \ldots, n$, and let $Q_\theta(Z(w))$ denote its $(1-\theta)$-quantile. The weights $\hat{\pi} \in \Delta^{n-1}$ considered in Algorithm 6.1 (assuming that $Q^{(t)}$ is the exact quantile of $\{F_i(w^{(t)}) \,:\, k \in S\}$) are given by a hard-thresholding based on whether $F_i(w)$ is larger than its $(1-\theta)$-quantile:

$$\tilde{\pi}_i = \mathbb{I}\big(F_i(w) \geq Q_\theta(Z(w))\big) \,, \quad \text{and,} \quad \hat{\pi}_i = \frac{\tilde{\pi}_i}{\sum_{i'=1}^n \tilde{\pi}_{i'}}. \tag{6.9}$$

The objective defined by these weights is $\hat{F}_\theta(w) = \sum_{i=1}^n \hat{\pi}_i F_i(w) + (\lambda/2)\|w\|^2$. The next proposition shows that $\hat{F}_\theta(w) = F_\theta(w)$ under certain conditions, or is a close approximation, in general.

Proposition 6.3. *Assume $F_1(w) < \cdots < F_n(w)$ and let $i^\star = \lceil \theta n \rceil$. Then, we have,*
(a) $\pi^\star = \arg\max_{\pi \in \mathcal{P}_\theta} \sum_{i=1}^n \pi_i F_i(w)$ is unique,
(b) $Q_\theta(Z(w)) = F_{i^\star}(w)$,
(c) if θn is an integer, then $\hat{\pi} = \pi^\star$ so that $\hat{F}_\theta(w) = F_\theta(w)$, and,
(d) if θn is not an integer, then

$$0 \leq F_\theta(w) - \hat{F}_\theta(w) \leq \frac{B}{\theta n} \,.$$

Proof. We assume w.l.o.g. that $\lambda = 0$. We apply the property that the superquantile is a tail mean (cf. Figure 6.3) for discrete random variables (Rockafellar and Uryasev, 2002, Proposition 8) to get

$$F_\theta(w) = \frac{1}{\theta n} \sum_{i=i^\star+1}^{n} F_i(w) + \left(1 - \frac{\lfloor \theta n \rfloor}{\theta n}\right) F_{i^\star}(w) \,.$$

Comparing with (6.6), this gives a closed-form expression for $\pi^\star$, which is unique because $F_{i^\star-1}(w) < F_{i^\star}(w) < F_{i^\star+1}(w)$. For (b), note that $Q_\theta(Z(w)) = \inf\{\eta \in \mathbb{R} \,:\, \mathbb{P}(Z(w) > \eta) \leq \theta\}$ equals $F_{i^\star}(w)$ by

definition of $i^\star$. Therefore, if θn is an integer, $\pi^\star$ coincides exactly with $\hat{\pi}$. When θn is not an integer, we have

$$\hat{F}_\theta(w) = \frac{1}{n - i^\star + 1} \sum_{i=i^\star}^{n} F_i(w)\,.$$

The bound on $\hat{F}_\theta(w) - F_\theta(w)$ follows from elementary manipulations together with $0 \leq F_i(w) \leq B$. $\square$

Proposition 6.3 shows that when θm is an integer, Algorithm 6.3 is identical to Algorithm 6.1 where line 5 exactly computes the quantile of the per-client losses. We record another consequence of Proposition 6.3, namely, that the reweighting $\pi^{(t)}$ is sparse.

Remark 6.4. *Proposition 6.3 shows that Δ-FL's reweighting $\pi^{(t)}$ (line 3 of Algorithm 6.3) is sparse. That is, $\pi_i^{(t)}$ is non-zero only for exactly $\lceil \theta m \rceil$ clients with the largest losses.*

Bias due to Partial Participation. Note that we define the objective (6.6) as the maximum over all distributions in $\mathcal{P}_\theta$, but Algorithm 6.3 only maximizes weights over a set S of m clients in each round (line Line 3). Therefore, the updates performed by Algorithm 6.3 are not unbiased. In particular, Algorithm 6.3 minimizes the objective:

$$\overline{F}_\theta(w) := \mathbb{E}_{S \sim U_m}\left[F_{\theta,S}(w)\right], \quad \text{where } F_{\theta,S}(w) = \max_{\pi \in \mathcal{P}_{\theta,S}} \sum_{i \in S} \pi_i F_i(w) + \frac{\lambda}{2}\|w\|^2$$

is the analogue of (6.6) defined on a sample $S \subset [n]$ of clients, and U_m is the uniform distribution over subsets of $[n]$ of size m. Fortunately, the bias introduced by Line 3 can be bounded as (Levy et al., 2020b, Prop. 1)

$$\sup_{w \in \mathbb{R}^d} \left|\overline{F}_\theta(w) - F_\theta(w)\right| \leq \frac{B}{\sqrt{\theta m}}\,. \tag{6.10}$$

Our general strategy will be to study the convergence (near-stationarity or near-optimality) in terms of the objective $\overline{F}_\theta$ which Algorithm 6.3 actually minimizes, and then translate that to a convergence result on the original objective F_θ using the bound (6.10).

Convergence: Nonconvex Case. We start with the convergence analysis in the nonconvex case with no regularization (i.e., $\lambda = 0$). Since $\overline{F}_\theta$ is nonsmooth and nonconvex, we state the convergence guarantee in terms of the Moreau envelope of $\overline{F}_\theta$ (Hiriart-Urruty and Lemaréchal, 1996) following the idea of (Drusvyatskiy and Paquette, 2019; Davis and Drusvyatskiy, 2019). Given a parameter $\mu > 0$, we define the Moreau envelope of $\overline{F}_\theta$ as

$$\overline{\Phi}_\theta^\mu(w) = \inf_{v \in \mathbb{R}^d} \left\{ \overline{F}_\theta(v) + \frac{\mu}{2}\|v - w\|^2 \right\}. \tag{6.11}$$

The Moreau envelope satisfies a number of remarkable properties for $\mu > L$ (Drusvyatskiy and Paquette, 2019, Lemma 4.3). It is well-defined, and the infimum on the right-hand side admits a unique minimizer, called the proximal point of w, and denoted $\operatorname{prox}_{\overline{F}_\theta/\mu}(w)$. Second, the Moreau envelope is continuously differentiable with $\nabla \overline{\Phi}_\theta^\mu(w) = \mu(w - \operatorname{prox}_{\overline{F}_\theta/\mu}(w))$. Finally, the stationary points of $\overline{\Phi}_\theta^\mu$ and $\overline{F}_\theta$ coincide. Interestingly, the bound $\|\nabla \overline{\Phi}_\theta^\mu(w)\| \leq \varepsilon$ directly implies a near-stationarity on $\overline{F}_\theta$, and hence the original F_θ, in the following variational sense: the proximal point $z = \operatorname{prox}_{\overline{F}_\theta/\mu}(w)$ satisfies (Drusvyatskiy and Paquette, 2019, Sec. 4.1):

(a) v is close to w; that is, $\|v - w\| \le \varepsilon/\mu$,
(b) v is nearly stationary on $\overline{F}_\theta$; that is $\operatorname{dist}\big(0, \partial \overline{F}_\theta(v)\big) \le \varepsilon$, where $\partial \overline{F}_\theta$ refers to the regular subdifferential, and,
(c) $\overline{F}_\theta$ is uniformly close to F_θ as per (6.10).

Thus, we state the convergence guarantee of our algorithm in the nonsmooth nonconvex case in terms of the Moreau envelope $\overline{\Phi}_\theta^\mu$ (although it never appeared in the algorithm).

Theorem 6.5. *Let the number of rounds T be fixed and set $\mu = 2L$. Denote $\Delta F_0 = F_\theta(w^{(0)}) - \inf \overline{F}_\theta$. Let $\hat{w}$ denote a uniformly random sample from the sequence $\big(w^{(0)}, \cdots, w^{(T-1)}\big)$ produced by Algorithm 6.3. Then, there exists a learning rate γ depending on the number of rounds T and problem parameters $\tau, L, G, \Delta F_0$ such that*

$$\mathbb{E}\big\|\nabla \overline{\Phi}_\theta^\mu(\hat{w})\big\|^2 \le \sqrt{\frac{\Delta_0 L G^2}{T}} + (1 - \tau^{-1})^{1/3} \left(\frac{\Delta_0 L G}{T}\right)^{2/3} + \frac{\Delta_0 L}{T}\,.$$

Proof. Let $v^{(t)} = \operatorname{prox}_{\overline{F}_\theta/\mu}(w^{(t)})$ be the proximal point of $w^{(t)}$. We expand out the recursion $w^{(t+1)} = w^{(t)} - \gamma \sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla F_i(w_{i,k}^{(t)})$ to get

$$\begin{aligned}
\overline{\Phi}_\theta^\mu(w^{(t+1)}) &\le \overline{F}_\theta(v^{(t)}) + \frac{\mu}{2}\Big\|v^{(t)} - w^{(t+1)}\Big\|^2 \\
&= \overline{F}_\theta(v^{(t)}) + \frac{\mu}{2}\Big\|v^{(t)} - w^{(t)}\Big\|^2 + \mu\gamma\left\langle v^{(t)} - w^{(t)}, \sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla F_i(w_{i,k}^{(t)}) \right\rangle \\
&\qquad + \frac{\mu\gamma^2}{2}\left\|\sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla F_i(w_{i,k}^{(t)})\right\|^2 \\
&= \overline{\Phi}_\theta^\mu(w^{(t)}) + \mathcal{T}_1 + \mathcal{T}_2\,.
\end{aligned}$$

The term $\mathcal{T}_1$ which carries a $O(\gamma)$-coefficient controls the rate of convergence while $\mathcal{T}_2$ carries a $O(\gamma^2)$-coefficient and is a noise term. The latter can be controlled by making the learning rate small. We can handle the first term $\mathcal{T}_1$ by leveraging a property of $\overline{F}_\theta$ known as *weak convexity*, meaning that adding a quadratic makes it convex. In particular, $\overline{F}_\theta + (L/2)\|\cdot\|^2$ is convex, so that

$$\left\langle v^{(t)} - w^{(t)}, \sum_{i \in S} \pi_i^{(t)} \nabla F_i(w^{(t)}) \right\rangle \le F_{\theta,S}(v^{(t)}) - F_{\theta,S}(w^{(t)}) + \frac{L}{2}\Big\|v^{(t)} - w^{(t)}\Big\|^2\,.$$

Next, we take an expectation with respect to the sampling S of clients (i.e., conditioned on $\mathcal{F}^{(t)} = \sigma(w^{(t)})$, the σ-algebra generated by $w^{(t)}$). Since $v^{(t)}$ is independent of S (i.e., $v^{(t)}$ is $\mathcal{F}^{(t)}$-measurable), we get $\overline{F}_\theta$ on the right-hand side. Next, we use that $v^{(t)}$ minimizes the strongly convex right hand side of (6.11) to get

$$\begin{aligned}
\mathbb{E}_t\left\langle v^{(t)} - w^{(t)}, \sum_{i \in S} \pi_i^{(t)} \nabla F_i(w^{(t)}) \right\rangle &\le -(\mu - L)\Big\|v^{(t)} - w^{(t)}\Big\|^2 \\
&= -\frac{\mu - L}{\mu^2}\Big\|\nabla \overline{\Phi}_\theta^\mu(w^{(t)})\Big\|^2\,.
\end{aligned}$$

We plug in $\mu = 2L$ to get a bound on $\mathcal{T}_1$ in terms of $\big\|\nabla \overline{\Phi}_\theta^\mu(w^{(t)})\big\|^2$. A standard argument to handle the noise term $\mathcal{T}_2$ and telescoping the resulting inequality over $t = 0, \cdots, T-1$ completes the proof. The full details are given in Section D.1.1. □

Convergence: Convex Case. We consider the convergence of function values in the case where each F_i is convex. Owing to the non-smoothness of F_θ and $\overline{F}_\theta$, we consider the following smoothed version of the objective in (6.6) and the corresponding modification to Algorithm 6.3. First, define the Kullback-Leibler (KL) divergence between $\pi \in \Delta^{|S|-1}$ and the uniform distribution $(1/|S|, \cdots, 1/|S|)$ over $S \subset [n]$ as

$$D_S(\pi) = \sum_{i \in S} \pi_i \log(\pi_i \, |S|) \, .$$

We simply write $D(\pi)$ when $S = [n]$. Inspired by (Nesterov, 2005b; Beck and Teboulle, 2012; Devolder et al., 2014), we define the smooth counterpart to (6.6) as

$$F_\theta^\nu(w) = \max_{\pi \in \mathcal{P}_\theta} \left\{ \sum_{i=1}^n \pi_i F_i(w) - \nu D(\pi) \right\} + \frac{\lambda}{2}\|w\|^2 \, , \tag{6.12}$$

where $\nu > 0$ is a fixed smoothing parameter. We have that $|F_\theta^\nu(w) - F_\theta(w)| \leq 2\nu \log n$. Finally, we modify line 3 of Algorithm 6.3 to handle F_θ^ν rather than F_θ as

$$\pi^{(t)} = \arg\max_{\pi \in \mathcal{P}_{\theta,S}} \left\{ \sum_{i \in S} \pi_i F_i(w^{(t)}) - \nu D_S(\pi) \right\} . \tag{6.13}$$

Theorem 6.6. *Suppose each function F_i is convex and $0 < \lambda < L$. Define a condition number $\kappa = (L+\lambda)/\lambda$ and fix a time horizon $T \geq \sqrt{2\kappa^3}$. Consider the sequence $(w^{(t)})_{t=0}^T$ of iterates produced by the Algorithm 6.3 with line 3 replaced by* (6.13)*. Define the averaged iterate*

$$\overline{w}^{(t)} = \frac{\sum_{j=0}^t \beta_j w^{(i)}}{\sum_{i=0}^t \beta_j}, \quad \textit{where} \quad \beta_j = \left(1 - \frac{\gamma\lambda\tau}{2}\right)^{-(1+j)} \, ,$$

and $w^\star = \arg\min_{w \in \mathbb{R}^d} F_\theta(w)$. Then, there exist learning rate γ and smoothing parameter ν depending on the number of communication rounds T as well as problem parameters $\tau, G, \lambda, L, \Delta_0, \theta, m$, such that the iterate $\overline{w}^{(T)}$ satisfies the bound

$$\mathbb{E}F_\theta(\overline{w}^{(T)}) - F_\theta(w^\star) \leq \lambda\|w^{(0)} - w^\star\|^2 \exp\left(-\frac{T}{\sqrt{2\kappa^3}}\right) + \frac{G^2}{\lambda T} + \frac{G^2\kappa^2}{\lambda T^2} + \frac{B}{\sqrt{\theta m}} \, ,$$

where we hide absolute constants and factors polylogarithmic in T and problem parameters G, λ, κ.

Remark 6.7 (About the Rate)**.** *As soon as $T \gtrsim \kappa^{3/2}$ (ignoring constants and polylog factors), we achieve the optimal rate of $1/(\lambda T)$ rate of strongly convex stochastic optimization up to the bias $B/\sqrt{\theta m}$.*

Further, the bias $B/\sqrt{\theta m}$ due to partial participation can be controlled by choosing the cohort size m large enough. In the experiments of Section 6.7, we obtain meaningful numerical results when m is around 50 or 100 and θ around $1/2$, indicating that the worst-case bound (6.10) *can be pessimistic.*

The proof is given in Appendix D.1.

6.4.2 Privacy Analysis

We now analyze the privacy and utility of Algorithm 6.2.

First, we recall the definition of zero-concentrated differential privacy (Bun and Steinke, 2016). A randomized algorithm $\mathcal{A}$ satisfies $(1/2)\varepsilon^2$-zero concentrated differential privacy if the Rényi α-divergence

$D_\alpha(\mathcal{A}(X)\|\mathcal{A}(X')) \le \alpha\varepsilon^2/2$ for all $\alpha \in (0,\infty)$ and all sequences X, X' of inputs that differ by the addition or removal of the data of one client. Intuitively, the addition or removal of the data contributed by one client should not change the output distribution of the randomized algorithm by much, as measured by the Rényi divergence. A smaller value of ε implies a stronger privacy guarantee. This notion of differential privacy can be translated back-and-forth with the usual one, cf. (Canonne et al., 2020).

Error Criterion. We approximate the $(1-\theta)$-quantile of the n per-client losses $\ell_i = F_i(w)$ for $i = 1, \cdots, n$ by the quantile of a histogram $h = (h_1, \cdots, h_b)$ of client losses with b bins. We assume the bin edges $0 = l_0 < l_1 < \cdots < l_b = B$ are given so that the entries of the histogram are given by $h_j = \sum_{i=1}^n \mathbb{I}(l_{j-1} \le \ell_i < l_j)$. The bin edge l_j corresponding to index $j \in [b]$ approximates the $(1-\theta)$-quantile well if $h_{j+1}+\cdots+h_b \approx \theta(h_1+\cdots+h_b)$. We measure this error of approximation by the difference between the two sides. Formally, we define the error $R_\theta(h,j)$ of approximating the $(1-\theta)$-quantile of histogram $h \in \mathbb{R}^b$ with index $j \in [b]$ by

$$R_\theta(h,j) = \left| \frac{h_{j+1}+\cdots+h_b}{h_1+\cdots+h_b} - \theta \right| . \tag{6.14}$$

We define the best achievable error $R^*_\theta(h)$ for estimating the $(1-\theta)$-quantile of histogram h and the best-approximating index $j^*(h)$ as

$$R^*_\theta(h) = \min_{j\in[b]} R_\theta(h,j)\,, \quad \text{and} \quad j^*_\theta(h) = \arg\min_{j\in[b]} R_\theta(h,j)\,, \tag{6.15}$$

where we assume ties are broken in an arbitrary but deterministic manner — note that $j^*_\theta(h)$ defined here is identical to (6.7). Lastly, we define the *quantile error* $\Delta_\theta(h,\hat{h})$ of estimating the quantile of h from that of $\hat{h}$ as

$$\Delta_\theta(\hat{h},h) = R_\theta\big(h, j^*_\theta(\hat{h})\big)\,.$$

Essentially, if the index $j^*_\theta(\hat{h})$ computed from the estimate $\hat{h}$ corresponds to the $(1-\theta')$-quantile of h, the quantile error satisfies $\Delta_\theta(h,\hat{h}) = |\theta-\theta'|$.

Privacy and Utility Analysis. We now analyze the differential privacy bound of Algorithm 6.2 as well as the error in the quantile computation.

Theorem 6.8. *Fix a $\delta > 0$. Suppose that $\sigma \ge 1/2$ and $c > 0$ are given, and and the modular arithmetic is performed on base $M \ge 2 + 2cn + 2n\sqrt{2\sigma^2\log(4nb/\delta)}$. Then we have the following with probability at least $1-\delta$:*

(a) Algorithm 6.2 satisfies $(1/2)\varepsilon^2$-concentrated DP with

$$\varepsilon = \min\left\{ \sqrt{\frac{c^2}{n\sigma^2} + \frac{\psi b}{2}}, \frac{c}{\sqrt{n}\sigma} + \psi\sqrt{b} \right\},$$

where $\psi = 10\sum_{i=1}^{n-1} \exp\big(-2\pi^2\sigma^2 i/(i+1)\big) \le 10(n-1)\exp(-2\pi^2\sigma^2)$.

(b) The quantile error of histogram $\hat{h}$ returned by Algorithm 6.2 is at most

$$\Delta_\theta(\hat{h},h) \le R^*_\theta(\hat{h})\left(1 + \sqrt{\frac{2\sigma^2 b}{c^2 n}\log\frac{4}{\delta}}\right) + (2-\theta)\sqrt{\frac{2\sigma^2 b}{c^2 n}\log\frac{4}{\delta}}\,,$$

where $R^*_\theta(\hat{h})$ is the error in the estimation of $(1-\theta)$-quantile of histogram $\hat{h}$.

Let us interpret the result. The effective noise scale is σ/c. Since the dominant term of the privacy error is $\varepsilon \approx c/(\sigma\sqrt{n})$, we choose $\sigma/c \approx (\varepsilon\sqrt{n})^{-1}$, so that the algorithm satisfies $(1/2)\varepsilon^2$-concentrated DP. The role of c is to avoid degeneracy of the discrete Gaussian as $\sigma \to 0$. In particular, the theorem requires $\sigma \geq 1/2$. The error resulting quantile error $\Delta_\theta(\hat{h}, h)$ is (ignoring constants and log factors)

$$\Delta_\theta(\hat{h}, h) \lesssim R^*_\theta(\hat{h}) \left(1 + \frac{\sqrt{b}}{\varepsilon n}\right) + (1+\rho)\frac{\sqrt{b}}{\varepsilon n}\,.$$

That is, the quantile error scales as $\sqrt{b}/(\varepsilon n)$. The total communication cost is $O(bn \log M)$ bits. If we take $\sigma = O(1)$ and $c = O(\varepsilon\sqrt{n})$, we require $M \gtrsim n^{3/2}$, so that the total communication cost is $O(bn \log n)$.

Proof. Define the event

$$E_{\mathrm{mod}} = \bigcap_{i=1}^{n}\bigcap_{j=1}^{b}\left\{-\frac{M-2}{2n} \leq cx_{i,j} + \xi_{i,j} \leq \frac{M-2}{2n}\right\}. \tag{6.16}$$

Note that under E_{mod}, no modular wraparound occurs in the algorithm, i.e., $\tilde{x}_i = cx_i + \xi_i$ and $\hat{h} = \sum_{i=1}^n \frac{\tilde{x}_i}{c} = \sum_{i=1}^n \left(x_i + \frac{\xi_i}{c}\right)$. It follows from standard concentration arguments that E_{mod} holds with high probability under the given assumptions (details in Appendix D, Section D.2). For the rest of this proof, we assume that it holds.

The analysis of the privacy follows from the sensitivity of the sum query. Namely, let $X = (x_1, \cdots, x_n)$ be a sequence and define $A(X) = \sum_{i=1}^n cx_i$ as the (rescaled) sum query. In our case, each x_i is a canonical basis vector since it is a local histogram constructed from a single scalar. Algorithm 6.2 adds discrete Gaussian noise to the sum query to make it differentially private. That is, we get the randomized algorithm $\mathcal{A}(X) = A(X) + \sum_{i=1}^n \xi_i$. It was shown in (Kairouz et al., 2021a, Corollary 12) that $\mathcal{A}(X)$ is approximately distributed as $\mathcal{N}_{\mathbb{Z}}(A(X), n\sigma^2)$, so the desired privacy guarantee follows from that of the discrete Gaussian mechanism (Canonne et al., 2020). In particular, for two sequences X and X' differing by the addition or removal of a single basis vector x', we have that

$$D_\alpha(\mathcal{A}(X)\|\mathcal{A}(X')) \approx D_\alpha(\mathcal{N}_{\mathbb{Z}}(A(X), n\sigma^2)\|\mathcal{N}_{\mathbb{Z}}(A(X'), n\sigma^2)) = \frac{\alpha c^2}{2n\sigma^2}\,.$$

A rigorous analysis of the error, following the recipe of (Kairouz et al., 2021a) leads to the first part of the theorem; the details can be found in Section D.2.

For the second part, we analyze the quantile error. Define $\hat{n} = \sum_{j=1}^b \hat{h}_j$, as the analogue to $n = \sum_{j=1}^b h_j$ and shorthand $\rho = 1 - \theta$. We bound the quantile error as

$$\begin{aligned}
\Delta_\theta\left(\hat{h}, h\right) &= \left|\frac{1}{n}\sum_{j=1}^{j^*_\theta(\hat{h})} h_j - \rho\right| \\
&\leq \frac{1}{n}\left|\sum_{j=1}^{j^*_\theta(\hat{h})} h_j - \hat{h}_j\right| + \frac{1}{n}\left|\sum_{j=1}^{j^*_\theta(\hat{h})} \hat{h}_j - \hat{n}\rho\right| + \frac{\rho}{n}\left|\hat{n} - n\right| \\
&\leq \max_{j' \in [n]} \frac{1}{cn}\left|\sum_{j=1}^{j'}\sum_{i=1}^{n} \xi_{i,j}\right| + \left(1 + \frac{|\hat{n} - n|}{n}\right) R^*_\theta(\hat{h}) + \frac{\rho}{n}\left|\hat{n} - n\right|\,.
\end{aligned}$$

Let us define an event E_{sum} under which we can bound each of the terms to get the desired bound:

$$E_{\text{sum}} = \left\{ \max_{j \in [b]} \left| \sum_{j'=1}^{j} \sum_{i=1}^{n} \xi_{k,j'} \right| \leq \sqrt{2\sigma^2 nb \log(4/\delta)} \right\} . \tag{6.17}$$

Finally, it remains to bound the probability of E_{mod} and E_{sum}. This can be achieved using standard concentration arguments, which we defer to Section D.2. □

By using a classical technique to answer range queries known as the hierarchical histogram method or tree aggregation (Hay et al., 2010; Dwork et al., 2010; Chan et al., 2011; Smith et al., 2017a), the $\sqrt{b}/\varepsilon n$ bound in Theorem 6.8 can modified to the asymptotically better bound of $\log_2^2 b/\varepsilon n$ at $2\times$ the communication cost. However, the former bound is smaller for $b \leq 6.56 \times 10^4$. In light of the usual settings of cross-device federated learning (where the number of clients per round is of the order of 50 to 5000 (Kairouz et al., 2021b)), we opt for the flat histogram method with the $\sqrt{b}/\varepsilon n$ bound.

6.5 Discussion

We discuss connections of Δ-FL to risk measures and fair resource allocation.

Connection to Risk Measures. The framework of risk measures in economics and finance formalizes the notion of minimizing the worst-case cost over a set of distributions (Föllmer and Schied, 2002; Rockafellar and Uryasev, 2013; Föllmer and Schied, 2016). The superquantile $\mathbb{S}_\theta(\cdot)$ is a special case of a risk measure. The Δ-FL framework, which minimizes the superquantile of the per-client losses (Property 6.2), can be extended to other risk measures $\mathbb{M}$ by minimizing the objective

$$F_M(w) := \mathbb{M}(Z(w)) + \frac{\lambda}{2}\|w\|^2 ,$$

where $Z(w)$ is a discrete random variable which takes value $F_i(w)$ with probability $1/n$ for $i \in [n]$. Another example of a risk measure is the *entropic risk measure*, which is defined for $\nu > 0$ as $\mathbb{M}^\nu_{\text{ent}}(Z) = \mathbb{E}[\exp(\nu Z)]/\nu$ where $\nu \in \mathbb{R}_+$ is a parameter. The entropic risk measure is well defined provided the moment generating function $\mathbb{E}[\exp(\nu Z)]$ exists, for instance, for sub-Gaussian Z. The analogue of Δ-FL with the entropic risk minimizes

$$F^\nu_{\text{ent}}(w) = \frac{1}{\nu} \log \left(\frac{1}{n} \sum_{i=1}^{n} \exp\left(\nu F_i(w)\right) \right) + \frac{\lambda}{2}\|w\|^2 .$$

This objective $F^\nu_{\text{ent}}(w)$ coincides with the one studied recently in (Li et al., 2021a) under the name Tilted-ERM subsequent to the first presentation of this work (Laguel et al., 2020b). Finally, we note that F^ν_{ent} is also related to the smoothed objective F^ν_θ from (6.12) as the limit

$$F^\nu_{\text{ent}}(w) = \lim_{\theta \to 0} F^\nu_\theta(w) ,$$

Maximin Strategy for Resource Allocation. We would like to point out an interesting analogy between distributional robustness and proportional fairness. The superquantile-based objective in Eq. (6.6) is a maximin-type objective that is reminiscent of maximin objectives used in load balancing and network scheduling (Kubiak, 2008; Stanczak et al., 2009; Pantelidou and Ephremides, 2011).

We can draw an analogy between the two worlds, federated learning and resource allocation resp., by identifying errors to rates and clients to users. The maximin fair strategy to resource allocation seeks to treat all users as fairly as possible by making their rates as large and as close as possible, so that no rate can be increased without sacrificing other rates that are smaller or equal (Pantelidou and Ephremides, 2011).

Our superquantile-based Δ-FL framework builds off the maximin decision-theoretic foundation to frame an objective that we *optimize* with respect to *parameters* of models, and this, iteratively, over multiple rounds of client-server communication, while preserving the privacy of each client.

This compositional nature of our problem, where we optimize a composition (in the mathematical sense) of a maximin-type objective and a loss function and model predictions is a difference with resource allocation in communication networks. Further explorations of the analogy are left for future work.

Model Family and Conformity Levels θ. Using a single global value of the conformity level θ for all clients could fail to balance supporting clients with low conformity with fitting the population. On the other hand, measuring the conformity of clients requires a transfer of user data, a violation of privacy. To circumvent this issue, we use a similar idea to the one of (Li et al., 2020d) where a family of models is trained simultaneously for various levels, and each test client can then tune its conformity.

6.6 Related Work

Federated learning was introduced by (McMahan et al., 2017) to handle distributed on-client learning; we refer to the surveys (Kairouz et al., 2021b; Li et al., 2020a; Gafni et al., 2022) for broader context. A plethora of recent extensions have also been proposed (Yurochkin et al., 2019; Sattler et al., 2020; Mills et al., 2020; Wei et al., 2020; Mohammadi Amiri and Gündüz, 2020; Shlezinger et al., 2021; Jhunjhunwala et al., 2021; Sery et al., 2021; Collins et al., 2021). Our approach of addressing the statistical heterogeneity by proposing a new objective function, is broadly applicable in these settings.

Distributionally robust optimization (Ben-Tal et al., 2013), which aims to train models that perform uniformly well across all subgroups instead of just on average, has witnessed a flurry of recent research (Lee and Raginsky, 2018; Duchi and Namkoong, 2019; Kuhn et al., 2019). This approach is closely related to the risk measures studied in economics and finance (Artzner et al., 1999; Rockafellar and Uryasev, 2000; Ben-Tal and Teboulle, 2007; Föllmer and Schied, 2016). The recent works (Laguel et al., 2020a; Levy et al., 2020a; Curi et al., 2020) study optimization algorithms for risk measures. More broadly, risk measures have been successfully utilized in problems ranging from bandits (Sani et al., 2012; Cassel et al., 2018), reinforcement learning (Chow et al., 2015; Tamar et al., 2015; Chow et al., 2017), and fairness in machine learning (Williamson and Menon, 2019; Rezaei et al., 2021). The federated learning method here is based on the superquantile (Rockafellar and Uryasev, 2002), a popular risk measure. We propose a stochastic optimization algorithm adapted to the federated setting and prove the convergence.

Addressing statistical heterogeneity in federated learning has led to two lines of work. The first includes algorithmic advances to alleviate the effect of heterogeneity on convergence rates while still minimizing the classical expectation-based objective function of empirical risk minimization. These techniques include the use of proximal terms (Li et al., 2020b), control variates (Karimireddy et al., 2020) or augmenting the server updates (Wang et al., 2020b; Reddi et al., 2021); we refer to the recent survey (Wang et al., 2021)

Table 6.1: Dataset description and statistics.

Task	Dataset	#Classes	Devices	#Data per client	
				Median	Max
Image Recognition	EMNIST	62	1730	179	447
Sentiment Analysis	Sent140	2	877	69	549

for details. More generally, the framework of local SGD has been used to study federated optimization algorithms (Stich, 2019; Zhou and Cong, 2018; Haddadpour et al., 2019; Dieuleveut and Patel, 2019; Li et al., 2020e; Khaled et al., 2020; Koloskova et al., 2020). Compared to these works which study federated optimization algorithms in the smooth case, we tackle in our analysis the added challenge of nonsmoothness of the superquantile-based objective in both the general nonconvex and strongly convex cases.

The second line of work addressing heterogeneity involves designing new objective functions by modeling statistical heterogeneity and designing optimization algorithms. The AFL framework to minimize the worst-case error across all training clients and associated generalization bounds were given in (Mohri et al., 2019). The concurrent work of (Li et al., 2020d) proposes the q-FFL framework whose objective is inspired by fair resource allocation to minimize the L^p norm of the per-client losses. Several related works were also published following the initial presentation of the current work (Laguel et al., 2020b). A federated optimization algorithm for AFL was proposed and its convergence was analyzed in (Deng et al., 2020). Distributional robustness to affine shifts in the data was considered in (Reisizadeh et al., 2020) along with convergence guarantees. Finally, a classical risk measure, namely the entropic risk measure, was considered in (Li et al., 2021a). We note that no convergence guarantees are currently known for the stochastic optimization algorithms of (Li et al., 2020d).

6.7 Experiments

In this section, we demonstrate the effectiveness of Δ-FL in handling heterogeneity in federated learning. Our experiments were implemented in Python using automatic differentiation provided by PyTorch while the data was preprocessed using LEAF (Caldas et al., 2018). The code to reproduce our experiments can be found online.[3] We start by describing the datasets, tasks, and models in Section 6.7.1. We present numerical comparisons to several recent works – we list them in Section 6.7.2 and present the experimental results in Section 6.7.3. Finally, we demonstrate that Δ-FL provides the most favorable tradeoff between average error and the error on nonconforming clients in Section 6.7.4. Full details regarding the experiments as well as additional results are provided in the supplementary material.

6.7.1 Datasets, Tasks and Models

We consider two learning tasks. The dataset and task statistics are summarized in Table 6.1.

(a) *Character Recognition*: We use the EMNIST dataset (Cohen et al., 2017), where the input x is a 28×28 grayscale image of a handwritten character and the output y is its label (0-9, a-z, A-Z). Each client is a writer of the character x. The weight α_i assigned to author i is the number of characters written by this

[3] `https://github.com/krishnap25/simplicial-fl`

Table 6.2: 90^{th} percentile of the distribution of misclassification error (in $\%$) on the test devices. Each entry is the mean over five random seeds while the standard deviation is reported in the subscript. The boldfaced/highlighted entries denote the smallest value for each dataset-model pair.

	EMNIST		**Sent140**	
	Linear	**ConvNet**	**Linear**	**RNN**
FedAvg	$49.66_{0.67}$	$28.46_{1.07}$	$46.83_{0.54}$	$49.67_{3.95}$
FedAvg-Sub	$50.28_{0.77}$	$27.57_{0.81}$	$46.60_{0.38}$	$46.94_{3.84}$
FedProx	$49.15_{0.74}$	$27.01_{1.86}$	$46.83_{0.54}$	$49.86_{4.07}$
q-FFL	$49.90_{0.58}$	$28.02_{0.80}$	$\mathbf{46.39}_{0.40}$	$48.66_{4.68}$
Tilted-ERM	$48.59_{0.62}$	$25.46_{1.49}$	$46.69_{0.49}$	$46.54_{3.27}$
AFL	$51.62_{0.28}$	$45.08_{1.00}$	$47.52_{0.32}$	$57.78_{1.19}$
Δ-FL, $\theta = 0.8$	$49.10_{0.24}$	$26.23_{1.15}$	$46.44_{0.38}$	$\mathbf{46.46}_{4.39}$
Δ-FL, $\theta = 0.5$	$\mathbf{48.44_{0.38}}$	$\mathbf{23.69_{0.94}}$	$46.64_{0.41}$	$50.48_{8.24}$
Δ-FL, $\theta = 0.1$	$50.34_{0.95}$	$25.46_{2.77}$	$51.39_{1.07}$	$86.45_{10.95}$

author. We train both a linear model and a convolutional neural network architecture (ConvNet). The ConvNet consists of two 5×5 convolutional layers with max-pooling followed by one fully connected layer. Outputs are vectors of scores with respect to each of the 62 classes. The multinomial logistic loss is used to train both models.

(b) *Sentiment Analysis*: We use the Sent140 dataset (Go et al., 2009) where the input x is a tweet and the output $y = \pm1$ is its sentiment. Each client is a distinct Twitter user. The weight α_i assigned to user i is the number of tweets published by this user. We train both a logistic regression and a Long-Short Term Memory neural network architecture (LSTM). The LSTM is built on the GloVe embeddings of the words of the tweet (Hochreiter and Schmidhuber, 1997). The hidden dimension of the LSTM is the same as the embedding dimension, i.e., 50. We refer to the latter as "RNN". The loss used to train both models is the binary logistic loss.

6.7.2 Algorithms and Hyperparameters

We list here the recent works we perform numerical comparisons with and discuss their hyperparameters.

Algorithms. As discussed in Section 6.2, a federated learning method is characterized by the objective function as well as the federated optimization algorithm. We compare to the following:

(a) Vanilla FL objective: We consider two methods which attempt to minimize the vanilla FL objective: FedAvg (McMahan et al., 2017) and FedProx (Li et al., 2020b). The latter augments FedAvg with a proximal term for more stable optimization.

(b) Heterogeneity-aware objectives: We consider Tilted-ERM (Li et al., 2021a), which is the analogue of Δ-FL with the entropic risk measure (cf. Section 6.5) and AFL (Mohri et al., 2019), whose objective is obtained as the limit $\lim_{\theta\to0} F_\theta(w)$ of the Δ-FL objective. We also consider q-FFL (Li et al., 2020d), which raises the per-client loss F_i to the $(q+1)^{\text{th}}$ power, for some $q > 0$. We optimize q-FFL and Tilted-ERM with the federated optimization algorithms proposed in their respective papers. We use

Table 6.3: Mean of the distribution of misclassification error (in %) on the test devices. Each entry is the mean over five random seeds while the standard deviation is reported in the subscript. The boldfaced/highlighted entries denote the smallest value for each dataset-model pair.

	EMNIST		Sent140	
	Linear	**ConvNet**	**Linear**	**RNN**
FedAvg	$34.38_{0.38}$	$16.64_{0.50}$	$34.75_{0.31}$	$30.16_{0.44}$
FedAvg-Sub	$34.51_{0.47}$	$16.23_{0.23}$	$34.47_{0.03}$	$\mathbf{29.86}_{0.46}$
FedProx	$\mathbf{33.82}_{\mathbf{0.30}}$	$16.02_{0.54}$	$34.74_{0.31}$	$30.20_{0.48}$
q-FFL	$34.34_{0.33}$	$16.59_{0.30}$	$34.48_{0.06}$	$29.96_{0.56}$
Tilted-ERM	$34.02_{0.30}$	$15.68_{0.38}$	$34.70_{0.31}$	$30.04_{0.25}$
AFL	$39.33_{0.27}$	$33.01_{0.37}$	$35.98_{0.08}$	$37.74_{0.65}$
Δ-FL, $\theta = 0.8$	$34.49_{0.26}$	$16.09_{0.40}$	$\mathbf{34.41}_{0.22}$	$30.31_{0.33}$
Δ-FL, $\theta = 0.5$	$35.02_{0.20}$	$\mathbf{15.49}_{\mathbf{0.30}}$	$35.29_{0.25}$	$33.59_{2.44}$
Δ-FL, $\theta = 0.1$	$38.33_{0.48}$	$16.37_{1.03}$	$37.79_{0.89}$	$51.98_{11.81}$

q-FFL with $q = 10$ in place of AFL, as it was found to have more stable convergence with similar performance.

We compare one more baseline for the vanilla FL objective. Note that Δ-FL the weight $\pi^{(t)}$ (see line 3 of Algorithm 6.3) is sparse, i.e., it is non-zero for only some of the m selected clients, cf. Proposition 6.3. This is equivalent to a fewer number of effective clients per round, which is θm on average. We use as baseline FedAvg with θm clients per round, where m is the number of clients per round in Δ-FL; we call it FedAvg-Sub.

Hyperparameters. We fix the number of clients per round to be $m = 100$ for each dataset-model pair with the exception of Sent140-RNN, for which we use $m = 50$. We fixed an iteration budget for each dataset during which FedAvg converged. We tuned a learning rate schedule using a grid search to find the smallest terminal loss averaged over training clients for FedAvg. The same iteration budget and learning rate schedule were used for *all* other methods including Δ-FL. Each method, except FedAvg-Sub, selected m clients per round for training, as specified earlier. The regularization parameter λ, and the proximal weight of *FedProx* were tuned to minimize the 90^{th} percentile of the misclassification error on a held-out subset of training clients. We run q-FFL for $q \in \{10^{-3}, 10^{-2}, \ldots, 10\}$ and report q with the smallest 90^{th} percentile of misclassification error on *test* clients. We run Tilted-ERM with a temperature parameter $\nu \in \{0.1, 0.5, 1, 5, 10, 50, 100, 200\}$ and also report ν with the smallest 90^{th} percentile of misclassification error on *test* clients. We optimize Δ-FL with Algorithm 6.3 for conformity $\theta \in \{0.8, 0.5, 0.1\}$.

6.7.3 Experimental Results

We measure in Table 6.2 the 90^{th} percentile of the misclassification error across the test clients, as a measure of the right tail of the per-client performance. We also measure in Table 6.3 the mean error, which measures the average test performance. Our main findings are summarized below.

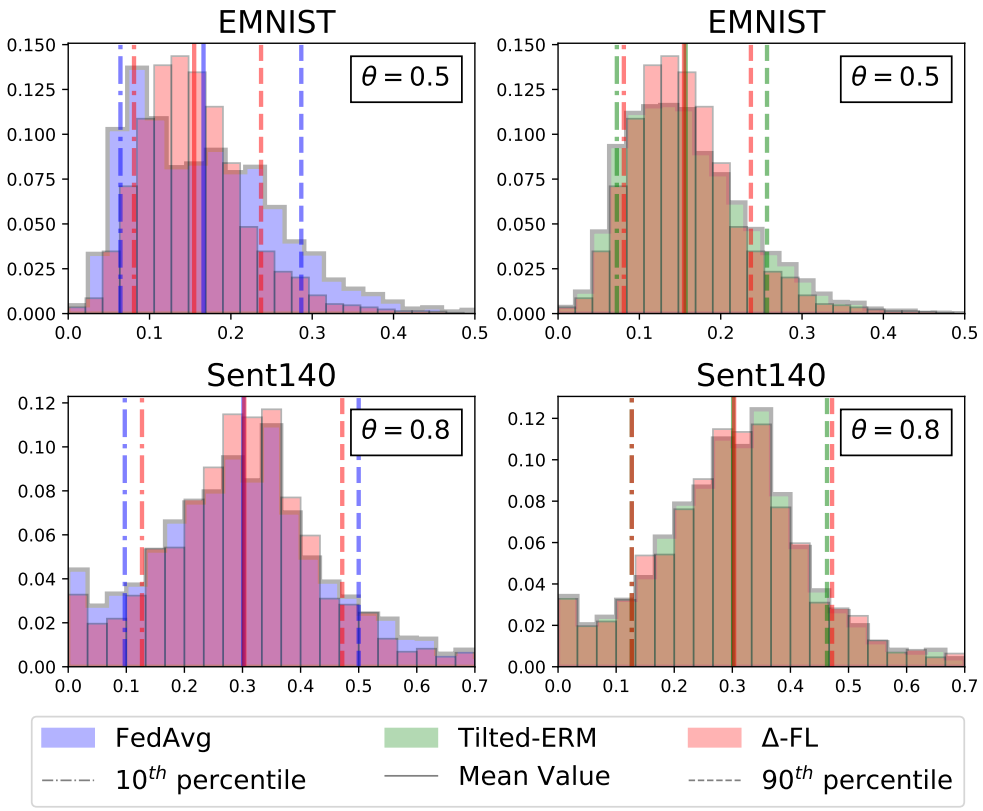


Figure 6.5: Histogram of misclassification error on test clients for the EMNIST-ConvNet and Sent140-RNN.

Δ-FL consistently achieves the smallest 90^{th} percentile error. Δ-FL achieves a 3.3% absolute (12% relative) improvement over any vanilla FL objective on EMNIST-ConvNet. Among the heterogeneity aware objectives, Δ-FL achieves 1.8% improvement over the next best objective, which is Tilted-ERM. We note that q-FFL marginally outperforms Δ-FL on Sent140-Linear, but the difference 0.05% is much smaller than the standard deviation across runs.

Δ-FL is competitive at multiple values of θ. For EMNIST-ConvNet, Δ-FL with $\theta \in \{0.5, 0.8\}$ is better in 90^{th} percentile error than *all* other methods we compare to, and Δ-FL with $\theta = 0.1$ is tied with $Tilted - ERM$, the next best method. We also empirically confirm that Δ-FL interpolates between FedAvg ($\theta \to 1$) and AFL ($\theta \to 0$).

Δ-FL works best for larger values of conformity levels. We observe that Δ-FL with $\theta = 0.1$ is unstable for Sent140-RNN. This is consistent with Theorem 6.6, which requires m to be much larger than $1/\theta$ (cf. Remark 6.7). Indeed, this can be explained by Δ-FL's sparse re-weighting, which only gives non-zero weights to $\theta m = 5$ clients on average in each round (cf. Remark 6.4).

Δ-FL is yet competitive in terms of average error. Perhaps surprisingly, Δ-FL actually gets the best test error performance on EMNIST-ConvNet and Sent140-Linear. This suggests that the average test distribution is shifted relative to the average training distribution p_α. In the other cases, we find that the

reduction in mean error is small relative to the gains in the 90^{th} percentile error compared to Vanilla FL methods.

Minimizing superquantile loss over all clients performs better than minimizing worst error over all clients. Specifically, AFL which aims to minimize the worst error among all clients, as well as other objectives which approximate it (Δ-FL with $\theta \to 0$, q-FFL with $q \to \infty$, Tilted-ERM with $\nu \to 0$) tend to achieve poor performance. We find that AFL achieves the highest error both in terms of 90^{th} percentile and the mean. Δ-FL offers a more nuanced and more effective approach via the constraint set $\text{conf}(p_\pi) \geq \theta$ than the straight pessimistic approach minimizing the worst error among all clients.

6.7.4 Exploring the Trade-off Between Average and Tail Error

We visualize in Figures 6.5 and 6.6 the distribution of test errors to explore the trade-off various methods provide between the average error and the error on nonconforming clients.

Δ-FL yields improved prediction on non-conforming clients. This can be observed from the histogram of Δ-FL in Figure 6.5, which exhibits thinner tails than FedAvg or Tilted-ERM. We see that the vanilla FL objective of FedAvg sacrifices performance on the nonconforming clients. Tilted-ERM does improve over FedAvg in this regard, but Δ-FL has a thinner right tail than Tilted-ERM, showing better handling of heterogeneity.

Δ-FL yields improved prediction on data-poor clients. We observe in Figure 6.6 that Tilted-ERM and q-FFL mainly improve the performance on data-rich clients, that is clients with lots of data. On the other hand, Δ-FL gives a greater reduction in misclassification error on data-poor clients, that is clients with little data (< 200 examples per client).

6.8 Discussion and Conclusion

We present the Δ-FL framework that operates with heterogeneous clients while still guaranteeing a minimal level of predictive performance to each individual client. We model the similarity between client data distributions using the conformity, which is a scalar summary of how closely a client conforms to the population. Δ-FL relies on a superquantile-based objective, parameterized by the conformity, to minimize the tail statistics of the prediction errors on the client data distributions. We present a federated optimization algorithm compatible with secure aggregation, which interleaves client reweighting steps with federated averaging steps. We derive finite time convergence guarantees that cover both convex and non-convex settings. Experimental results on federated learning benchmarks demonstrate superior performance of Δ-FL over state-of-the-art baselines on the upper quantiles of the error on test clients, with particular improvements on data-poor clients, while being competitive on the mean error.

The predominant approach in federated learning is to assume a set of fixed clients. As the number of smartphone users increase, an interesting scenario is one where the client pool grows dynamically over time. In particular, in the context of this chapter, one could consider new training users $1, 2, \cdots$ who arrive in an online fashion and their training distributions $p_1, p_2, \cdots$ are revealed incrementally over time. Moreover, it is also interesting to consider a scenario where the incremental population trend distribution $(1/t)\sum_{i=1}^{t} p_i$ drifts over time t. In this case, the goal is to track the best tail error minimizer over time, while minimizing the communication cost.

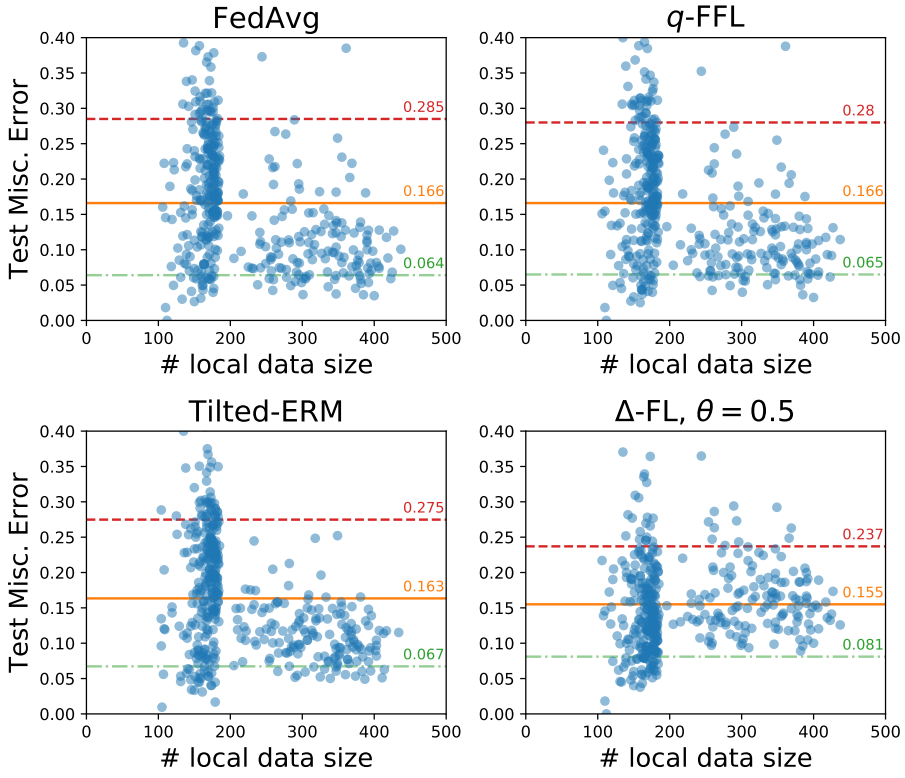


Figure 6.6: Scatter plots of misclassification error on test clients against its data size for the EMNIST-ConvNet.

Another avenue for future research is to consider the worst case over different sets of distributions in (6.6), rather than a conformity constraint. Potential examples include the χ^2-divergence $1/(2n)\sum_{i=1}^{n}(n\pi_i - 1)^2$ or the ℓ_1 norm $\|\pi - \mathbf{1}/n\|_1$. The key challenge in this case is to implement this algorithm with secure aggregation and differential privacy.

While personalization in federated learning has recently been analyzed for its ability to better handle heterogeneity (Agarwal et al., 2020; Dinh et al., 2020; Pillutla et al., 2022b), a complete analysis remains elusive. In the supervised learning setting, the data generating distribution p_i on client i can decomposed as $p_i = p_{x,i} \otimes p_{y|x,i}$, where $p_{x,i}$ is the marginal distribution over the input, and $p_{y|x,i}$ is the conditional distribution over the output given the input. The exact role of heterogeneity in each of these terms on the performance of federated learning algorithms remains an open problem. Intuitively, large heterogeneity in the conditionals $p_{y|x,i}$ requires personalization, but it is unclear the extent to which personalization helps in the case when most of the heterogeneity is in $p_{x,i}$.

Chapter 7

Conclusion

In this dissertation, we addressed some aspects of the robustness of machine learning models to heterogeneous environments across three recent revolutions in machine learning — the move to differentiable programming, the explosion of scale, and federated learning. There are several promising directions for future work building upon the material developed in this dissertation.

In Chapter 2, we studied how to incorporate combinatorial algorithms into differentiable programming pipelines in the context of structured prediction. A promising direction for future research is to incorporate logical constraints and symbolic reasoning into deep networks to augment their reasoning skills. While reductions of reasoning to combinatorial algorithms have been studied (Sun, 1994; Dechter, 1999), incorporating them into enormous language models remains a challenge. With a shift from task-specific training/finetuning to prompting (e.g., Liu et al., 2021b), designing test-time interventions to incorporate symbolic logic and reasoning into decoding algorithms is crucial, so that the resulting neurosymbolic approaches are compatible with enormous pretrained models such as GPT-3. This approach can also be generalized by basing the decisions of such a decoding algorithm on trainable parameters. These parameters can be trained based on data to demonstrate certain desired behavior, while the enormous language model is treated as fixed. Applications include training a retrieval model to augment a text generator. Finally, learning the right prompt to leverage the in-context learning of enormous models such as GPT-3 is a promising direction for future work.

In Chapter 3, we studied the trade-offs between incremental linearization algorithms and stochastic subgradient methods. We highlight two prominent directions for future work. First-order gradient-based optimization algorithms have been shown to implicitly bias the solution of the optimization (e.g., Soudry et al., 2018; Mei et al., 2018, and follow ups). An interesting direction for future work is to understand what, if any, implicit regularization effect is offered by linearization-based approaches such as the Gauss-Newton or prox-linear methods. Second, there is a need to bridge the growing divide between theory and practice in the optimization of deep networks. Prevailing theory cannot explain, for instance, the practical usefulness of algorithms such as Adam, or the commonly used learning rate decay schemes. Identifying a subclass of nonconvex functions, as well as assumptions on the structure of the data, for which theoretical results (lower and upper bounds on the rates of convergence) closely mirror practical observations can be greatly impactful. For example, the ubiquity of transformer-based models motivates the study of a single self-attention layer, which resembles classical nonparametric estimators such the k-nearest neighbor or Nadaraya-Watson kernel regression estimator, where the metric or kernel function depends on trainable parameters. It would be interesting to study such a model to explain why empirically useful practices such as using Adam or other adaptive optimizers rather than plain SGD, or starting with a smaller learning rate

and increasing it (learning rate warm-up) are beneficial.

In Chapter 4, we present MAUVE, a measure of a gap between neural text and human text. MAUVE, in directly comparing two distributions of text, offers a fundamentally different paradigm from typical reference-based measures prevalent in natural language processing. A challenging question in this context is to extend MAUVE to measure the gap between conditional distributions. This would enable automatic evaluation of dialog generation systems in a multi-round setting, a notoriously hard problem (Liu et al., 2016; Ghandeharioun et al., 2019). An orthogonal direction of significant practical interest is to reduce MAUVE's requirement of samples from the human distribution. This includes reference-free versions of MAUVE that do not require any samples from the human distribution. At a more fundamental level, the reason why decoding algorithms are required to generate text from autoregressive sequence models is not well understood. In particular, a precise and rigorous explanation of how errors accumulate when sampling text from a language model is lacking. In addition, a theoretical model that could explain how decoding algorithms mitigate this issue could be hugely influential in the design of better models and decoding algorithms.

In Chapter 5, we studied how to make the aggregation of federated learning robust to corrupted updates, followed by Chapter 6, where we presented a superquantile approach to guarantee better performance on non-conforming clients in federated learning. A more precise understanding of both these problem settings requires a careful analysis of federated averaging-type algorithms to disentangle the effect of different types of heterogeneity — that of the the marginal distribution of the inputs, the conditional distribution of the output given the input and the number of samples per client. A natural approach to this problem is to assume that the client distributions are sampled i.i.d. from a *meta-distribution*, where heterogeneity is characterized as a measure of the spread of the meta-distribution. Measures such as MAUVE from Chapter 4 can potentially be used to quantify the data heterogeneity. Secondly, evaluation is integral to learning, and the lack of enough data per client can make federated evaluation challenging. Techniques such as smoothed estimation methods (Good, 1953) offer a promising approach in this setting. More broadly, fundamental data science tasks such as hypothesis testing and causal inference remain open problems in the field of federated analytics and experimentation. Designing communication-efficient and privacy-preserving routines for this setting is an interesting direction for future work.

Bibliography

https://openai.com/blog/better-language-models/, 2019.

https://github.com/krishnap25/rfa, 2019a.

https://github.com/google-research/federated/tree/master/robust_aggregation, 2019b.

https://www.scmp.com/tech/tech-war/article/3135764/us-china-tech-war-beijing-funded-ai-researchers-surpass-google-and, 2021.

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

A. Agarwal, J. Langford, and C.-Y. Wei. Federated Residual Learning. *arXiv Preprint*, 2020.

S. K. Ainsworth, K. Lowrey, J. Thickstun, Z. Harchaoui, and S. S. Srinivasa. Faster Policy Learning with Continuous-Time Gradients. In *L4DC*, volume 144, pages 1054–1067, 2021.

S. Al-Sayed, A. M. Zoubir, and A. H. Sayed. Robust Distributed Estimation by Networked Agents. *IEEE Transactions on Signal Processing*, 65(15):3909–3921, 2017.

D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 31*, pages 4618–4628, 2018.

Z. Allen-Zhu. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. *Journal of Machine Learning Research*, 18:221:1–221:51, 2017.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *International Conference on Machine Learning*, pages 3–10, 2003.

M. Ammad-ud din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System. *arXiv Preprint*, 2019.

B. Amos. Differentiable Optimization-based Modeling for Machine Learning. *Ph.D. thesis*, 2019.

A. K. Arshadi, J. Webb, M. Salem, E. Cruz, S. Calad-Thomson, N. Ghadirian, J. Collins, E. Diez-Cecilia, B. Kelly, H. Goodarzi, and J. Yuan. Artificial Intelligence for COVID-19 Drug Discovery and Vaccine Development. *Frontiers Artif. Intell.*, 3:65, 2020.

P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent Measures of Risk. *Mathematical finance*, 9(3): 203–228, 1999.

D. Avdiukhin and S. P. Kasiviswanathan. Federated Learning under Arbitrary Communication Patterns. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 425–435. PMLR, 2021.

F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Advances in Neural Information Processing Systems*, pages 773–781, 2013.

E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How To Backdoor Federated Learning. In *Conference on Artificial Intelligence and Statistics*, volume 108, pages 2938–2948. PMLR, 2020.

J. Baker. The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975.

B. Balle, G. Barthe, M. Gaboardi, J. Hsu, and T. Sato. Hypothesis Testing Interpretations and Renyi Differential Privacy. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2496–2506. PMLR, 2020.

S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

D. Batra. An efficient message-passing algorithm for the M-best MAP problem. In *Conference on Uncertainty in Artificial Intelligence*, pages 121–130, 2012.

H. H. Bauschke, P. L. Combettes, et al. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, volume 408. Springer, 2011.

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic Differentiation in Machine Learning: a Survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017.

A. Beck and S. Sabach. Weiszfeld's Method: Old and New Results. *J. Optimization Theory and Applications*, 164(1):1–40, 2015.

A. Beck and M. Teboulle. Smoothing and First Order Methods: A Unified Framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

D. Belanger and A. McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992, 2016.

J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

R. Bellman. *Dynamic Programming*. Courier Dover Publications, 1957.

A. Belz, S. Mille, and D. M. Howcroft. Disentangling the Properties of Human Evaluation Methods: A Classification System to Support Comparability, Meta-Evaluation and Reproducibility Testing. In *Proc. of INLG*, pages 183–194, 2020.

A. Ben-Tal and M. Teboulle. An Old-New Concept of Convex Risk Measures: The Optimized Certainty Equivalent. *Mathematical Finance*, 17(3):449–476, 2007.

A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. Robust Solutions of Optimization Problems Affected by Uncertain Probabilities. *Management Science*, 59(2):341–357, 2013.

Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. *Neural Computation*, 7(6):1289–1303, 1995.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

Y. Bengio, N. Léonard, and A. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv Preprint*, 2013.

D. Berend and A. Kontorovich. The missing mass problem. *Statistics & Probability Letters*, 82(6), 2012.

Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with Differentiable Perturbed Optimizers. *arXiv preprint arXiv:2002.08676*, 2020.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo. Analyzing Federated Learning through an Adversarial Lens. In *ICML*, volume 97, pages 634–643, 2019.

M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. In *Proc. of ICLR*, 2018.

Å. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

P. Blanchard, R. Guerraoui, E. M. El Mhamdi, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30*, pages 119–129, 2017.

M. Blondel. Structured Prediction with Projection Oracles. In *Advances in Neural Information Processing Systems*, pages 12145–12156, 2019.

M. Blondel, A. F. Martins, and V. Niculae. Learning with Fenchel-Young Losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al. On the Opportunities and Risks of Foundation Models. *arXiv Preprint*, 2021.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems*, 2019.

L. Bottou and P. Gallinari. A Framework for the Cooperation of Learning Algorithms. In *Advances in Neural Information Processing Systems*, pages 781–788, 1990.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *Siam Review*, 60(2):223–311, 2018.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. In *Proc. of NeurIPS*, 2020.

Y. Bu, S. Zou, Y. Liang, and V. V. Veeravalli. Estimation of KL divergence: Optimal minimax rate. *IEEE Transactions on Information Theory*, 64(4), 2018.

M. Bun and T. Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In M. Hirt and A. D. Smith, editors, *Theory of Cryptography Conference*, volume 9985, pages 635–658, 2016.

J. Buolamwini and T. Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *FAT*, volume 81, pages 77–91, 2018.

J. V. Burke. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33(3):260–279, 1985.

J. V. Burke and M. C. Ferris. A gauss—newton method for convex composite optimization. *Mathematical Programming*, 71(2):179–194, 1995.

M. Caccia, L. Caccia, W. Fedus, H. Larochelle, J. Pineau, and L. Charlin. Language GANs Falling Short. In *Proc. of ICLR*, 2020.

S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. *arXiv Preprint*, 2018.

C. L. Canonne, G. Kamath, and T. Steinke. The Discrete Gaussian for Differential Privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

X. Cao and L. Lai. Distributed Gradient Descent Algorithm Robust to an Arbitrary Number of Byzantine Attackers. *IEEE Transactions on Signal Processing*, 67(22):5850–5864, 2019.

N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel. Extracting Training Data from Large Language Models. In *USENIX 2021*, pages 2633–2650, 2021.

A. Cassel, S. Mannor, and A. Zeevi. A General Approach to Multi-Armed Bandits Under Risk Criteria. In *Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1295–1306, 2018.

A. Celikyilmaz, E. Clark, and J. Gao. Evaluation of Text Generation: A Survey. *arXiv Preprint*, 2020.

T. H. Chan, E. Shi, and D. Song. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.*, 14 (3):26:1–26:24, 2011.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

O. Chapelle, C. B. Do, C. H. Teo, Q. V. Le, and A. J. Smola. Tighter Bounds for Structured Estimation. In *Advances in Neural Information Processing Systems*, pages 281–288, 2009.

L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos. DRACO: Byzantine-resilient Distributed Training via Redundant Gradients. In *International Conference on Machine Learning*, pages 902–911, 2018a.

M. Chen, C. Gao, Z. Ren, et al. Robust Covariance and Scatter Matrix Estimation under Huber's Contamination Model. *Annals of Statistics*, 46(5):1932–1960, 2018b.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, pages 6572–6583, 2018c.

Y. Chen, L. Su, and J. Xu. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

Y. Chen, S. Kar, and J. M. Moura. Resilient Distributed Parameter Estimation With Heterogeneous Data. *IEEE Transactions on Signal Processing*, 67(19):4918–4933, 2019.

Y. Cheng, I. Diakonikolas, and R. Ge. High-Dimensional Robust Mean Estimation in Nearly-Linear Time. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 2755–2771, 2019.

Y.-Q. Cheng, V. Wu, R. Collins, A. R. Hanson, and E. M. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. In *Visual Communications and Image Processing*, volume 2727, pages 453–463, 1996.

Y. Chow, A. Tamar, S. Mannor, and M. Pavone. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *Advances in Neural Information Processing Systems 28*, pages 1522–1530, 2015.

Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *J. Mach. Learn. Res.*, 18:167:1–167:51, 2017.

A. Cichocki and S.-i. Amari. *Adaptive blind signal and image processing: learning algorithms and applications*. John Wiley & Sons, 2002.

C. Ciliberto, L. Rosasco, and A. Rudi. A Consistent Regularization Approach for Structured Prediction. In *Advances in neural information processing systems*, pages 4412–4420, 2016.

E. Clark, A. Celikyilmaz, and N. A. Smith. Sentence Mover's Similarity: Automatic Evaluation for Multi-Sentence Texts. In *Proc. of ACL*, 2019.

E. Clark, T. August, S. Serrano, N. Haduong, S. Gururangan, and N. A. Smith. All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text. In *Proc. of ACL*, 2021.

S. Clémençon and N. Vayatis. Nonparametric estimation of the precision-recall curve. In *Proc. of ICML*, pages 185–192, 2009.

S. Clémençon and N. Vayatis. Overlaying classifiers: a practical approach to optimal scoring. *Constructive Approximation*, 32:619–648, 2010.

G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv Preprint*, 2017.

M. B. Cohen, Y. T. Lee, G. L. Miller, J. Pachocki, and A. Sidford. Geometric Median in Nearly Linear Time. In *Symposium on Theory of Computing*, pages 9–21, 2016.

L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai. Exploiting Shared Representations for Personalized Federated Learning. In *International Conference on Machine Learning*, volume 139, pages 2089–2099. PMLR, 2021.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9 (Aug):1775–1822, 2008.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. In *ACL*, pages 8440–8451, 2020.

G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

C. Cortes and M. Mohri. Confidence intervals for the area under the ROC curve. In *Proc. of NeurIPS*, volume 17, 2005.

B. Cox, A. Juditsky, and A. Nemirovski. Dual subgradient algorithms for large-scale nonsmooth learning problems. *Mathematical Programming*, 148(1-2):143–180, 2014.

K. Crammer and Y. Singer. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

S. Curi, K. Y. Levy, S. Jegelka, and A. Krause. Adaptive Sampling for Stochastic Risk-Averse Learning. In *Neural Information Processing Systems*, 2020.

A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, et al. Underspecification Presents Challenges for Credibility in Modern Machine Learning. *arXiv Preprint*, 2020.

H. Daumé III and D. Marcu. Learning as search optimization: approximate large margin methods for structured prediction. In *International Conference on Machine Learning*, pages 169–176, 2005.

D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2(1):25–36, 1992.

R. Dechter. Bucket Elimination: A Unifying Framework for Reasoning. *Artif. Intell.*, 113(1-2):41–85, 1999.

A. Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pages 676–684, 2016.

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

Y. Deng, M. M. Kamani, and M. Mahdavi. Distributionally Robust Federated Averaging. In *Neural Information Processing Systems*, 2020.

T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *European Conference on Computer Vision*, pages 452–466, 2010.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*, pages 4171–4186, 2019.

O. Devolder, F. Glineur, and Y. E. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146(1-2):37–75, 2014.

I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart. Robust Estimators in High Dimensions without the Computational Intractability. In *Symposium on Foundations of Computer Science*, pages 655–664, 2016.

A. Dieuleveut and K. K. Patel. Communication Trade-offs for Local-SGD with Large Step Size. In *Advances in Neural Information Processing Systems*, pages 13579–13590, 2019.

E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, J. Williams, J. Pineau, M. Burtsev, and J. Weston. The Second Conversational Intelligence Challenge (ConvAI2), 2019.

C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized Federated Learning with Moreau Envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020.

J. Djolonga and A. Krause. Differentiable Learning of Submodular Models. In *Advances in Neural Information Processing Systems*, pages 1013–1023, 2017.

J. Djolonga, M. Lucic, M. Cuturi, O. Bachem, O. Bousquet, and S. Gelly. Precision-Recall Curves Using Information Divergence Frontiers. In *Proc. of AISTATS*, pages 2550–2559, 2020.

J. Domke. Generic Methods for Optimization-Based Modeling. In *AISTATS*, volume 22, pages 318–326, 2012.

D. L. Donoho and P. J. Huber. The notion of breakdown point. *A festschrift for Erich L. Lehmann*, 157184, 1983.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of ICLR*, 2021.

D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Math. Oper. Res.*, 43(3):919–948, 2018. doi: 10.1287/moor.2017.0889.

D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1):503–558, 2019.

J. C. Duchi and H. Namkoong. Variance-based Regularization with Convex Objectives. *Journal of Machine Learning Research*, 20(68):1–55, 2019.

C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006a.

C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006b.

C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010.

C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

B. Eikema and W. Aziz. Is MAP Decoding All You Need? The Inadequacy of the Mode in Neural Machine Translation. In *Proc. of CoLING*, 2020.

European Union General Data Protection Regulation (GDPR). Regulation (EU) 2016/679 of the European Parliament on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). *Official Journal*, 119:1, 2016.

D. Evans, V. Kolesnikov, M. Rosulek, et al. A Pragmatic Introduction to Secure Multi-Party Computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

A. Fallah, A. Mokhtari, and A. E. Ozdaglar. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems*, 2020.

A. Fan, M. Lewis, and Y. N. Dauphin. Hierarchical Neural Story Generation. In *Proc. of ACL*, pages 889–898, 2018.

C. Fang, C. J. Li, Z. Lin, and T. Zhang. SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path Integrated Differential Estimator. *arXiv preprint arXiv:1807.01695*, 2018.

J. Fisher, L. Jiang, K. Pillutla, S. Swayamdipta, J. Hessel, Z. Harchaoui, and Y. Choi. On Model Revision: Content Re-mapping and Avoidance. *Preprint*, 2022.

P. Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, 2012.

N. Flerova, R. Marinescu, and R. Dechter. Searching for the M Best Solutions in Graphical Models. *Journal of Artificial Intelligence Research*, 55:889–952, 2016.

H. Föllmer and A. Schied. Convex measures of risk and trading constraints. *Finance Stochastics*, 6, 2002. doi: 10.1007/s007800200072.

H. Föllmer and A. Schied. *Stochastic finance. An introduction in discrete time.* Berlin: de Gruyter, 2016. doi: 10.1515/9783110463453.

M. Fromer and A. Globerson. An LP view of the M-best MAP problem. In *Advances in Neural Information Processing Systems*, pages 567–575, 2009.

R. Frostig, R. Ge, S. Kakade, and A. Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548, 2015.

T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor. Federated Learning: A Signal Processing Perspective. *IEEE Signal Processing Magazine*, 39(3):14–41, 2022.

M. J. F. Gales, S. Watanabe, and E. Fosler-Lussier. Structured Discriminative Models for Speech Recognition: An Overview. *IEEE Signal Processing Magazine*, 29(6):70–81, 2012.

X. Gao, H. Zhang, A. Panahi, and T. Arodz. Differentiable Combinatorial Losses through Generalized Gradients of Linear Programs. *arXiv preprint arXiv:1910.08211*, 2019.

S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *EMNLP*, pages 3356–3369, 2020.

C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.

A. Ghandeharioun, J. H. Shen, N. Jaques, C. Ferguson, N. Jones, À. Lapedriza, and R. W. Picard. Approximating Interactive Human Evaluation with Self-Play for Open-Domain Dialog Systems. In *NeurIPS*, pages 13658–13669, 2019.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

A. Go, R. Bhayani, and L. Huang. Twitter Sentiment Classification using Distant Supervision. *CS224N Project Report, Stanford*, page 2009, 2009.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4), 1953.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *Proc. of NeurIPS*, 2014.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

A. Griewank and A. Walther. *Evaluating Derivatives - Principles and Techniques of Algorithmic Differentiation (2. ed.)*. SIAM, 2008.

J. Guan and M. Huang. UNION: An Unreferenced Metric for Evaluating Open-ended Story Generation. In *Proc. of EMNLP*, pages 9157–9166, 2020.

J. Guyomarch. Warcraft II open-source map-editor, 2017. URL `http://github.com/war2/war2edit`.

F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe. Local SGD with Periodic Averaging: Tighter Analysis and Adaptive Synchronization. In *Advances in Neural Information Processing Systems*, pages 11080–11092, 2019.

P. Hämäläinen and A. Solin. Deep Residual Mixture Models. *arXiv preprint*, 2020.

T. S. Han and K. Kobayashi. *Mathematics of Information and Coding*, volume 203. American Mathematical Soc., 2007.

A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep Speech: Scaling up end-to-end speech recognition. *arXiv preprint*, 2014.

A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated Learning for Mobile Keyboard Prediction. *arXiv Preprint*, 2018.

T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar. TOXIGEN: Controlling Language Models to Generate Implied and Adversarial Toxicity. In *ACL*, 2022.

T. Hashimoto, H. Zhang, and P. Liang. Unifying human and statistical evaluation for natural language generation. In *Proc. of NAACL*, pages 1689–1701, 2019.

M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *VLDB*, 3(1):1021–1032, 2010.

T. Hazan, J. Keshet, and D. A. McAllester. Direct Loss Minimization for Structured Prediction. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 1594–1602. 2010.

T. Hazan, A. G. Schwing, and R. Urtasun. Blending Learning and Inference in Conditional Random Fields. *Journal of Machine Learning Research*, 17:237:1–237:25, 2016.

L. He, K. Lee, M. Lewis, and L. Zettlemoyer. Deep Semantic Role Labeling: What Works and What's Next. In *Annual Meeting of the Association for Computational Linguistics*, pages 473–483, 2017.

L. He, A. Bian, and M. Jaggi. COLA: Decentralized Linear Learning. In *Advances in Neural Information Processing Systems 31*, pages 4541–4551, 2018.

N. He and Z. Harchaoui. Semi-Proximal Mirror-Prox for Nonsmooth Composite Minimization. In *Advances in Neural Information Processing Systems*, pages 3411–3419, 2015.

M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, Dec 1974.

J. L. Herring, J. Nagy, and L. Ruthotto. Gauss–Newton Optimization for Phase Recovery from the Bispectrum. *IEEE Transactions on Computational Imaging*, 6:235–247, 2019.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proc. of NeurIPS*, page 6629–6640, 2017.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I: Fundamentals*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1996. ISBN 9783540568506.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pages 2305–2313, 2015.

A. Holtzman, J. Buys, M. Forbes, and Y. Choi. The Curious Case of Neural Text Degeneration. In *Proc. of ICLR*, 2020.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

D. J. Hsu and S. Sabato. Loss Minimization and Parameter Estimation with Heavy Tails. *Journal of Machine Learning Research*, 17:18:1–18:40, 2016.

W. Hsu, B. Bolte, Y. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE ACM Trans. Audio Speech Lang. Process.*, 29:3451–3460, 2021.

K. Huang and X. Fu. Low-Complexity Proximal Gauss-Newton Algorithm for Nonnegative Matrix Factorization. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu. Patient Clustering Improves Efficiency of Federated Machine Learning to Predict Mortality and Hospital stay time using Distributed Electronic Medical Records. *Journal of Biomedical Informatics*, 99:103291, 2019.

P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 03 1964.

P. J. Huber. *Robust Statistics*. Springer, 2011.

D. R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32(1):384–406, 2004.

D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In *Proc. of ACL*, pages 1808–1822, July 2020.

H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1998.

P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, V. K. Pillutla, and A. Sidford. A Markov Chain Theory Approach to Characterizing the Minimax Optimality of Stochastic Gradient Descent (for Least Squares). In *Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 2:1–2:10, 2017a.

P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18:223:1–223:42, 2017b.

F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on computing*, 22(5):1087–1116, 1993.

D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar. Adaptive Quantization of Model Updates for Communication-Efficient Federated Learning. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3110–3114. IEEE, 2021.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1): 27–59, 2009.

J. K. Johnson. *Convex relaxation methods for graphical models: Lagrangian and maximum entropy approaches*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *International Conference on Machine Learning*, pages 503–510, 2010.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, 2021.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, Pearson Education International, 2009.

D. Jurafsky, J. H. Martin, P. Norvig, and S. Russell. *Speech and Language Processing*. Pearson Education, 2014. ISBN 9780133252934.

P. Kairouz, Z. Liu, and T. Steinke. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5201–5212. PMLR, 2021a.

P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021b.

D. Kakwani, A. Kunchukuttan, S. Golla, G. N. C., A. Bhattacharyya, M. M. Khapra, and P. Kumar. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *EMNLP*, pages 4948–4961, 2020.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling Laws for Neural Language Models. *arXiv Preprint*, 2020.

S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic Controlled Averaging for Federated Learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

M. Karpinska, N. Akoury, and M. Iyyer. The Perils of Using Mechanical Turk to Evaluate Open-Ended Text Generation. In *Proc. of EMNLP*, 2021.

Y. Kassahun, B. Yu, A. T. Tibebu, D. Stoyanov, S. Giannarou, J. H. Metzen, and E. B. V. Poorten. Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions. *Int. J. Comput. Assist. Radiol. Surg.*, 11(4):553–568, 2016.

I. N. Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6(1):89–104, 1974.

S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics Speech and Signal Processing*, 35(3), 1987.

J. Keshet, D. McAllester, and T. Hazan. PAC-Bayesian approach for minimization of phoneme error rate. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2224–2227. IEEE, 2011.

A. Khaled, K. Mishchenko, and P. Richtárik. Tighter Theory for Local SGD on Identical and Heterogeneous Data. In *International Conference on Artificial Intelligence and Statistics*, 2020.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

W. Knight. A self-driving Uber has killed a pedestrian in Arizona. *Ethical Tech*, March 2018.

D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. ISBN 978-0-262-01319-2.

V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich. A Unified Theory of Decentralized SGD with Changing Topology and Local Updates. In *ICML*, 2020.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

W. Kubiak. *Proportional Optimization and Fairness*. International Series in Operations Research & Management Science. Springer US, 2008.

D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh. Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning. In *Operations Research & Management Science in the Age of Analytics*, pages 130–166. 2019.

H. W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4(1):98–107, Dec 1973.

M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From Word Embeddings to Document Distances. In *Proc. of ICML*, pages 957–966. PMLR, 2015.

T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved Precision and Recall Metric for Assessing Generative Models. In *Proc. of NeurIPS*, 2019.

S. Lacoste-Julien and M. Jaggi. On the Global Linear Convergence of Frank-Wolfe Optimization Variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

S. Lacoste-Julien, M. Schmidt, and F. Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *International Conference on Machine Learning*, pages 53–61, 2013.

J. Lafferty, A. McCallum, and F. C. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*, pages 282–289, 2001.

Y. Laguel, J. Malick, and Z. Harchaoui. First-Order Optimization for Superquantile-Based Supervised Learning. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2020a.

Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. Device Heterogeneity in Federated Learning: A Superquantile Approach. *arXiv Preprint*, 2020b.

Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. A Superquantile Approach to Federated Learning with Heterogeneous Devices. In *CISS*, pages 1–6. IEEE, 2021a.

Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. Superquantiles at Work: Machine Learning Applications and Efficient Subgradient Computation. *Set-Valued and Variational Analysis*, pages 1–30, 2021b.

C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

B. Laurent and P. Massart. Adaptive Estimation of a Quadratic Functional by Model Selection. *Annals of Statistics*, pages 1302–1338, 2000.

N. Le Roux, M. W. Schmidt, and F. R. Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2012.

R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods. *Journal of Machine Learning Research*, 19, 2018.

G. Lecué and M. Lerasle. Robust machine learning by median-of-means: theory and practice. *The Annals of Statistics*, 48(2):906–931, 2020.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

J. Lee and M. Raginsky. Minimax statistical learning with Wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 2687–2696, 2018.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

D. Levy, Y. Carmon, J. C. Duchi, and A. Sidford. Large-Scale Methods for Distributionally Robust Optimization. In *Neural Information Processing Systems*, 2020a.

D. Levy, Y. Carmon, J. C. Duchi, and A. Sidford. Large-Scale Methods for Distributionally Robust Optimization. In *Advances in Neural Information Processing Systems*, 2020b.

M. Lewis and M. Steedman. A* CCG parsing with a supertag-factored model. In *Conference on Empirical Methods in Natural Language Processing*, pages 990–1000, 2014.

L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling. RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets. In *AAAI Conference on Artificial Intelligence*, pages 1544–1551. AAAI Press, 2019.

T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020a.

T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated Optimization in Heterogeneous Networks. In *MLSys*. 2020b.

T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*, 2020c.

T. Li, M. Sanjabi, and V. Smith. Fair Resource Allocation in Federated Learning. In *ICLR*, 2020d.

T. Li, A. Beirami, M. Sanjabi, and V. Smith. Tilted Empirical Risk Minimization. In *International Conference on Learning Representations*, 2021a.

T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. 2021b.

X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the Convergence of FedAvg on Non-IID Data. In *ICLR*, 2020e.

C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.

H. Lin, J. Mairal, and Z. Harchaoui. A Universal Catalyst for First-Order Optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

H. Lin, J. Mairal, and Z. Harchaoui. Catalyst Acceleration for First-Order Convex Optimization: From Theory to Practice. *Journal of Machine Learning Research*, 18(1):7854–7907, 2018.

S. Lin, G. Yang, and J. Zhang. A Collaborative Learning Framework via Federated Meta-Learning. In *IEEE International Conference on Distributed Computing Systems*, pages 289–299. IEEE, 2020.

C. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *EMNLP*, pages 2122–2132, 2016.

L. Liu, K. Pillutla, S. Welleck, S. Oh, Y. Choi, and Z. Harchaoui. Divergence Frontiers for Generative Models: Sample Complexity, Quantization Effects, and Frontier Integrals. In *NeurIPS*, 2021a.

P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv Preprint*, 2021b.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv Preprint*, 2019.

D. Lopez-Paz and M. Oquab. Revisiting Classifier Two-Sample Tests. In *Proc. of ICLR*, 2017.

H. P. Lopuhaa and P. J. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *Annals of Statistics*, 19(1):229–248, 03 1991.

Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *ACL*, pages 8086–8098, 2022.

G. Lugosi and S. Mendelson. Risk minimization by median-of-means tournaments. *Journal of the European Mathematical Society*, 22(3):925–965, 2019a.

G. Lugosi and S. Mendelson. Regularization, sparse recovery, and median-of-means tournaments. *Bernoulli*, 25(3):2075–2106, 2019b.

C. Ma, J. Konečný, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takác. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

J. Mairal. Optimization with First-Order Surrogate Functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001. ISBN 978-0-262-13360-9.

J. I. Marden. *Analyzing and modeling rank data*, volume 64 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1995. ISBN 0-412-99521-2.

A. A. Markov. An example of statistical investigation of the text "Eugene Onegin" concerning the connection of samples in chains. (In Russian.). *Bulletin of the Imperial Academy of Sciences of St. Petersburg*, 7 (3):153–162, 1913.

R. Maronna, D. Martin, and V. Yohai. *Robust Statistics: Theory and Methods*. Wiley, 2006.

A. Martins and R. Astudillo. From Softmax to Sparsemax: A Sparse model of Attention and Multi-label Classification. In *Proc. of ICML*, pages 1614–1623. PMLR, 2016.

P. H. Martins, Z. Marinho, and A. F. T. Martins. Sparse Text Generation. In *Proc. EMNLP*, pages 4252–4273, 2020.

L. Massarelli, F. Petroni, A. Piktus, M. Ott, T. Rocktäschel, V. Plachouras, F. Silvestri, and S. Riedel. How Decoding Strategies Affect the Verifiability of Generated Text. *arXiv preprint arXiv:1911.03587*, 2019.

D. McAllester and J. Keshet. Generalization Bounds and Consistency for Latent Structural Probit and Ramp Loss. In *Advances in Neural Information Processing Systems 24*, pages 2205–2212. 2011.

D. Mcallester and L. Ortiz. Concentration inequalities for the missing mass and for histogram rule error. *Journal of Machine Learning Research*, 4, 2003.

D. A. McAllester and R. E. Schapire. On the convergence rate of Good-Turing estimators. In *COLT*, 2000.

R. J. McEliece, D. J. C. MacKay, and J. Cheng. Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

R. Mehta, K. Pillutla, V. Roulet, and Z. Harchaoui. Stochastic Ordered Empirical Risk Minimization. *Preprint*, 2022.

S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

A. Mensch and M. Blondel. Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, pages 3459–3468, 2018.

O. Meshi, T. S. Jaakkola, and A. Globerson. Convergence Rate Analysis of MAP Coordinate Minimization Algorithms. In *Advances in Neural Information Processing Systems*, pages 3023–3031, 2012.

R. Metz. Microsoft's neo-Nazi sexbot was a great lesson for makers of AI assistants. *Artificial Intelligence*, March 2018.

K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Springer Science & Business Media, 2012.

T. Mihaylova, V. Niculae, and A. F. T. Martins. Understanding the Mechanics of SPIGOT: Surrogate Gradients for Latent Structure Learning. In *EMNLP*, 2020.

T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010*, pages 1045–1048. ISCA, 2010.

J. Mills, J. Hu, and G. Min. Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT. *IEEE Internet Things J.*, 7(7):5986–5994, 2020.

S. Minsker. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

S. Minsker. Uniform Bounds for Robust Mean Estimators. *arXiv Preprint*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

M. Mohammadi Amiri and D. Gündüz. Machine Learning at the Wireless Edge: Distributed Stochastic Gradient Descent Over-the-Air. *IEEE Transactions on Signal Processing*, 68:2155–2169, 2020.

M. Mohri, G. Sivek, and A. T. Suresh. Agnostic Federated Learning. In *International Conference on Machine Learning*, volume 97, pages 4615–4625, 2019.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

A. S. Nemirovski and D. B. Yudin. Problem Complexity and Method Efficiency in Optimization. 1983.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005a.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005b.

Y. Nesterov. Modified Gauss–Newton scheme with worst case guarantees for global performance. *Optimisation methods and software*, 22(3):469–483, 2007.

Y. Nesterov. Random gradient-free minimization of convex functions. Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011.

Y. Nesterov. *Introductory Lectures on Convex Optimization Vol. I: Basic course*, volume 87. Springer Science & Business Media, 2013.

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A Novel Method for Machine Learning Problems using Stochastic Recursive Gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

V. Niculae, A. F. Martins, M. Blondel, and C. Cardie. SparseMAP: Differentiable Sparse Structured Inference. In *International Conference on Machine Learning*, pages 3796–3805, 2018.

D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173, 1998.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

J. Novikova, O. Dušek, A. Cercas Curry, and V. Rieser. Why We Need New Evaluation Metrics for NLG. In *Proc. of EMNLP*, 2017.

E. A. Nurminskii. The quasigradient method for the solving of the nonlinear programming problems. *Cybernetics*, 9(1):145–150, 1973.

J. Opitz and A. Frank. Towards a Decomposable Metric for Explainable Evaluation of Text Generation from AMR. In *Proc. of EACL*, pages 1504–1518, 2021.

A. Orlitsky and A. T. Suresh. Competitive distribution estimation: Why is Good-Turing good. In *NeurIPS*, 2015.

A. Orlitsky, N. P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. In *FOCS*, 2003.

A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. Dokania, and S. Lacoste-Julien. Minding the gaps for block Frank-Wolfe optimization of structured SVMs. In *International Conference on Machine Learning*, pages 593–602, 2016.

A. Osokin, F. Bach, and S. Lacoste-Julien. On structured prediction theory with calibrated convex surrogate losses. In *Advances in Neural Information Processing Systems 30*, pages 302–313. 2017.

T. Ouyang, D. Rybach, F. Beaufays, and M. Riley. Mobile Keyboard Input Decoding with Finite-State Transducers. arXiv Preprint 1704.03987, 2017.

B. Palaniappan and F. Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1408–1416, 2016.

V. M. Panaretos and Y. Zemel. *An Invitation to Statistics in Wasserstein Space*. Springer Nature, 2020.

A. Pantelidou and A. Ephremides. *Scheduling in Wireless Networks*. Foundations and trends in networking. Now Publishers, 2011.

A. Pantelopoulos and N. G. Bourbakis. A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, 2009.

G. Papandreou and A. L. Yuille. Perturb-and-MAP Random Fields: Using Discrete Optimization to Learn and Sample from Energy Models. In *2011 International Conference on Computer Vision*, pages 193–200. IEEE, 2011.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*, pages 311–318, 2002.

C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst for gradient-based nonconvex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 613–622, 2018.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

H. Peng, S. Thomson, and N. A. Smith. Backpropagating through Structured Argmax using a SPIGOT. In *ACL*, pages 1863–1873, 2018.

J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. In *Proc. of NAACL*, pages 2227–2237, 2018.

K. Pillutla, V. Roulet, S. M. Kakade, and Z. Harchaoui. A Smoother Way to Train Structured Prediction Models. In *NeurIPS*, 2018. URL `https://arxiv.org/pdf/1902.03228.pdf`.

K. Pillutla, Y. Laguel, J. Malick, and Z. Harchaoui. Federated Learning with Heterogeneous Devices: A Superquantile Optimization Approach. *arXiv Preprint 2112.09429*, 2021a.

K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui. MAUVE: Measuring the Gap Between Neural Text and Human Text with Divergence Frontiers. In *NeurIPS*, 2021b.

K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust Aggregation for Federated Learning. *IEEE Transactions on Signal Processing*, 2022a.

K. Pillutla, K. Malik, A. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao. Federated Learning with Partial Model Personalization. In *ICML*, volume 162, pages 17716–17758, 2022b.

K. Pillutla, V. Roulet, S. M. Kakade, and Z. Harchaoui. The Trade-offs of Incremental Linearization Algorithms for Nonsmooth Composite Problems. *Preprint*, 2022c.

M. V. Pogancic, A. Paulus, V. Musil, G. Martius, and M. Rolinek. Differentiation of Blackbox Combinatorial Solvers. In *ICLR*, 2020.

S. Prabhumoye, A. W. Black, and R. Salakhutdinov. Exploring Controllable Text Generation Techniques. In *COLING*, pages 1–14, 2020.

L. Qin, S. Welleck, D. Khashabi, and Y. Choi. COLD Decoding: Energy-based Constrained Text Generation with Langevin Dynamics. *arXiv Preprint*, 2022.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9, 2019.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proc. of ICML*, volume 139, pages 8748–8763, 2021.

H. Rashkin, A. Celikyilmaz, Y. Choi, and J. Gao. PlotTMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. *arXiv Preprint*, 2020.

N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum Margin Planning. In *International Conference Machine Learning*, volume 148, pages 729–736, 2006.

N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. (Approximate) Subgradient Methods for Structured Prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 380–387, 2007.

S. J. Reddi, S. Kale, and S. Kumar. On the Convergence of Adam and Beyond. In *ICLR*, 2018.

S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. Adaptive Federated Optimization. In *International Conference on Learning Representations*, 2021.

A. Reisizadeh, F. Farnia, R. Pedarsani, and A. Jadbabaie. Robust Federated Learning: The Case of Affine Distribution Shifts. In *Neural Information Processing Systems*, 2020.

J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang. Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things. *IEEE Access*, 7:69194–69201, 2019.

A. Repetti, E. Chouzenoux, and J.-C. Pesquet. A nonconvex regularized approach for phase retrieval. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1753–1757. IEEE, 2014.

A. Rezaei, A. Liu, O. Memarrast, and B. D. Ziebart. Robust Fairness Under Covariate Shift. In *AAAI Conference on Artificial Intelligence*, pages 9419–9427, 2021.

R. T. Rockafellar and S. Uryasev. Optimization of Conditional Value-at-Risk. *Journal of Risk*, 2:21–42, 2000.

R. T. Rockafellar and S. Uryasev. Conditional Value-at-Risk for General Loss Distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

R. T. Rockafellar and S. Uryasev. The Fundamental Risk Quadrangle in Risk Management, Optimization and Statistical Estimation. *Surveys in Operations Research and Management Science*, 18(1-2):33–53, 2013.

R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

R. T. Rockafellar, S. Uryasev, and M. Zabarankin. Risk tuning with generalized linear regression. *Mathematics of Operations Research*, 33(3):712–729, 2008.

V. Roulet, D. Drusvyatskiy, S. S. Srinivasa, and Z. Harchaoui. Iterative Linearized Control: Stable Algorithms and Complexity Guarantees. In *ICML*, volume 97, pages 5518–5527, 2019.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

A. Rush. Torch-struct: Deep structured prediction library. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342. Association for Computational Linguistics, July 2020. doi: 10.18653/v1/2020.acl-demos.38.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou. Spreading vectors for similarity search. In *Proc. of ICLR*, 2019.

A. B. Sai, A. K. Mohankumar, and M. M. Khapra. A Survey of Evaluation Metrics Used for NLG Systems. *arXiv Preprint*, 2020.

M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Proc. of NeurIPS*, 2018.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs, 2016.

A. Sani, A. Lazaric, and R. Munos. Risk-Aversion in Multi-armed Bandits. In *Advances in Neural Information Processing Systems 25s*, pages 3284–3292, 2012.

M. Sap, D. Card, S. Gabriel, Y. Choi, and N. A. Smith. The Risk of Racial Bias in Hate Speech Detection. In *ACL*, pages 1668–1678, 2019.

F. Sattler, K.-R. Müller, and W. Samek. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2020.

B. Savchynskyy, J. H. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *Conference on Computer Vision and Pattern Recognition*, pages 1817–1823, 2011.

A. H. Sayed. Adaptation, Learning, and Optimization over Networks. *Foundations and Trends in Machine Learning*, 7(4-5):311–801, 2014.

T. L. Scao and A. M. Rush. How many data points is a prompt worth? In *NAACL-HLT*, pages 2627–2636, 2021.

M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4 (113-130):1, 1976.

M. Schmidt, R. Babanezhad, M. Ahmed, A. Defazio, A. Clifton, and A. Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 819–828, 2015.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing Finite Sums with the Stochastic Average Gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient Estimation Using Stochastic Computation Graphs. In *NeurIPS*, pages 3528–3536, 2015.

R. Schuster, C. Song, E. Tromer, and V. Shmatikov. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. In *USENIX, 2021*, pages 1559–1575, 2021.

T. Sellam, D. Das, and A. P. Parikh. BLEURT: Learning Robust Metrics for Text Generation. In *Proc. of ACL*, pages 7881–7892, 2020.

S. Semeniuta, A. Severyn, and S. Gelly. On Accurate Evaluation of GANs for Language Generation, 2018. arXiv Preprint.

B. Seroussi and J. Golmard. An algorithm directly finding the K most probable configurations in Bayesian networks. *International Journal of Approximate Reasoning*, 11(3):205 – 233, 1994.

T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar. Over-the-Air Federated Learning From Heterogeneous Data. *IEEE Transactions on Signal Processing*, 69:3796–3811, 2021.

W. Shakespeare. The Complete Works of William Shakespeare. URL `https://www.gutenberg.org/ebooks/100.`

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, pages 64–72, 2014.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical programming*, 127(1):3–30, 2011.

C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

H. Shimanaka, T. Kajiwara, and M. Komachi. RUSE: Regressor Using Sentence Embeddingsfor Automatic Machine Translation Evaluation. In *Proc. of Conference on Machine Translation*, pages 751–758, 2018.

A. Shimorina and A. Belz. The Human Evaluation Datasheet 1.0: A Template for Recording Details of Human Evaluation Experiments in NLP. *arXiv Preprint*, 2021.

N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui. UVeQFed: Universal Vector Quantization for Federated Learning. *IEEE Trans. Signal Process.*, 69:500–514, 2021.

A. Sideris and J. E. Bobrow. An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2275–2280. IEEE, 2005.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

A. D. Smith, A. Thakurta, and J. Upadhyay. Is Interaction Necessary for Distributed Private Learning? In *IEEE Symposium on Security and Privacy*, pages 58–77, 2017a.

S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zheng, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. *arXiv Preprint*, 2022.

V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30*, pages 4424–4434, 2017b.

V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi. COCOA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research*, 18:230, 2018.

H. O. Song, R. B. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. In *International Conference on Machine Learning*, pages 1611–1619, 2014.

Y. Song, A. Schwing, R. S. Zemel, and R. Urtasun. Training Deep Neural Networks via Direct Loss Minimization. In *International Conference on Machine Learning*, pages 2169–2177, 2016.

D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The Implicit Bias of Gradient Descent on Separable Data. *J. Mach. Learn. Res.*, 19:70:1–70:57, 2018.

S. Stanczak, M. Wiczanowski, and H. Boche. *Fundamentals of Resource Allocation in Wireless Networks: Theory and Algorithms*. Foundations in Signal Processing, Communications and Networking. Springer Berlin Heidelberg, 2009.

S. U. Stich. Local SGD Converges Fast and Communicates Little. In *International Conference on Learning Representations*, 2019.

P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia. A Formal Foundation for Secure Remote Execution of Enclaves. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450, 2017.

R. Sun. *Integrating rules and connectionism for robust commonsense reasoning*. John Wiley & Sons, Inc., 1994.

Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan. Can You Really Backdoor Federated Learning? *arXiv Preprint*, 2019.

M. Sundermeyer, H. Ney, and R. Schlüter. From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. *IEEE ACM Trans. Audio Speech Lang. Process.*, 23(3):517–529, 2015.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, pages 3104–3112, 2014.

A. Tamar, Y. Chow, M. Ghavamzadeh, and S. Mannor. Policy Gradient for Coherent Risk Measures. In *Advances in Neural Information Processing Systems 28*, pages 1468–1476, 2015.

C. Tao, L. Mou, D. Zhao, and R. Yan. RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems. In *Proc. of AAAI*, 2018.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 102. ACM, 2004a.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, pages 25–32, 2004b.

B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 73–80, 2005.

B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7(Jul):1627–1653, 2006.

C. H. Teo, S. Vishwanathan, A. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 1(55), 2009.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Conference on Natural Language Learning*, pages 142–147, 2003.

M. Tkachenko and A. Simanovsky. Named entity recognition: Exploring features. In *Empirical Methods in Natural Language Processing*, pages 118–127, 2012.

Q. Tran-Dinh, N. Pham, and L. Nguyen. Stochastic Gauss-Newton Algorithms for Nonconvex Compositional Optimization. In *International Conference on Machine Learning*, pages 9572–9582. PMLR, 2020.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, page 104, 2004.

K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *International Conference on Computer Vision*, pages 1879–1886, 2011.

Y. Vardi and C.-H. Zhang. A modified Weiszfeld algorithm for the Fermat-Weber location problem. *Mathematical Programming*, 90(3):559–566, 2001.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Proc. of NeurIPS*, pages 5998–6008, 2017.

C. Villani. *Topics in Optimal Transportation*, volume 58. American Mathematical Soc., 2021.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J. Sohn, K. Lee, and D. S. Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Neural Information Processing Systems*, 2020a.

J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *Neural Information Processing Systems*, 2020b.

J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al. A Field Guide to Federated Optimization. *arXiv Preprint*, 2021.

M. Wang, E. X. Fang, and H. Liu. Stochastic Compositional Gradient Descent: Algorithms for Minimizing Compositions of Expected-value Functions. *Mathematical Programming*, 161(1-2):419–449, 2017.

S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.*, 37(6):1205–1221, 2019.

K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

S. Welleck, I. Kulikov, J. Kim, R. Y. Pang, and K. Cho. Consistency of a Recurrent Language Model With Respect to Incomplete Decoding. In *Proc. of EMNLP*, pages 5553–5568, 2020a.

S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston. Neural Text Generation With Unlikelihood Training. In *Proc. of ICLR*, 2020b.

B. Wilder, B. Dilkina, and M. Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

R. C. Williamson and A. K. Menon. Fairness Risk Measures. In *International Conference on Machine Learning*, 2019.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proc. of EMNLP*, pages 38–45, 10 2020.

B. E. Woodworth and N. Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, pages 3639–3647, 2016.

S. J. Wright. Convergence of an Inexact Algorithm for Composite Nonsmooth Optimization. *IMA Journal of Numerical Analysis*, 10(3):299–321, 07 1990.

Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. Applied Federated Learning: Improving Google Keyboard Query Suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

C. Yanover and Y. Weiss. Finding the M most probable configurations using loopy belief propagation. In *Advances in Neural Information Processing Systems*, pages 289–296, 2004.

D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5636–5645, 2018.

Y. Yu, H. Zhao, R. C. de Lamare, Y. Zakharov, and L. Lu. Robust Distributed Diffusion Recursive Least Squares Algorithms with Side Information for Adaptive Networks. *IEEE Transactions on Signal Processing*, 67(6):1566–1581, 2019.

M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni. Bayesian Nonparametric Federated Learning of Neural Networks. In *International Conference on Machine Learning*, pages 7252–7261, 2019.

R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending Against Neural Fake News. In *Proc. of NeurIPS*, 2019.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

H. Zhang, D. Duckworth, D. Ippolito, and A. Neelakantan. Trading off diversity and quality in natural language generation. In *Proc. of HumEval*, pages 25–33, 2021.

J. Zhang and L. Xiao. Stochastic Variance-Reduced Prox-Linear Algorithms for Nonconvex Composite Optimization. *arXiv preprint arXiv:2004.04357*, 2020.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with BERT. In *Proc. of ICLR*, 2020.

X. Zhang, A. Saha, and S. Vishwanathan. Accelerated training of max-margin markov networks with kernels. *Theoretical Computer Science*, 519:88–102, 2014.

W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. In *Proc. of EMNLP*, 2019.

F. Zhou and G. Cong. On the Convergence Properties of a K-step Averaging Stochastic Gradient Descent Algorithm for Nonconvex Optimization. In *International Joint Conference on Artifical Intelligence*, pages 3219–3227, 07 2018.

Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A Benchmarking Platform for Text Generation Models, 2018.

W. Zhuang, X. Gan, Y. Wen, S. Zhang, and S. Yi. Collaborative Unsupervised Visual Representation Learning from Decentralized Data. In *ICCV*, pages 4912–4921, 2021.

Appendix A

Casimir: Full Proofs

This chapter contains details and proofs from Chapters 2 and 3. The outline is as follows.

- Section A.1: We review the properties of smoothing and prove the local smoothness bound of ℓ_2^2 smoothing.
- Section A.2: We give details of inference oracles with branch and bound search.
- Section A.3: We prove the smoothness of differentiable dynamic programming.
- Section A.4: We give the analytical form of SPIGOT (Peng et al., 2018) in our setting.
- Section A.5: We give the convergence proofs of Casimir from Chapter 2.
- Section A.6: We give the convergence proofs of the prox-linear method with incremental gradient inner loops, from Chapter 3.
- Section A.7: We give a self-contained proof of the local quadratic convergence of the exact prox-linear method (Proposition 3.1 from Chapter 3).

A.1 Smoothing Basics

Here, we concretely define the dual form of the infimal convolution smoothing used throughout this work (Beck and Teboulle, 2012; Nesterov, 2005b), and recall some of its properties.

Definition A.1. For a given closed, proper, convex function $h : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, a smoothing function $\omega : \operatorname{dom} h^* \to \mathbb{R}$ which is 1-strongly convex with respect to $\|\cdot\|_\alpha$ (for $\alpha \in \{1, 2\}$), and a parameter $\mu > 0$, define

$$h_{\mu\omega}(\boldsymbol{z}) = \max_{\boldsymbol{u} \in \operatorname{dom} h^*} \{\langle \boldsymbol{u}, \boldsymbol{z} \rangle - h^*(\boldsymbol{u}) - \mu\omega(\boldsymbol{u})\} \ .$$

as the smoothing of h by $\mu\omega$.

We now show how the parameter μ controls both the approximation error and the level of the smoothing (cf. Beck and Teboulle, 2012, Thm. 4.1, Lemma 4.2).

Property A.2. *Consider the setting of Def. A.1. The smoothing $h_{\mu\omega}$ is continuously differentiable and its gradient, given by*

$$\nabla h_{\mu\omega}(\boldsymbol{z}) = \operatorname*{arg\,max}_{\boldsymbol{u}\in\operatorname{dom} h^*} \{\langle \boldsymbol{u}, \boldsymbol{z}\rangle - h^*(\boldsymbol{u}) - \mu\omega(\boldsymbol{u})\}$$

is $1/\mu$-Lipschitz with respect to $\|\cdot\|_\alpha^*$. Moreover, letting $h_{\mu\omega} \equiv h$ for $\mu = 0$, the smoothing satisfies, for all $\mu_1 \geq \mu_2 \geq 0$,

$$(\mu_1 - \mu_2) \inf_{\boldsymbol{u}\in\operatorname{dom} h^*} \omega(\boldsymbol{u}) \leq h_{\mu_2\omega}(\boldsymbol{z}) - h_{\mu_1\omega}(\boldsymbol{z}) \leq (\mu_1 - \mu_2) \sup_{\boldsymbol{u}\in\operatorname{dom} h^*} \omega(\boldsymbol{u})\,.$$

Proof. The first part showing the gradient and smoothness is (cf. Beck and Teboulle, 2012, Thm. 4.1). The second part regarding is approximation error is a refinement of Lemma 4.2 of Beck and Teboulle (2012), which proves the following statement for the special case of $\mu_2 = 0$. We successively deduce,

$$\begin{aligned}
h_{\mu_1\omega}(\boldsymbol{z}) &= \sup_{\boldsymbol{u}\in\operatorname{dom} h^*} \{\langle \boldsymbol{u}, \boldsymbol{z}\rangle - h^*(\boldsymbol{u}) - \mu_1\omega(\boldsymbol{u})\} \\
&= \sup_{\boldsymbol{u}\in\operatorname{dom} h^*} \{\langle \boldsymbol{u}, \boldsymbol{z}\rangle - h^*(\boldsymbol{u}) - \mu_2\omega(\boldsymbol{u}) - (\mu_1 - \mu_2)\omega(\boldsymbol{u})\} \\
&\geq \sup_{\boldsymbol{u}\in\operatorname{dom} h^*} \left\{\langle \boldsymbol{u}, \boldsymbol{z}\rangle - h^*(\boldsymbol{u}) - \mu_2\omega(\boldsymbol{u}) + \inf_{\boldsymbol{u}'\in\operatorname{dom} h^*} \left\{-(\mu_1 - \mu_2)\omega(\boldsymbol{u}')\right\}\right\} \\
&= h_{\mu_2\omega}(\boldsymbol{z}) - (\mu_1 - \mu_2) \sup_{\boldsymbol{u}'\in\operatorname{dom} h^*} \omega(\boldsymbol{u}')\,,
\end{aligned}$$

since $\mu_1 - \mu_2 \geq 0$. The other side follows using instead that

$$-(\mu_1 - \mu_2)\omega(\boldsymbol{u}) \leq \sup_{\boldsymbol{u}'\in\operatorname{dom} h^*} \left\{-(\mu_1 - \mu_2)\omega(\boldsymbol{u}')\right\}\,.$$

□

Next, we recall the following equivalent definition of a matrix norm defined in Eq. (2.9).

$$\|\boldsymbol{A}\|_{\beta,\alpha} = \sup_{\boldsymbol{y}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}^\top\boldsymbol{y}\|_\beta^*}{\|\boldsymbol{y}\|_\alpha} = \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_\alpha^*}{\|\boldsymbol{x}\|_\beta} = \|\boldsymbol{A}^\top\|_{\alpha,\beta}\,. \tag{A.1}$$

Now, we consider the smoothness of a composition of a smooth function with an affine map.

Lemma A.3. *Suppose $h : \mathbb{R}^m \to \mathbb{R}$ is L-smooth with respect to $\|\cdot\|_\alpha^*$. Then, for any $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$, we have that the map $\boldsymbol{w} \mapsto h(\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b})$ is $\left(L\|\boldsymbol{A}^\top\|_{\alpha,\beta}^2\right)$-smooth with respect to $\|\cdot\|_\beta$.*

We now give a proof of the tight local smoothness bound of Lemma 2.4.

Lemma 2.4. *Let $h(\boldsymbol{z}) = \max_{j\in[m]} z_j$ be the max function and let $h_\mu \equiv h_{\mu\ell_2^2}$ denote its Euclidean smoothing. Let $g_j : \mathbb{R}^d \to \mathbb{R}$ be B_j-Lipschitz and L_j-smooth w.r.t. $\|\cdot\|_2$ for $j \in [m]$ and define $\boldsymbol{g} = (g_1, \cdots, g_m) : \mathbb{R}^d \to \mathbb{R}^m$. Further, for any $\boldsymbol{w} \in \mathbb{R}^d$, let*

$$S(\boldsymbol{w}) = \{j \in [m] \,:\, [\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}))]_j \neq 0\} = \left\{j \in [m] \,:\, [\operatorname{proj}_{\Delta^{m-1}}(\boldsymbol{g}(\boldsymbol{w})/\mu)]_j\right\}\,.$$

Then, we have for any $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$ that

$$\begin{aligned}
\|\nabla(h_\mu \circ \boldsymbol{g})(\boldsymbol{w}) - \nabla(h_\mu \circ \boldsymbol{g})(\boldsymbol{w}')\|_2 &\leq \left(\Big(\sum_{j\in S(\boldsymbol{w})\cup S(\boldsymbol{w}')} B_j^2\Big)^{1/2} + \max_{j\in[m]} L_j\right) \|\boldsymbol{w} - \boldsymbol{w}'\|_2 \\
&\leq \left(\big|S(\boldsymbol{w}) \cup S(\boldsymbol{w}')\big| \max_{j\in m} B_j + \max_{j\in m} L_j\right) \|\boldsymbol{w} - \boldsymbol{w}'\|_2\,.
\end{aligned}$$

As a consequence, $h_\mu \circ \boldsymbol{g}$ *is* $\left(\frac{1}{\mu}\sum_{j=1}^m B_j^2 + \max_{j\in[m]} L_j\right)$*-smooth w.r.t.* $\|\cdot\|_2$ *over* $\mathbb{R}^d$*.*

Proof. Fix $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$ and let $f := h_\mu \circ \boldsymbol{g}$. By the chain rule, we have $\nabla f(\boldsymbol{w}) = \nabla \boldsymbol{g}(\boldsymbol{w})^\top \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}))$, where $\nabla \boldsymbol{g}(\boldsymbol{w}) \in \mathbb{R}^{m\times d}$ is the Jacobian of $\boldsymbol{g}$. We have by the triangle inequality,

$$\begin{aligned}
&\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|_2 \\
&= \|\nabla \boldsymbol{g}(\boldsymbol{w})^\top \left(\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\right) + \left(\nabla \boldsymbol{g}(\boldsymbol{w}) - \nabla \boldsymbol{g}(\boldsymbol{w}')\right)^\top \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 \\
&\le \|\nabla \boldsymbol{g}(\boldsymbol{w})^\top \left(\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\right)\|_2 + \|\left(\nabla \boldsymbol{g}(\boldsymbol{w}) - \nabla \boldsymbol{g}(\boldsymbol{w}')\right)^\top \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 \\
&\overset{\text{(A.1)}}{\le} \|\boldsymbol{A}(\boldsymbol{w}, \boldsymbol{w}')\|_{2,2}\|\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 + \|\left(\nabla \boldsymbol{g}(\boldsymbol{w}) - \nabla \boldsymbol{g}(\boldsymbol{w}')\right)^\top \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2\,,
\end{aligned}$$

where $\boldsymbol{A}(\boldsymbol{w}, \boldsymbol{w}') \in \mathbb{R}^{m\times d}$ is such that its j^{th} row if $\nabla g_j(\boldsymbol{w})$ if $j \in S(\boldsymbol{w}) \cup S(\boldsymbol{w}')$, and is zero otherwise. We look at each term in turn. Define shorthand $S \equiv S(\boldsymbol{w}) \cup S(\boldsymbol{w}')$. Using the triangle inequality of the fact that g_j is B_j-Lipschitz, we get,

$$\begin{aligned}
\|\boldsymbol{A}(\boldsymbol{w}, \boldsymbol{w}')\|_{2,2} &\overset{\text{(A.1)}}{=} \max_{\|\boldsymbol{u}\|_2\le 1} \|\boldsymbol{A}(\boldsymbol{w}, \boldsymbol{w}')^\top \boldsymbol{u}\|_2 = \max_{\|\boldsymbol{u}\|_2\le 1} \left\|\sum_{j\in S} u_j \nabla g_j(\boldsymbol{w})\right\|_2 \\
&\le \max_{\|\boldsymbol{u}\|_2\le 1} \sum_{j\in S} |u_j| \|\nabla g_j(\boldsymbol{w})\|_2 \le \max_{\|\boldsymbol{u}\|_2\le 1} \sum_{j\in S} |u_j| B_j = \sqrt{\sum_{j\in S} B_j^2}\,.
\end{aligned}$$

Recall $\nabla h_\mu(\boldsymbol{u}) = \mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{u}/\mu)$. For $S \subset [m]$, let $\boldsymbol{g}_S(\boldsymbol{w}) \in \mathbb{R}^m$ be the restriction to $\boldsymbol{g}$ to S, i.e.,

$$[\boldsymbol{g}_S(\boldsymbol{w})]_j = g_j(\boldsymbol{w})\ \mathbb{I}(j \in S)\,.$$

Using the firm nonexpansiveness of the projection operator, we get for $S = S(\boldsymbol{w}) \cup S(\boldsymbol{w}')$ that

$$\begin{aligned}
\|\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2^2 &\le \langle \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}')), \boldsymbol{g}(\boldsymbol{w})/\mu - \boldsymbol{g}(\boldsymbol{w}')/\mu\rangle \\
&= \langle \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}')), \boldsymbol{g}_S(\boldsymbol{w})/\mu - \boldsymbol{g}_S(\boldsymbol{w}')/\mu\rangle \\
&\le \frac{1}{\mu}\|\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 \|\boldsymbol{g}_S(\boldsymbol{w}) - \boldsymbol{g}_S(\boldsymbol{w}')\|_2\,.
\end{aligned}$$

From here, the fact that f_j is B_j-Lipschitz gives

$$\begin{aligned}
\|\nabla h_\mu(\boldsymbol{g}(\boldsymbol{w})) - \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 &= \frac{1}{\mu}\left(\sum_{j\in S} \left(g_j(\boldsymbol{w}) - g_j(\boldsymbol{w}')\right)^2\right)^{1/2} \\
&\le \frac{1}{\mu}\left(\sum_{j\in S} B_j^2\right)^{1/2} \|\boldsymbol{w} - \boldsymbol{w}'\|_2\,.
\end{aligned}$$

Finally, using the fact that $\nabla h_\mu(\boldsymbol{z}) \in \Delta^{m-1}$ together with the fact that the maximum of a convex function over a compact polytope occurs at one of the vertices, we get,

$$\begin{aligned}
\|\left(\nabla \boldsymbol{g}(\boldsymbol{w}) - \nabla \boldsymbol{g}(\boldsymbol{w}')\right)^\top \nabla h_\mu(\boldsymbol{g}(\boldsymbol{w}'))\|_2 &\le \max_{\boldsymbol{u}\in\Delta^{m-1}} \|\left(\nabla \boldsymbol{g}(\boldsymbol{w}) - \nabla \boldsymbol{g}(\boldsymbol{w}')\right)^\top \boldsymbol{u}\|_2 \\
&\le \max_{j\in[m]} \|\nabla g_j(\boldsymbol{w}) - \nabla g_j(\boldsymbol{w}')\|_2 \le \max_{j\in[m]} L_j \|\boldsymbol{w} - \boldsymbol{w}'\|_2\,.
\end{aligned}$$

Putting these together completes the proof. □

Algorithm A.1. Top-K oracle from top-K outputs

Input: Augmented score function ψ, $\boldsymbol{w} \in \mathbb{R}^d$, $\mu > 0$ $Y_K = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_K\}$ such that $\boldsymbol{y}_k = \max^{(k)}_{\boldsymbol{y} \in \mathcal{Y}} \psi(\boldsymbol{y}; \boldsymbol{w})$.

Populate $\boldsymbol{z} \in \mathbb{R}^K$ so that $z_k = \frac{1}{\mu}\psi(\boldsymbol{y}_k; \boldsymbol{w})$.

Compute $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \Delta^{K-1}} \|\boldsymbol{u} - \boldsymbol{z}\|_2^2$ by a projection on the simplex.

return $s = \sum_{k=1}^K u_k^* \, \psi(\boldsymbol{y}_k; \boldsymbol{w})$ and $\boldsymbol{v} = \sum_{k=1}^K u_k^* \, \nabla_{\boldsymbol{w}} \psi(\boldsymbol{y}_k; \boldsymbol{w})$.

Algorithm A.2. Top-K best-first branch and bound search

Input: Augmented score function $\psi(\cdot, \cdot; \boldsymbol{w})$, integer $K > 0$, search space $\mathcal{Y}$, upper bound $\widehat{\psi}$, split strategy.

Initialization: Initialize priority queue with single entry $\mathcal{Y}$ with priority $\widehat{\psi}(\mathcal{Y}; \boldsymbol{w})$, and solution set $\mathcal{S}$ as the empty list.

while $|\mathcal{S}| < K$ **do**

  Pop $\widehat{\mathcal{Y}}$ from the priority queue.

  if $\widehat{\mathcal{Y}} = \{\widehat{\boldsymbol{y}}\}$ is a singleton **then**

    Append $(\widehat{\boldsymbol{y}}, \psi(\widehat{\boldsymbol{y}}; \boldsymbol{w}))$ to S.

  else

    $\mathcal{Y}_1, \mathcal{Y}_2 \leftarrow \text{split}(\widehat{\mathcal{Y}})$.

    Add $\mathcal{Y}_1$ with priority $\widehat{\psi}(\mathcal{Y}_1; \boldsymbol{w})$ and $\mathcal{Y}_2$ with priority $\widehat{\psi}(\mathcal{Y}_2; \boldsymbol{w})$ to the priority queue.

return $\mathcal{S}$.

Shown in Algorithm A.1 is the procedure to compute the outputs of the top-K oracle from the K best scoring outputs obtained, for instance, from the top-K max-product algorithm.

We now state a lemma about a Euclidean projection on the probability simplex. See (e.g., Held et al., 1974) for a proof.

Lemma A.4. *Fix some $\boldsymbol{z} \in \mathbb{R}^m$. The projection $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \Delta^{m-1}} \|\boldsymbol{u} - \boldsymbol{z}\|_2^2$ satisfies $u_i^* = (z_i/\mu + \rho^*)_+$, where ρ^* is the unique solution of ρ in the equation*

$$\sum_{i=1}^m (z_i + \rho)_+ = 1 \, .$$

A.2 Inference Oracles with Branch and Bound Search

We review the branch and bound algorithm, as well as its top-K extension. Algorithm A.2 with the input $K = 1$ is the standard best-first branch and bound search algorithm. Effectively, the top-K oracle is implemented by simply continuing the search procedure until K outputs have been produced - compare Algorithm A.2 with inputs $K = 1$ and $K > 1$. We now prove the correctness guarantee.

Proposition 2.15. *Consider an augmented score function $\psi(\cdot, \cdot, \boldsymbol{w})$, an integer $K > 0$ and a smoothing parameter $\mu > 0$. Suppose the upper bound function $\widehat{\psi}(\cdot, \cdot; \boldsymbol{w}) : \mathcal{X} \times 2^{\mathcal{Y}} \to \mathbb{R}$ satisfies the following properties:*

(a) $\widehat{\psi}(\widehat{\mathcal{Y}}; \boldsymbol{w})$ is finite for every $\widehat{\mathcal{Y}} \subseteq \mathcal{Y}$,

(b) $\widehat{\psi}(\widehat{\mathcal{Y}}; \boldsymbol{w}) \geq \max_{\boldsymbol{y} \in \widehat{\mathcal{Y}}} \psi(\boldsymbol{y}; \boldsymbol{w})$ for all $\widehat{\mathcal{Y}} \subseteq \mathcal{Y}$, and,

(c) $\widehat{\psi}(\{\boldsymbol{y}\}; \boldsymbol{w}) = \psi(\boldsymbol{y}; \boldsymbol{w})$ *for every* $\boldsymbol{y} \in \mathcal{Y}$.

Then, we have the following:

(i) *Algorithm A.2 with* $K = 1$ *is a valid implementation of the max oracle.*

(ii) *Algorithm A.2 followed by a projection onto the simplex (Algorithm A.1 in Appendix A.1) is a valid implementation of the top-*K *oracle.*

Proof. Suppose at some point during the execution of the algorithm, we have a $\widehat{\mathcal{Y}} = \{\widehat{\boldsymbol{y}}\}$ on Line 4 and that $|\mathcal{S}| = k$ for some $0 \leq k < K$. From the properties of the quality upper bound $\widehat{\psi}$, and using the fact that $\{\widehat{\boldsymbol{y}}\}$ had the highest priority in the priority queue (denoted by $(*)$), we get,

$$\begin{aligned}
\psi(\widehat{\boldsymbol{y}}; \boldsymbol{w}) &= \widehat{\psi}(\{\widehat{\boldsymbol{y}}\}; \boldsymbol{w}) \\
&\overset{(*)}{\geq} \max_{Y \in \mathcal{P}} \widehat{\psi}(Y; \boldsymbol{w}) \\
&\geq \max_{Y \in \mathcal{P}} \max_{\boldsymbol{y} \in Y} \psi(\boldsymbol{y}; \boldsymbol{w}) \\
&\overset{(\#)}{=} \max_{\boldsymbol{y} \in \mathcal{Y} - \mathcal{S}} \psi(\boldsymbol{y}; \boldsymbol{w}) \,,
\end{aligned}$$

where the equality $(\#)$ followed from the fact that any $\boldsymbol{y} \in \mathcal{Y}$ exits the priority queue only if it is added to $\mathcal{S}$. This shows that if a $\widehat{\boldsymbol{y}}$ is added to $\mathcal{S}$, it has a score that is no less than that of any $\boldsymbol{y} \in \mathcal{Y} - \mathcal{S}$. In other words, Algorithm A.2 returns the top-K highest scoring $\boldsymbol{y}$'s. □

A.3 Smoothness of Related Methods

Here, we give the proof of Proposition 2.12, an analysis of differentiable dynamic programming.

Proposition 2.12. *At a smoothing parameter* $\mu > 0$*, we have that* $f_{\mathrm{DP},\mu}$ *as defined in* (2.22) *is* 1*-Lipschitz globally. Further, it satisfies,*

$$\|\nabla f_{\mathrm{DP},\mu}(\boldsymbol{w}) - \nabla f_{\mathrm{DP},\mu}(\boldsymbol{w}')\|_2 \leq \frac{1}{\mu} \sum_{i=1}^{N} |P_i'(\boldsymbol{w}) \cup P_i'(\boldsymbol{w}')| \, \|\boldsymbol{w} - \boldsymbol{w}'\|_2 \,,$$

where $P_i'(\boldsymbol{w})$ *denotes the "active" parents of node* i *at* $\boldsymbol{w}$*, i.e.,*

$$P_i'(\boldsymbol{w}) = \left\{ j \in P_i \,:\, [\boldsymbol{u}_\star^{(i)}(\boldsymbol{w})]_j \neq 0 \right\} \,,$$

where $\boldsymbol{u}_\star^{(i)}(\boldsymbol{w})$ *denotes the argmax in the definition of* $f_i(\boldsymbol{w})$ *in* (2.22)*. In particular,* $f_{\mathrm{DP},\mu}$ *is* $\sum_{i=1}^{N} |P_i|/\mu$*-smooth w.r.t.* $\|\cdot\|_2$ *globally.*

Proof. Fix $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^d$. Let $\boldsymbol{w}_{\leq i} = \left(w_{j,i'}\right)_{i' \in [i], j \in P_{i'}}$ denote the portion of $\boldsymbol{w}$ corresponding to the scores of the of edges in the subgraph induced by nodes $[n]$. We use shorthand $P_i' = P_i'(\boldsymbol{w}) \cup P_i'(\boldsymbol{w}')$ for the active parents of node i.

Part 1. We first show that

$$|f_{i,\mu}(\boldsymbol{w}) - f_{i,\mu}(\boldsymbol{w}')| \leq B_i \|\boldsymbol{w}_{\leq i} - \boldsymbol{w}'_{\leq i}\|_2 \,, \quad \|f_{i,\mu}(\boldsymbol{w}) - f_{i,\mu}(\boldsymbol{w}')\|_2 \leq L_i \|\boldsymbol{w}_{\leq i} - \boldsymbol{w}'_{\leq i}\|_2 \,,$$

where the constants B_i, L_i satisfy the recursion $B_1 = 0 = L_1$ and for $i \geq 2$,

$$B_i \leq 1 \,, \quad \text{and,} \quad L_i \leq \frac{1}{\mu}|P'_i| + \max_{j \in P_i} L_j \,.$$

For the Lipschitz constant of f_i, we prove it inductively. The base case it obviously true since node 1 has no parents. Suppose it is true for all $j \leq i-1$. We have,

$$\begin{aligned} f_{i,\mu}(\boldsymbol{w}) - f_{i,\mu}(\boldsymbol{w}') &\leq \max_{\boldsymbol{u} \in \Delta^{|P_i|-1}} \left\{ \left\langle \boldsymbol{u}, \left(f_{j,\mu}(\boldsymbol{w}) + w_{j,i} - f_{j,\mu}(\boldsymbol{w}') - w'_{j,i} \right)_{j \in P_i} \right\rangle \right\} \\ &= \max_{j \in P_i} f_{j,\mu}(\boldsymbol{w}) + w_{j,i} - f_{j,\mu}(\boldsymbol{w}') - w'_{j,i} \\ &\leq \max_{j \in P_i} B_j \|\boldsymbol{w}_{\leq j} - \boldsymbol{w}'_{\leq j}\|_2 + |w_{ji} - w'_{ji}| \,. \end{aligned}$$

Repeating the same argument with $f_{i,\mu}(\boldsymbol{w}) - f_{i,\mu}(\boldsymbol{w}')$ and noting that the edges defining $\boldsymbol{w}_{\leq j}$ and $w_{j,i}$ are disjoint gives $B_i \leq 1 \vee \max_{j \in P_i} B_j$. This readily gives the bound $B_i \leq 1$ for all $i \in [N]$. For the smoothness, we invoke Lemma 2.4 to get

$$L_i \leq \frac{1}{\mu} \sum_{j \in P'_i} B_j^2 + \max_{j \in P_i} L_j \leq \frac{1}{\mu}|P'_i| + \max_{j \in P_i} L_j \,.$$

Part 2. We now prove by induction again that the established recurrence gives $L_i \leq \sum_{j=1}^{i} |P'_j|/\mu$. The base case is true since the first node has no parents. Suppose it is true for nodes $[i-1]$. We then deduce,

$$\begin{aligned} L_i &\leq \frac{1}{\mu}|P'_i| + \max_{j \in P_i} L_j \leq \frac{1}{\mu}|P'_i| + \max_{j \in P_i} \sum_{j'=1}^{j} \frac{1}{\mu}|P'_{j'}| \\ &\leq \frac{1}{\mu}|P'_i| + \sum_{j=1}^{i-1} \frac{1}{\mu}|P'_j| = \sum_{j=1}^{i} \frac{1}{\mu}|P'_j| \,. \end{aligned}$$

□

A.4 SPIGOT: Analytical Form

Here, we write out the analytical form of SPIGOT (Peng et al., 2018) for the minimization of the task loss (2.6).

For simplicity, fix and example $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \in \mathcal{X} \times \mathcal{Y}$ and consider the associated loss $\ell\left(\overline{\boldsymbol{y}}, \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \phi(\overline{\boldsymbol{x}}, \boldsymbol{y}; \cdot)\right)$. This loss is piecewise constant. Using a parameter $\nu > 0$, SPIGOT proposes the following surrogate gradient for this loss:

$$\widetilde{\nabla} f_{\text{Spi}}(\boldsymbol{w}; \overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} u_{\boldsymbol{y}} \nabla_{\boldsymbol{w}} \phi(\overline{\boldsymbol{x}}, \boldsymbol{y}; \boldsymbol{w}) \,, \quad \text{where,} \tag{A.2}$$

$$\boldsymbol{u} = \frac{1}{\nu}\left(\boldsymbol{e}_{\boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w})} - \text{proj}_{\Delta^{m-1}}\left(\boldsymbol{e}_{\boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w})} - (\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}))_{\boldsymbol{y} \in \mathcal{Y}} \right) \right) \,. \tag{A.3}$$

We have the following analytical form for the gradient estimator of SPIGOT.

Proposition A.5. *Fix $\nu > 0$ and an input-output pair $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$. Suppose the task loss ℓ satisfies $\ell(\boldsymbol{y}, \boldsymbol{y}') \geq 0$ for all $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$ with equality iff $\boldsymbol{y} = \boldsymbol{y}'$. Denote $\boldsymbol{y}^\star = \boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w})$. Define the set $\mathcal{Y}_s = \left\{\boldsymbol{y} \in \mathcal{Y} \setminus \{\boldsymbol{y}^\star\} : \ell(\overline{\boldsymbol{y}}, \boldsymbol{y}) \leq s\right\}$ for a parameter $s > 0$. Then, we have the following:*

(a) If $\boldsymbol{y}^\star = \overline{\boldsymbol{y}}$, then $\widetilde{\nabla} f_{\mathrm{Spi}}(\boldsymbol{w}) = \boldsymbol{0}$.

(b) If $\boldsymbol{y}^\star \neq \overline{\boldsymbol{y}}$, then there exists some scalars $c, l > 0$, and weights $\beta_{\boldsymbol{y}} \geq 0$ for $\boldsymbol{y} \in \mathcal{Y}_l$ with $\sum_{\boldsymbol{y} \in \mathcal{Y}_l} \beta_{\boldsymbol{y}} = 1$ such that

$$\widetilde{\nabla} f_{\mathrm{Spi}}(\boldsymbol{w}) = c \left(\nabla_{\boldsymbol{w}} \phi(\overline{\boldsymbol{x}}, \boldsymbol{y}^\star; \boldsymbol{w}) - \sum_{\boldsymbol{y} \in \mathcal{Y}_l} \beta_{\boldsymbol{y}} \nabla_{\boldsymbol{w}} \phi(\overline{\boldsymbol{x}}, \boldsymbol{y}; \boldsymbol{w}) \right) .$$

Furthermore, the weights $\beta_{\boldsymbol{y}}$ satisfy the following order-preserving property:

$$\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}) < \ell(\overline{\boldsymbol{y}}, \boldsymbol{y}') \iff \beta_{\boldsymbol{y}} > \beta_{\boldsymbol{y}'} ,$$

with $\beta_{\boldsymbol{y}} = \beta_{\boldsymbol{y}'}$ iff $\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}) = \ell(\overline{\boldsymbol{y}}, \boldsymbol{y}')$.

(c) In particular, if $\nu < 1$ and $\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star) < 1/\nu + 1$, then

$$\widetilde{\nabla} f_{\mathrm{Spi}}(\boldsymbol{w}) = \frac{\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)}{2} \left(\nabla_{\boldsymbol{w}} \phi(\overline{\boldsymbol{x}}, \boldsymbol{y}^\star; \boldsymbol{w}) - \nabla_{\boldsymbol{w}} \phi(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}; \boldsymbol{w}) \right) .$$

Proof. Let us label the outputs $\mathcal{Y} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_m\}$. Without loss of generality, let $\boldsymbol{y}^\star = \boldsymbol{y}_1$, and $\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}_3) \geq \cdots \geq \ell(\overline{\boldsymbol{y}}, \boldsymbol{y}_m)$. Denote $\boldsymbol{\ell} = (\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}))_{\boldsymbol{y} \in \mathcal{Y}} \in \mathbb{R}^m$. Let $\boldsymbol{z} := \boldsymbol{e}_{\boldsymbol{y}^\star} - \boldsymbol{\ell}$ and the proof requires us to reason about $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z})$. We can write

$$\boldsymbol{z} = \begin{pmatrix} 1 - \nu \ell(\boldsymbol{y}^\star, \overline{\boldsymbol{y}}) \\ -\nu \ell(\boldsymbol{y}_2, \overline{\boldsymbol{y}}) \\ \vdots \\ -\nu \ell(\boldsymbol{y}_m, \overline{\boldsymbol{y}}) \end{pmatrix} .$$

Proof of Part (a). In this case, $z_1 = 1$ and $z_j < 0$ for $2 \leq j \leq m$. We verify using Lemma A.4 that $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}) = \boldsymbol{e}_1$, and the conclusion follows.

Proof of Part (b). Suppose now that $\boldsymbol{y}^* \neq \overline{\boldsymbol{y}}$. Without loss of generality, suppose $\overline{\boldsymbol{y}} = \boldsymbol{y}_2$, so that $z_2 = 0$.

By Lemma A.4, we have that $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}) = (\boldsymbol{z} - \rho^*)_+$, where $\rho^* \in \mathbb{R}$ is the unique solution to

$$1 = \left(1 - \nu \ell(\boldsymbol{y}^\star, \overline{\boldsymbol{y}}) - \rho^*\right)_+ + (-\rho^*)_+ + \sum_{j=3}^{m} \left(- \nu \ell(\boldsymbol{y}_j, \overline{\boldsymbol{y}}) - \rho^*\right)_+ . \tag{A.4}$$

Firstly, observe that $\rho^* < 0$ so that $[\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z})]_2 > 0$. Let $j^* \geq 2$ be the smallest index such that $-\nu \ell(\boldsymbol{y}_j, \overline{\boldsymbol{y}}) - \rho^* > 0$, and let

$$l = \ell(\boldsymbol{y}_j, \overline{\boldsymbol{y}}) .$$

By definition, the only non-zero components of $\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}) = (\boldsymbol{z} - \rho^*)_+$ correspond to (a) $\boldsymbol{y}^\star$, and, (b) $\boldsymbol{y}$ such that $\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}) \leq l$, i.e., $\boldsymbol{y} \in \mathcal{Y}_l$. Therefore, the vector $\boldsymbol{u}$ from (A.2) is

$$\boldsymbol{u} = \begin{pmatrix} 1 - (z_1 - \rho^*)_+ \\ -z_2 + \rho^*, \\ \vdots \\ -z_{j^*} + \rho^* \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Set $\beta_{\boldsymbol{y}_j} = (z_j - \rho^*)/u_1$ for $\boldsymbol{y}_j \in \mathcal{Y}_l$. Since $[\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z})]_2 > 0$, we get that $[\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z})]_1 < 1$ and hence, $u_1 > 0$; thus $\beta_{\boldsymbol{y}_j}$ is well-defined. Finally, by letting $c = \big(1 - (z_1 - \rho^*)_+\big)/\nu$, we get the claimed form.

Next, we show the order property of the weights $\beta_{\boldsymbol{y}}$. Note for $\boldsymbol{y}_j \in \mathcal{Y}_l$ that

$$\beta_{\boldsymbol{y}_j} = (z_j - \rho^*)/u_1 = \big(- \nu\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}_j) - \rho^*\big)/u_1\,,$$

from which the claimed results follow.

Proof of Part (c). In this case, note that $z_1 > -\nu$, $z_2 = 0$, $z_3 < -\nu$. We can verify that $\rho^* = -\nu\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2$ satisfies (A.4). Indeed, we have that

$$z_1 - \rho^* = 1 - \frac{\nu}{2}\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star) > 1 - \frac{\nu}{2}\left(\frac{1}{\nu} + 1\right) = \frac{1}{2}(1 - \nu) > 0\,.$$

Therefore, $\rho^* = -\nu\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2$ is the unique solution of (A.4). Plugging this into (A.2) gives

$$\mathrm{proj}_{\Delta^{m-1}}(\boldsymbol{z}) = \begin{pmatrix} 1 - \nu\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2 \\ \nu\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{u} = \begin{pmatrix} \ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2 \\ -\ell(\overline{\boldsymbol{y}}, \boldsymbol{y}^\star)/2 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

which gives the desired result. □

Remark A.6. *We compare SPIGOT to the structured perceptron loss (Collins, 2002) and its ℓ_2^2-smoothing. The subdifferential of the structured percetron loss defined on example $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ is*

$$\partial f_{\mathrm{perc}}(\boldsymbol{w}) \ni \nabla_{\boldsymbol{w}}\phi(\overline{\boldsymbol{x}}, \boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w}); \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\phi(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}; \boldsymbol{w})\,.$$

That is, it compares the output $\boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w})$ of inference to the reference $\overline{\boldsymbol{y}}$. Its ℓ_2^2-smoothing takes the form

$$\nabla f_{\mathrm{perc},\mu\ell_2^2} = \sum_{\boldsymbol{y} \in \mathcal{Y}} u_{\boldsymbol{y}} \nabla_{\boldsymbol{w}}\phi(\overline{\boldsymbol{x}}, \boldsymbol{y}; \boldsymbol{w}) - \nabla_{\boldsymbol{w}}\phi(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}; \boldsymbol{w})\,,$$

where $\boldsymbol{u} = \mathrm{proj}_{\Delta^{m-1}}\Big(\big(\phi(\overline{\boldsymbol{x}}, \boldsymbol{y}; \boldsymbol{w})\big)_{\boldsymbol{y} \in \mathcal{Y}}/\mu\Big)$. Thus, it compares multiple outputs with high scores to the reference $\overline{\boldsymbol{y}}$. One the other hand, Proposition A.5 tells us that SPIGOT compares the output $\boldsymbol{y}^*(\overline{\boldsymbol{x}}; \boldsymbol{w})$ of inference to multiple reference outputs with small loss w.r.t. the true reference $\overline{\boldsymbol{y}}$.

While the ℓ_2^2 smoothing of the structured perceptron loss gives us a smooth surrogate, it is not clear if the gradient estimator given by SPIGOT is the gradient of any differentiable function (or indeed, even a continuous function).

A.5 Casimir: Convergence Proofs

This section contains the full proof of convergence of Casimir in the convex case from Section 2.5. Throughout, we shall assume that ω is fixed and drop the subscript in A_ω, D_ω. Moreover, an unqualified norm $\|\cdot\|$ refers to the Euclidean norm $\|\cdot\|_2$.

A.5.1 Convergence Analysis: Proof of Theorem 2.17

We now give a proof of Theorem 2.17. The proofs of a number of technical auxiliary results are presented later in this section.

We first analyze the sequence $(\alpha_k)_{k\geq 0}$. The proof follows from the algebra of Eq. (2.32) and has been given in Section A.5.2.

Lemma A.7. *Given a positive, non-decreasing sequence $(\kappa_k)_{k\geq 1}$ and $\lambda \geq 0$, consider the sequence $(\alpha_k)_{k\geq 0}$ defined by (2.32), where $\alpha_0 \in (0,1)$ such that $\alpha_0^2 \geq \lambda/(\lambda+\kappa_1)$. Then, we have for every $k \geq 1$ that $0 < \alpha_k \leq \alpha_{k-1}$ and, $\alpha_k^2 \geq \lambda/(\lambda+\kappa_{k+1})$.*

We now characterize the effect of an approximate proximal point step on $F_{\mu\omega}$.

Lemma A.8. *Suppose $\widehat{\boldsymbol{w}} \in \mathbb{R}^d$ satisfies $F_{\mu\omega,\kappa}(\widehat{\boldsymbol{w}};\boldsymbol{z}) - \min_{\boldsymbol{w}\in\mathbb{R}^d} F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z}) \leq \widehat{\varepsilon}$ for some $\widehat{\varepsilon} > 0$. Then, for all $0 < \theta < 1$ and all $\boldsymbol{w} \in \mathbb{R}^d$, we have,*

$$F_{\mu\omega}(\widehat{\boldsymbol{w}}) + \frac{\kappa}{2}\|\widehat{\boldsymbol{w}} - \boldsymbol{z}\|_2^2 + \frac{\kappa+\lambda}{2}(1-\theta)\|\boldsymbol{w} - \widehat{\boldsymbol{w}}\|_2^2 \leq F_{\mu\omega}(\boldsymbol{w}) + \frac{\kappa}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2 + \frac{\widehat{\varepsilon}}{\theta}\,. \tag{A.5}$$

Proof. Let $\widehat{F}^* = \min_{\boldsymbol{w}\in\mathbb{R}^d} F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z})$. Let $\widehat{\boldsymbol{w}}^*$ be the unique minimizer of $F_{\mu\omega,\kappa}(\cdot\,;\boldsymbol{z})$. We have, from $(\kappa+\lambda)$-strong convexity of $F_{\mu\omega,\kappa}(\cdot\,;\boldsymbol{z})$,

$$\begin{aligned} F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z}) &\geq \widehat{F}^* + \frac{\kappa+\lambda}{2}\|\boldsymbol{w} - \widehat{\boldsymbol{w}}^*\|_2^2 \\ &\geq (F_{\mu\omega,\kappa}(\widehat{\boldsymbol{w}};\boldsymbol{z}) - \widehat{\varepsilon}) + \frac{\kappa+\lambda}{2}(1-\theta)\|\boldsymbol{w} - \widehat{\boldsymbol{w}}\|_2^2 - \frac{\kappa+\lambda}{2}\left(\frac{1}{\theta} - 1\right)\|\widehat{\boldsymbol{w}} - \widehat{\boldsymbol{w}}^*\|_2^2\,, \end{aligned}$$

where we used that $\widehat{\varepsilon}$ was sub-optimality of $\widehat{\boldsymbol{w}}$ and Lemma A.13 from Appendix A.5.4. From $(\kappa+\lambda)$-strong convexity of $F_{\mu\omega,\kappa}(\cdot;\boldsymbol{z})$, we have,

$$\frac{\kappa+\lambda}{2}\|\widehat{\boldsymbol{w}} - \widehat{\boldsymbol{w}}^*\|_2^2 \leq F_{\mu\omega,\kappa}(\widehat{\boldsymbol{w}};\boldsymbol{z}) - \widehat{F}^* \leq \widehat{\varepsilon}\,,$$

Since $(1/\theta - 1)$ is non-negative, we can plug this into the previous statement to get,

$$F_{\mu\omega,\kappa}(\boldsymbol{w};\boldsymbol{z}) \geq F_{\mu\omega,\kappa}(\widehat{\boldsymbol{w}};\boldsymbol{z}) + \frac{\kappa+\lambda}{2}(1-\theta)\|\boldsymbol{w} - \widehat{\boldsymbol{w}}\|_2^2 - \frac{\widehat{\varepsilon}}{\theta}\,.$$

Substituting the definition of $F_{\mu\omega,\kappa}(\cdot\,;\boldsymbol{z})$ from (2.30) completes the proof. □

We now define a few auxiliary sequences integral to the proof. Define sequences $(\boldsymbol{v}_k)_{k\geq 0}$, $(\gamma_k)_{k\geq 0}$,

$(\eta_k)_{k\geq 0}$, and $(\boldsymbol{r}_k)_{k\geq 1}$ as

$$\boldsymbol{v}_0 = \boldsymbol{w}_0 \tag{A.6}$$

$$\boldsymbol{v}_k = \boldsymbol{w}_{k-1} + \frac{1}{\alpha_{k-1}}(\boldsymbol{w}_k - \boldsymbol{w}_{k-1})\,,\ k \geq 1\,, \tag{A.7}$$

$$\gamma_0 = \frac{(\kappa_1 + \lambda)\alpha_0^2 - \lambda\alpha_0}{1-\alpha_0}\,, \tag{A.8}$$

$$\gamma_k = (\kappa_k + \lambda)\alpha_{k-1}^2\,,\ k \geq 1\,, \tag{A.9}$$

$$\eta_k = \frac{\alpha_k\gamma_k}{\gamma_{k+1} + \alpha_k\gamma_k}\,,\ k \geq 0\,, \tag{A.10}$$

$$\boldsymbol{r}_k = \alpha_{k-1}\boldsymbol{w}^* + (1-\alpha_{k-1})\boldsymbol{w}_{k-1}\,,\ k \geq 1\,. \tag{A.11}$$

One might recognize γ_k and $\boldsymbol{v}_k$ from their resemblance to counterparts from the proof in (Nesterov, 2013). We will need some properties of these sequences. Their proof, based solely on algebraic manipulations and Jensen's inequality, is given in Section A.5.

Lemma A.9. *For the sequences defined above, we have,*

$$\|\boldsymbol{r}_k - \boldsymbol{z}_{k-1}\|_2^2 \leq \frac{\alpha_{k-1}\lambda(1-\alpha_{k-1})}{\kappa_k}\|\boldsymbol{w}_{k-1} - \boldsymbol{w}^*\|_2^2 + \alpha_{k-1}\eta_{k-1}\|\boldsymbol{v}_{k-1} - \boldsymbol{w}^*\|_2^2\,, \tag{A.12}$$

$$\alpha_k\eta_k = (1-\alpha_k)\alpha_{k-1}^2\frac{\kappa_k + \lambda}{\kappa_{k+1}}\,. \tag{A.13}$$

Next, we define the sequence $(S_k)_{k\geq 0}$ to play the role of a potential function here.

$$\begin{aligned} S_0 &= (1-\alpha_0)(F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)) + \frac{\alpha_0\kappa_1\eta_0}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2\,, \\ S_k &= (1-\alpha_k)(F_{\mu_k\omega}(\boldsymbol{w}_k) - F_{\mu_k\omega}(\boldsymbol{w}^*)) + \frac{\alpha_k\kappa_{k+1}\eta_k}{2}\|\boldsymbol{v}_k - \boldsymbol{w}^*\|_2^2\,,\ k \geq 1\,. \end{aligned} \tag{A.14}$$

The final ingredient is the approximation property of smoothing in Eq. (2.13): for all $\mu \geq \mu' \geq 0$, we have,

$$0 \leq F_{\mu\omega}(\boldsymbol{w}) - F_{\mu'\omega}(\boldsymbol{w}) \leq (\mu - \mu')D_\omega\,. \tag{A.15}$$

We are now ready to analyze the effect of one outer loop. This lemma is the crux of the analysis.

Lemma A.10. *Suppose $F_{\mu_k\omega,\kappa_k}(\boldsymbol{w}_k;\boldsymbol{z}) - \min_{\boldsymbol{w}\in\mathbb{R}^d} F_{\mu_k\omega,\kappa_k}(\boldsymbol{w};\boldsymbol{z}) \leq \varepsilon_k$ for some $\varepsilon_k > 0$. The following statement holds for all $0 < \theta_k < 1$:*

$$\frac{S_k}{1-\alpha_k} \leq S_{k-1} + (\mu_{k-1} - \mu_k)D_\omega + \frac{\varepsilon_k}{\theta_k} - \frac{\kappa_k}{2}\|\boldsymbol{w}_k - \boldsymbol{z}_{k-1}\|_2^2 + \frac{\kappa_{k+1}\eta_k\alpha_k\theta_k}{2(1-\alpha_k)}\|\boldsymbol{v}_k - \boldsymbol{w}^*\|_2^2\,, \tag{A.16}$$

where we set $\mu_0 := 2\mu_1$.

Proof. For ease of notation, let $F_k := F_{\mu_k\omega}$, and $D := D_\omega$. By λ-strong convexity of $F_{\mu_k\omega}$, we have,

$$F_k(\boldsymbol{r}_k) \leq \alpha_{k-1}F_k(\boldsymbol{w}^*) + (1-\alpha_{k-1})F_k(\boldsymbol{w}_{k-1}) - \frac{\lambda\alpha_{k-1}(1-\alpha_{k-1})}{2}\|\boldsymbol{w}_{k-1} - \boldsymbol{w}^*\|_2^2\,. \tag{A.17}$$

We now invoke Lemma A.8 on the function $F_{\mu_k\omega,\kappa_k}(\cdot;\boldsymbol{z}_{k-1})$ with $\widehat{\varepsilon}=\varepsilon_k$ and $\boldsymbol{w}=\boldsymbol{r}_k$ to get,

$$F_k(\boldsymbol{w}_k)+\frac{\kappa_k}{2}\|\boldsymbol{w}_k-\boldsymbol{z}_{k-1}\|_2^2+\frac{\kappa_k+\lambda}{2}(1-\theta_k)\|\boldsymbol{r}_k-\boldsymbol{w}_k\|_2^2\le F_k(\boldsymbol{r}_k)+\frac{\kappa_k}{2}\|\boldsymbol{r}_k-\boldsymbol{z}_{k-1}\|_2^2+\frac{\varepsilon_k}{\theta_k}\,. \tag{A.18}$$

We now separately manipulate the left and right hand sides of (A.18). We start with the right hand side, which we call $\mathcal{R}$. Upon plugging in (A.17) and (A.12) into $\mathcal{R}$, the terms containing $\|\boldsymbol{w}_{k-1}-\boldsymbol{w}^*\|_2^2$ cancel out to yield,

$$\mathcal{R}\le(1-\alpha_{k-1})F_k(\boldsymbol{w}_{k-1})+\alpha_{k-1}F_k(\boldsymbol{w}^*)+\frac{\kappa_k\alpha_{k-1}\eta_{k-1}}{2}\|\boldsymbol{v}_{k-1}-\boldsymbol{w}^*\|_2^2+\frac{\varepsilon_k}{\theta_k}\,. \tag{A.19}$$

To move on to the left hand side, we note that

$$F_k(\boldsymbol{w}_k)-F_k(\boldsymbol{w}^*)+\frac{\kappa_k+\lambda}{2}\alpha_{k-1}^2\|\boldsymbol{v}_k-\boldsymbol{w}^*\|_2^2\overset{(A.14),(A.13)}{=}\frac{S_k}{1-\alpha_k}\,. \tag{A.20}$$

Using $\boldsymbol{r}_k-\boldsymbol{w}_k\overset{(A.7)}{=}\alpha_{k-1}(\boldsymbol{w}^*-\boldsymbol{v}_k)$, we simplify the left hand side of (A.18), which we call $\mathcal{L}$, as

$$\begin{aligned}\mathcal{L}&=F_k(\boldsymbol{w}_k)-F_k(\boldsymbol{w}^*)+\frac{\kappa_k}{2}\|\boldsymbol{w}_k-\boldsymbol{z}_{k-1}\|_2^2+\frac{\kappa_k+\lambda}{2}(1-\theta_k)\alpha_{k-1}^2\|\boldsymbol{v}_k-\boldsymbol{w}^*\|_2^2\\&\overset{(A.20)}{=}\frac{S_k}{1-\alpha_k}+F_k(\boldsymbol{w}^*)+\frac{\kappa_k}{2}\|\boldsymbol{w}_k-\boldsymbol{z}_{k-1}\|_2^2-\frac{\kappa_{k+1}\alpha_k\eta_k\theta_k}{2(1-\alpha_k)}\|\boldsymbol{v}_k-\boldsymbol{w}^*\|_2^2\,.\end{aligned} \tag{A.21}$$

In view of (A.19) and (A.21), we can simplify (A.18) as

$$\begin{aligned}&\frac{S_k}{1-\alpha_k}+\frac{\kappa_k}{2}\|\boldsymbol{w}_k-\boldsymbol{z}_{k-1}\|_2^2-\frac{\kappa_{k+1}\alpha_k\eta_k\theta_k}{2(1-\alpha_k)}\|\boldsymbol{v}_k-\boldsymbol{w}^*\|_2^2\\&\qquad\le(1-\alpha_{k-1})\left(F_k(\boldsymbol{w}_{k-1})-F_k(\boldsymbol{w}^*)\right)+\frac{\kappa_k\alpha_{k-1}\eta_{k-1}}{2}\|\boldsymbol{v}_{k-1}-\boldsymbol{w}^*\|_2^2+\frac{\varepsilon_k}{\theta_k}\,.\end{aligned} \tag{A.22}$$

We make a distinction for $k\ge2$ and $k=1$ here. For $k\ge2$, the condition that $\mu_{k-1}\ge\mu_k$ gives us,

$$F_k(\boldsymbol{w}_{k-1})-F_k(\boldsymbol{w}^*)\overset{(A.15)}{\le}F_{k-1}(\boldsymbol{w}_{k-1})-F_{k-1}(\boldsymbol{w}^*)+(\mu_{k-1}-\mu_k)D\,. \tag{A.23}$$

The right hand side of (A.22) can now be upper bounded by

$$(1-\alpha_{k-1})(\mu_{k-1}-\mu_k)D+S_{k-1}+\frac{\varepsilon_k}{\theta_k}\,,$$

and noting that $1-\alpha_{k-1}\le1$ yields (A.16) for $k\ge2$.

For $k=1$, we note that $S_{k-1}(=S_0)$ is defined in terms of $F(\boldsymbol{w})$. So we have,

$$F_1(\boldsymbol{w}_0)-F_1(\boldsymbol{w}^*)\le F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)+\mu_1D=F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)+(\mu_0-\mu_1)D\,,$$

because we used $\mu_0=2\mu_1$. This is of the same form as (A.23). Therefore, (A.16) holds for $k=1$ as well. □

We now restate and prove Theorem 2.17.

Theorem 2.17. *Consider Problem* (2.27). *Suppose $\delta_k \in [0,1)$ for all $k \geq 1$, the sequence $(\mu_k)_{k\geq 1}$ is non-negative and non-increasing, and the sequence $(\kappa_k)_{k\geq 1}$ is strictly positive and non-decreasing. Further, suppose the smoothing function $\omega : \operatorname{dom} h^* \to \mathbb{R}$ satisfies $-D_\omega \leq \omega(\boldsymbol{u}) \leq 0$ for all $\boldsymbol{u} \in \operatorname{dom} h^*$ and that $\alpha_0^2 \geq \lambda/(\lambda+\kappa_1)$. Then, the sequence $(\alpha_k)_{k\geq 0}$ generated by Algorithm 2.3 satisfies $0 < \alpha_k \leq \alpha_{k-1} < 1$ for all $k \geq 1$. Furthermore, the sequence $(\boldsymbol{w}_k)_{k\geq 0}$ of iterates generated by Algorithm 2.3 satisfies*

$$F(\boldsymbol{w}_k) - F^* \leq \frac{\mathcal{A}_0^{k-1}}{\mathcal{B}_1^k}\Delta_0 + \mu_k D_\omega + \sum_{j=1}^{k} \frac{\mathcal{A}_j^{k-1}}{\mathcal{B}_j^k}\left(\mu_{j-1} - (1-\delta_j)\mu_j\right) D_\omega \,, \tag{2.35}$$

where $\mathcal{A}_i^j := \prod_{r=i}^{j}(1-\alpha_r)$, $\mathcal{B}_i^j := \prod_{r=i}^{j}(1-\delta_r)$, $\Delta_0 := F(\boldsymbol{w}_0) - F^* + \frac{(\kappa_1+\lambda)\alpha_0^2 - \lambda\alpha_0}{2(1-\alpha_0)}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2$ and $\mu_0 := 2\mu_1$.

Proof. We continue to use shorthand $F_k := F_{\mu_k\omega}$, and $D := D_\omega$. We now apply Lemma A.10. In order to satisfy the supposition of Lemma A.10 that $\boldsymbol{w}_k$ is ε_k-suboptimal, we make the choice $\varepsilon_k = \frac{\delta_k\kappa_k}{2}\|\boldsymbol{w}_k - \boldsymbol{z}_{k-1}\|_2^2$ (cf. (2.31)). Plugging this in and setting $\theta_k = \delta_k < 1$, we get from (A.16),

$$\frac{S_k}{1-\alpha_k} - \frac{\kappa_{k+1}\eta_k\alpha_k\delta_k}{2(1-\alpha_k)}\|\boldsymbol{v}_k - \boldsymbol{w}^*\|_2^2 \leq S_{k-1} + (\mu_{k-1} - \mu_k)D \,.$$

The left hand side simplifies to $S_k\,(1-\delta_k)/(1-\alpha_k) + \delta_k(F_k(\boldsymbol{w}_k) - F_k(\boldsymbol{w}^*))$. Note that $F_k(\boldsymbol{w}_k) - F_k(\boldsymbol{w}^*) \overset{(A.15)}{\geq} F(\boldsymbol{w}_k) - F(\boldsymbol{w}^*) - \mu_k D \geq -\mu_k D$. From this, noting that $\alpha_k \in (0,1)$ for all k, we get,

$$S_k\left(\frac{1-\delta_k}{1-\alpha_k}\right) \leq S_{k-1} + \delta_k\mu_k D + (\mu_{k-1} - \mu_k)D \,,$$

or equivalently,

$$S_k \leq \left(\frac{1-\alpha_k}{1-\delta_k}\right) S_{k-1} + \left(\frac{1-\alpha_k}{1-\delta_k}\right)(\mu_{k-1} - (1-\delta_k)\mu_k)D \,.$$

Unrolling the recursion for S_k, we now have,

$$S_k \leq \left(\prod_{j=1}^{k}\frac{1-\alpha_j}{1-\delta_j}\right) S_0 + \sum_{j=1}^{k}\left(\prod_{i=j}^{k}\frac{1-\alpha_i}{1-\delta_i}\right)(\mu_{j-1} - (1-\delta_j)\mu_j)D \,. \tag{A.24}$$

Now, we need to reason about S_0 and S_k to complete the proof. To this end, consider η_0:

$$\begin{aligned}
\eta_0 &\overset{(A.10)}{=} \frac{\alpha_0\gamma_0}{\gamma_1 + \alpha_0\gamma_0} \\
&\overset{(A.8)}{=} \frac{\alpha_0\gamma_0}{(\kappa_1+\lambda)\alpha_0^2 + \frac{\alpha_0}{1-\alpha_0}\left((\kappa_1+\lambda)\alpha_0^2 - \lambda\alpha_0\right)} \\
&= \frac{\alpha_0\gamma_0(1-\alpha_0)}{(\kappa_1+\lambda)\alpha_0^2 - \lambda\alpha_0^2} = (1-\alpha_0)\frac{\gamma_0}{\kappa_1\alpha_0} \,.
\end{aligned} \tag{A.25}$$

With this, we can expand out S_0 to get

$$\begin{aligned}
S_0 &\overset{(A.14)}{=} (1-\alpha_0)\left(F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)\right) + \frac{\alpha_0\kappa_1\eta_0}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 \\
&\overset{(A.25)}{=} (1-\alpha_0)\left(F(\boldsymbol{w}_0) - F^* + \frac{\gamma_0}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2\right) \,.
\end{aligned}$$

Lastly, we reason about S_k for $k \geq 1$ as,

$$S_k \overset{\text{(A.14)}}{\geq} (1-\alpha_k)\left(F_k(\boldsymbol{w}_k) - F_k(\boldsymbol{w}^*)\right) \overset{\text{(A.15)}}{\geq} (1-\alpha_k)\left(F(\boldsymbol{w}_k) - F(\boldsymbol{w}^*) - \mu_k D\right) .$$

Plugging this into the left hand side of (A.24) completes the proof. □

A.5.2 Properties of the Auxiliary Sequences

Lemma A.7. *Given a positive, non-decreasing sequence $(\kappa_k)_{k\geq 1}$ and $\lambda \geq 0$, consider the sequence $(\alpha_k)_{k\geq 0}$ defined by (2.32), where $\alpha_0 \in (0,1)$ such that $\alpha_0^2 \geq \lambda/(\lambda+\kappa_1)$. Then, we have for every $k \geq 1$ that $0 < \alpha_k \leq \alpha_{k-1}$ and, $\alpha_k^2 \geq \lambda/(\lambda+\kappa_{k+1})$.*

Proof. It is clear that (2.32) always has a positive root, so the update is well defined. Define sequences $(c_k)_{k\geq 1}, (d_k)_{k\geq 0}$ as

$$c_k = \frac{\lambda+\kappa_k}{\lambda+\kappa_{k+1}}, \quad \text{and} \quad d_k = \frac{\lambda}{\lambda+\kappa_{k+1}} .$$

Therefore, we have that $c_k d_{k-1} = d_k$, $0 < c_k \leq 1$ and $0 \leq d_k < 1$. With these in hand, the rule for α_k can be written as

$$\alpha_k = \frac{-(c_k\alpha_{k-1}^2 - d_k) + \sqrt{(c_k\alpha_{k-1}^2 - d_k)^2 + 4c_k\alpha_{k-1}^2}}{2} . \tag{A.26}$$

We show by induction that that $d_k \leq \alpha_k^2 < 1$. The base case holds by assumption. Suppose that α_{k-1} satisfies the hypothesis for some $k \geq 1$. Noting that $\alpha_{k-1}^2 \geq d_{k-1}$ is equivalent to $c_k\alpha_{k-1}^2 - d_k \geq 0$, we get that

$$\begin{aligned}\sqrt{(c_k\alpha_{k-1}^2 - d_k)^2 + 4c_k\alpha_{k-1}^2} &\leq \sqrt{(c_k\alpha_{k-1}^2 - d_k)^2 + 4c_k\alpha_{k-1}^2 + 2(c_k\alpha_{k-1}^2 - d_k)(2\sqrt{c_k}\alpha_{k-1})} \\ &= c_k\alpha_{k-1}^2 - d_k + 2\sqrt{c_k}\alpha_{k-1} .\end{aligned} \tag{A.27}$$

We now conclude from (A.26) and (A.27) that

$$\begin{aligned}\alpha_k &\leq \frac{-(c_k\alpha_{k-1}^2 - d_k) + (c_k\alpha_{k-1}^2 - d_k + 2\sqrt{c_k}\alpha_{k-1})}{2} \\ &= \sqrt{c_k}\alpha_{k-1} \leq \alpha_{k-1} < 1 ,\end{aligned} \tag{A.28}$$

since $c_k \leq 1$ and $\alpha_{k-1} < 1$. To show the other side, we expand out (A.26) and apply (A.27) again to get

$$\begin{aligned}\alpha_k^2 - d_k &= \frac{1}{2}(c_k\alpha_{k-1}^2 - d_k)^2 + (c_k\alpha_{k-1}^2 - d_k) - \frac{1}{2}(c_k\alpha_{k-1}^2 - d_k)\sqrt{(c_k\alpha_{k-1}^2 - d_k)^2 + 4c_k\alpha_{k-1}^2} \\ &= \frac{1}{2}(c_k\alpha_{k-1}^2 - d_k)\left(2 + (c_k\alpha_{k-1}^2 - d_k) - \sqrt{(c_k\alpha_{k-1}^2 - d_k)^2 + 4c_k\alpha_{k-1}^2}\right) \\ &\geq \frac{1}{2}(c_k\alpha_{k-1}^2 - d_k)\left(2 + (c_k\alpha_{k-1}^2 - d_k) - (c_k\alpha_{k-1}^2 - d_k + 2\sqrt{c_k}\alpha_{k-1})\right) \\ &= (c_k\alpha_{k-1}^2 - d_k)(1 - \sqrt{c_k}\alpha_{k-1}) \geq 0 .\end{aligned}$$

The fact that $(\alpha_k)_{k\geq 0}$ is a non-increasing sequence follows from (A.28). □

We now state some auxiliary results.

Property A.11. *For the sequences defined in* (A.6)-(A.11)*, we have,*

$$\gamma_k = \frac{(\kappa_{k+1}+\lambda)\alpha_k^2 - \lambda\alpha_k}{1-\alpha_k}\,,\ k \geq 0\,, \tag{A.29}$$

$$\gamma_{k+1} = (1-\alpha_k)\gamma_k + \lambda\alpha_k\,,\ k \geq 0\,, \tag{A.30}$$

$$\eta_k = \frac{\alpha_k\gamma_k}{\gamma_k + \alpha_k\lambda}\,,\ k \geq 0 \tag{A.31}$$

$$\boldsymbol{z}_k = \eta_k \boldsymbol{v}_k + (1-\eta_k)\boldsymbol{w}_k\,,\ k \geq 0\,, \tag{A.32}$$

$$\alpha_{k-1} - \eta_{k-1} = \frac{\lambda}{\kappa_k}(1-\alpha_{k-1}) \tag{A.33}$$

$$\alpha_k\eta_k = (1-\alpha_k)\alpha_{k-1}^2\frac{\kappa_k+\lambda}{\kappa_{k+1}}\,. \tag{A.34}$$

The proof of this property is similar to standard results for accelerated gradient descent; see e.g., (Nesterov, 2013, Sec. 2.2).

Property A.12. *The sequence* $(\boldsymbol{r}_k)_{k\geq 1}$ *from* (A.11) *satisfies*

$$\|\boldsymbol{r}_k - \boldsymbol{z}_{k-1}\|_2^2 \leq \alpha_{k-1}(\alpha_{k-1}-\eta_{k-1})\|\boldsymbol{w}_{k-1} - \boldsymbol{w}^*\|_2^2 + \alpha_{k-1}\eta_{k-1}\|\boldsymbol{v}_{k-1} - \boldsymbol{w}^*\|_2^2\,. \tag{A.35}$$

See e.g., (Lin et al., 2018, Theorem 3) for a proof.

A.5.3 Information Based Complexity of Casimir-SVRG

Presented below are the proofs of Propositions 2.25 to 2.28 from Section 2.5.3. We use the following values of C, τ, see e.g., Hofmann et al. (2015).

$$\tau(L,\lambda) = \frac{1}{8\frac{L}{\lambda}+n} \geq \frac{1}{8\left(\frac{L}{\lambda}+n\right)} \quad \text{and} \quad C(L,\lambda) = \frac{L}{\lambda}\left(1 + \frac{n\frac{L}{\lambda}}{8\frac{L}{\lambda}+n}\right)\,.$$

Proposition 2.25. *Consider the setting of Theorem 2.17 with* $\lambda > 0$ *and fix* $\varepsilon > 0$*. If we run Algorithm 2.3 with SVRG as the inner solver with parameters:* $\mu_k = \mu = \varepsilon/10D_\omega$*,* $\kappa_k = k$ *chosen as*

$$\kappa = \begin{cases} \frac{A}{\mu n} - \lambda\,, & \text{if } \frac{A}{\mu n} > 4\lambda \\ \lambda\,, & \text{otherwise} \end{cases}\,,$$

$q = \lambda/(\lambda+\kappa)$*,* $\alpha_0 = \sqrt{q}$*, and* $\delta = \sqrt{q}/(2-\sqrt{q})$*. Then, the number of iterations* N *to obtain* $\boldsymbol{w}$ *such that* $F(\boldsymbol{w}) - F^* \leq \varepsilon$ *is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(n + \sqrt{\frac{A_\omega D_\omega n}{\lambda\varepsilon}}\right)\,.$$

Proof. We use shorthand $A := A_\omega$, $D := D_\omega$, $L_\mu = \lambda + A/\mu$ and $\Delta F_0 = F(\boldsymbol{w}_0) - F^*$. Let C, τ be the linear convergence parameters of SVRG. From Corollary 2.18, the number of outer iterations K required to obtain $F(\boldsymbol{w}_K) - F^* \leq \varepsilon$ is

$$K \leq \frac{2}{\sqrt{q}}\log\left(\frac{2\Delta F_0}{\varepsilon - c_q\mu D}\right)\,,$$

where $c_q = (3-\sqrt{q})/(1-\sqrt{q})$. From Lemma 2.23, the number T_k of inner iterations for inner loop k is, from $\delta_k = \sqrt{q}/(2-\sqrt{q})$,

$$\mathbb{E}[T_k] \leq \frac{2}{\tau(L_\mu+\kappa, \lambda+\kappa)} \log\left(\frac{8C(L_\mu+\kappa,\lambda+\kappa)}{\tau(L_\mu+\kappa,\lambda+\kappa)} \cdot \frac{L_\mu+\kappa}{\kappa} \cdot \frac{2-\sqrt{q}}{\sqrt{q}}\right).$$

Let the total number N of iterations of SVRG to obtain an iterate $\boldsymbol{w}$ that satisfies $F(\boldsymbol{w}) - F^* \leq \varepsilon$. Next, we upper bound $\mathbb{E}[N] \leq \sum_{i=1}^{K} \mathbb{E}[T_k]$ as

$$\mathbb{E}[N] \leq \frac{4}{\sqrt{q}\tau(L_\mu+\kappa,\lambda+\kappa)} \log\left(\frac{8C(L_\mu+\kappa,\lambda+\kappa)}{\tau(L_\mu+\kappa,\lambda+\kappa)} \frac{L_\mu+\kappa}{\kappa} \frac{2-\sqrt{q}}{\sqrt{q}}\right) \log\left(\frac{2(F(\boldsymbol{w}_0)-F^*)}{\varepsilon - c_q \mu D}\right). \tag{A.36}$$

Next, we shall plug in C, τ for SVRG in two different cases:

- Case 1: $A > 4\mu\lambda n$, in which case $\kappa + \lambda = A/(\mu n)$ and $q < 1/4$.
- Case 2: $A \leq 4\mu\lambda n$, in which case, $\kappa = \lambda$ and $q = 1/2$.

We first consider the term outside the logarithm. It is, up to constants,

$$\frac{1}{\sqrt{q}}\left(n + \frac{A}{\mu(\lambda+\kappa)}\right) = n\sqrt{\frac{\lambda+\kappa}{\lambda}} + \frac{A}{\mu\sqrt{\lambda(\lambda+\kappa)}}.$$

For Case 1, plug in $\kappa + \lambda = A/(\mu n)$ so this term evaluates to $\sqrt{ADn/(\lambda\varepsilon)}$. For Case 2, we use the fact that $A \leq 4\mu\lambda n$ so that this term can be upper bounded by,

$$n\left(\sqrt{\frac{\lambda+\kappa}{\lambda}} + 4\sqrt{\frac{\lambda}{\lambda+\kappa}}\right) = 3\sqrt{2}n,$$

since we chose $\kappa = \lambda$. It remains to consider the logarithmic terms. Noting that $\kappa \geq \lambda$ always, it follows that the first log term of (A.36) is clearly logarithmic in the problem parameters.

As for the second logarithmic term, we must evaluate c_q. For Case 1, we have that $q < 1/4$ so that $c_q < 5$ and $c_q \mu D < \varepsilon/2$. For Case 2, we get that $q = 1/2$ and $c_q < 8$ so that $c_q \mu D < 4\varepsilon/5$. Thus, the second log term of (A.36) is also logarithmic in problem parameters. □

Proposition 2.26. *Consider the setting of Theorem 2.17. Suppose $\lambda > 0$ and $\kappa_k = \kappa$, for all $k \geq 1$ and that α_0, $(\mu_k)_{k\geq 1}$ and $(\delta_k)_{k\geq 1}$ are chosen as in Corollary 2.19, with $q = \lambda/(\lambda+\kappa)$ and $\eta = 1 - \sqrt{q}/2$. If we run Algorithm 2.3 with SVRG as the inner solver with these parameters, the number of iterations N of SVRG required to obtain $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(n + \frac{A_\omega}{\mu(\lambda+\kappa)\varepsilon}\left(F(\boldsymbol{w}_0) - F^* + \frac{\mu D_\omega}{1-\sqrt{q}}\right)\right).$$

Proof. We continue to use shorthand $A := A_\omega$, $D := D_\omega$. First, let us consider the minimum number of outer iterations K required to achieve $F(\boldsymbol{w}_K) - F^* \leq \varepsilon$. From Corollary 2.19, if we have $\eta^{-K/2}\Delta_0 \leq \varepsilon$, or,

$$K \geq K_{\min} := \frac{\log(\Delta_0/\varepsilon)}{\log\left(1/\sqrt{\eta}\right)}.$$

For this smallest value, we have,

$$\mu_{K_{\min}} = \mu\eta^{K_{\min}/2} = \frac{\mu\varepsilon}{\Delta_0}\,. \tag{A.37}$$

Let C, τ be the linear convergence parameters of SVRG, and define $L_k := \lambda + A/\mu_k$ for each $k \geq 1$. Further, let $\mathcal{T}'$ be such that

$$\mathcal{T}' \geq \max_{k\in\{1,\cdots,K_{\min}\}} \log\left(8\frac{C(L_k+\kappa,\lambda+\kappa)}{\tau(L_k+\kappa,\lambda+\kappa)}\frac{L_k+\kappa}{\kappa\delta}\right).$$

Then, the total complexity is, from Lemma 2.23, (ignoring absolute constants)

$$\mathbb{E}[N] \leq \sum_{k=1}^{K_{\min}}\left(n + \frac{\lambda+\kappa+\frac{A}{\mu_k}}{\lambda+\kappa}\right)\mathcal{T}' \leq \left((n+1)\frac{\log\left(\frac{\Delta_0}{\varepsilon}\right)}{\log(1/\sqrt{\eta})} + \frac{A/\mu}{\lambda+\kappa}\frac{1}{1-\sqrt{\eta}}\frac{\Delta_0}{\varepsilon}\right)\mathcal{T}'.$$

It remains to bound $\mathcal{T}'$. Here, we use $\lambda + \frac{A}{\mu} \leq L_k \leq \lambda + \frac{A}{\mu_K}$ for all $k \leq K$ together with (A.37) to note that $\mathcal{T}'$ is logarithmic in $\Delta_0/\varepsilon, n, AD, \mu, \kappa, \lambda^{-1}$. □

Proposition 2.27. *Consider the setting of Theorem 2.17 and fix $\varepsilon > 0$. If we run Algorithm 2.3 with SVRG as the inner solver with parameters: $\mu_k = \mu = \varepsilon/20D_\omega$, $\alpha_0 = \frac{\sqrt{5}-1}{2}$, $\delta_k = 1/(k+1)^2$, and $\kappa_k = \kappa = A_\omega/\mu(n+1)$. Then, the number of iterations N to get a point $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(n\sqrt{\frac{F(\boldsymbol{w}_0)-F^*}{\varepsilon}} + \sqrt{A_\omega D_\omega n}\frac{\|\boldsymbol{w}_0-\boldsymbol{w}^*\|_2}{\varepsilon}\right).$$

Proof. We use shorthand $A := A_\omega$, $D := D_\omega$, $L_\mu = A/\mu$ and $\Delta F_0 = F(\boldsymbol{w}_0) - F^* + \frac{\kappa}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2$. Further, let C, τ be the linear convergence parameters of SVRG. In Corollary 2.20, the fact that $K \geq 1$ allows us to bound the contribution of the smoothing as $10\mu D$. So, we get that the number of outer iterations K required to get $F(\boldsymbol{w}_K) - F^* \leq \varepsilon$ can be bounded as

$$K + 1 \leq \sqrt{\frac{8\Delta F_0}{\varepsilon - 10\mu D}}\,.$$

Moreover, from our choice $\delta_k = 1/(k+1)^2$, the number of inner iterations T_k for inner loop k is, from Lemma 2.23,

$$\mathbb{E}[T_k] \leq \frac{2}{\tau(L_\mu+\kappa,\kappa)}\log\left(\frac{8C(L_\mu+\kappa,\kappa)}{\tau(L_\mu+\kappa,\kappa)}\cdot\frac{L_\mu+\kappa}{\kappa}\cdot\frac{8\Delta F_0}{\varepsilon-10\mu D}\right).$$

Next, we consider the total number N of iterations of SVRG to obtain an iterate $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \leq \varepsilon$. Using the fact that $\mathbb{E}[N] \leq \sum_{i=1}^{K}\mathbb{E}[T_k]$, we bound it as

$$\mathbb{E}[N] \leq \frac{1}{\tau(L_\mu+\kappa,\kappa)}\sqrt{\frac{8\Delta F_0}{\varepsilon-10\mu D}}\log\left(\frac{64C(L_\mu+\kappa,\kappa)}{\tau(L_\mu+\kappa,\kappa)}\frac{L_\mu+\kappa}{\kappa}\frac{\Delta F_0}{\varepsilon-10\mu D}\right). \tag{A.38}$$

Now, we plug into (A.38) the values of C, τ for SVRG. Note that $\kappa = L_\mu/(n+1)$. So we have,

$$\frac{1}{\tau(L_\mu+\kappa,\kappa)} = 8\left(\frac{L_\mu+\kappa}{\kappa}+n\right) = 16(n+1)\,, \text{ and,}$$
$$C(L_\mu+\kappa,\kappa) = \frac{L_\mu+\kappa}{\kappa}\left(1+\frac{n\frac{L_\mu+\kappa}{\kappa}}{8\frac{L+\kappa}{\kappa}+n}\right) \le (n+2)\left(1+\tfrac{n}{8}\right)\,.$$

It now remains to assign $\mu = \varepsilon/(20D)$ and plug C, τ from above into (A.38), noting that $\kappa = 20AD/(\varepsilon(n+1))$. □

Proposition 2.28. *Consider the setting of Theorem 2.17. Suppose $\lambda = 0$ and that α_0, $(\mu_k)_{k\ge1}$,$(\kappa_k)_{k\ge1}$ and $(\delta_k)_{k\ge1}$ are chosen as in Corollary 2.21. If we run Algorithm 2.3 with SVRG as the inner solver with these parameters, the number of iterations N of SVRG required to obtain $\boldsymbol{w}$ such that $F(\boldsymbol{w}) - F^* \le \varepsilon$ is bounded in expectation as*

$$\mathbb{E}[N] \le \widetilde{O}\left(\frac{1}{\varepsilon}\left(F(\boldsymbol{w}_0) - F^* + \kappa\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 + \mu D\right)\left(n + \frac{A_\omega}{\mu\kappa}\right)\right)\,.$$

Proof. Define short hand $A := A_\omega$, $D := D_\omega$ and

$$\Delta_0 := 2(F(\boldsymbol{w}_0) - F^*) + \kappa\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2 + 27\mu D\,. \tag{A.39}$$

From Corollary 2.21, the number of iterations K required to obtain $F(\boldsymbol{w}_K) - F^* \le \frac{\log(K+1)}{K+1}\Delta_0 \le \varepsilon$ is (see Lemma A.16),

$$K+1 = \frac{2\Delta_0}{\varepsilon}\log\frac{2\Delta_0}{\varepsilon}\,. \tag{A.40}$$

Let C, τ be such that SVRG is linearly convergent with parameters C, τ, and define $L_k := A/\mu_k$ for each $k \ge 1$. Further, let $\mathcal{T}'$ be such that

$$\mathcal{T}' \ge \max_{k\in\{1,\cdots,K\}} \log\left(8\frac{C(L_k+\kappa,\kappa)}{\tau(L_k+\kappa,\kappa)}\frac{L_k+\kappa}{\kappa\delta_k}\right)\,.$$

Clearly, $\mathcal{T}'$ is logarithmic in K, n, AD, μ, κ. From Lemma 2.23, the minimum total complexity is (ignoring absolute constants)

$$\mathbb{E}[N] = \sum_{k=1}^{K}\left(n + \frac{A/\mu_k + \kappa_k}{\kappa_k}\right)\mathcal{T}' \le \left(n+1+\frac{A}{\mu\kappa}\right)K\mathcal{T}'\,,$$

and plugging in K from (A.40) completes the proof. □

A.5.4 Some Helper Lemmas

The first lemma is a property of the squared Euclidean norm from Lin et al. (2018, Lemma 5), which we restate here.

Lemma A.13. *For any vectors, $\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{r} \in \mathbb{R}^d$, we have, for any $\theta > 0$,*

$$\|\boldsymbol{w} - \boldsymbol{z}\|^2 \ge (1-\theta)\|\boldsymbol{w} - \boldsymbol{r}\|^2 + \left(1 - \frac{1}{\theta}\right)\|\boldsymbol{r} - \boldsymbol{z}\|^2\,.$$

The next lemmas consider rates of the sequences (α_k) and (A_k) under different recursions; see (Pillutla et al., 2018) for proofs.

Lemma A.14. *Define a sequence $(\alpha_k)_{k\geq 0}$ as*

$$\alpha_0 = \frac{\sqrt{5}-1}{2}$$
$$\alpha_k^2 = (1-\alpha_k)\alpha_{k-1}^2 \,.$$

Then this sequence satisfies

$$\frac{\sqrt{2}}{k+3} \leq \alpha_k \leq \frac{2}{k+3} \,.$$

Moreover, $A_k := \prod_{j=0}^{k}(1-\alpha_k)$ satisfies

$$\frac{2}{(k+3)^2} \leq A_k \leq \frac{4}{(k+3)^2} \,.$$

Lemma A.15. *Consider a sequence $(\alpha_k)_{k\geq 0}$ defined by $\alpha_0 = \frac{\sqrt{5}-1}{2}$, and α_{k+1} as the non-negative root of*

$$\frac{\alpha_k^2}{1-\alpha_k} = \alpha_{k-1}^2 \frac{k}{k+1} \,.$$

Further, define

$$A_k = \prod_{i=0}^{k}(1-\alpha_i) \,.$$

Then, we have for all $k \geq 0$,

$$\frac{1}{k+1}\left(1-\frac{1}{\sqrt{2}}\right) \leq A_k \leq \frac{1}{k+2} \,. \tag{A.41}$$

Lemma A.16. *Fix some $\varepsilon > 0$. If $k \geq \frac{2}{\varepsilon}\log\frac{2}{\varepsilon}$, then we have that $\frac{\log k}{k} \leq \varepsilon$.*

Proof. We have, since $\log x \leq x$ for $x > 0$,

$$\frac{\log k}{k} \leq \frac{\log\frac{2}{\varepsilon} + \log\log\frac{2}{\varepsilon}}{\frac{2}{\varepsilon}\log\frac{2}{\varepsilon}} = \frac{\varepsilon}{2}\left(1 + \frac{\log\log\frac{2}{\varepsilon}}{\log\frac{2}{\varepsilon}}\right) \leq \varepsilon \,.$$

□

A.6 Prox-Linear with Incremental Gradient Methods: Convergence Proofs

This section contains proofs from Section 3.3.

A.6.1 Prox-Linear: Outer Loop Convergence Analysis

We first prove Lemma 3.4 that specifies the assumption required by the prox-linear in the case of structured prediction.

Lemma 3.4. *Consider the structural hinge loss $f(\boldsymbol{w}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \psi(\boldsymbol{y};\boldsymbol{w}) = h \circ \boldsymbol{g}(\boldsymbol{w})$ where $h, \boldsymbol{g}$ are as defined in* (2.11)*. If the mapping $\boldsymbol{w} \mapsto \psi(\boldsymbol{y};\boldsymbol{w})$ is L-smooth with respect to $\|\cdot\|_2$ for all $\boldsymbol{y} \in \mathcal{Y}$, then it holds for all $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$ that*

$$|h(\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z})) - h(\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})| \leq \frac{L}{2}\|\boldsymbol{z}\|_2^2 \,.$$

Proof. For any $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{w} \in \mathbb{R}^d$, and $\|\boldsymbol{A}\|_{2,1}$ defined in (2.9), notice that

$$\|\boldsymbol{A}\boldsymbol{w}\|_\infty \leq \|\boldsymbol{A}\|_{2,1}\|\boldsymbol{w}\|_2 \,. \tag{A.42}$$

Now using the fact that max function h satisfies $|h(\boldsymbol{u}') - h(\boldsymbol{u})| \leq \|\boldsymbol{u}' - \boldsymbol{u}\|_\infty$ and the fundamental theorem of calculus $(*)$, we deduce

$$\begin{aligned}
|h(\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z})) - h(\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})| &\leq \|\boldsymbol{g}(\boldsymbol{w}+\boldsymbol{z}) - (\boldsymbol{g}(\boldsymbol{w}) + \nabla\boldsymbol{g}(\boldsymbol{w})\boldsymbol{z})\,\|_\infty \\
&\overset{(*)}{\leq} \left\| \int_0^1 (\nabla\boldsymbol{g}(\boldsymbol{w}+t\boldsymbol{z}) - \nabla\boldsymbol{g}(\boldsymbol{w}))\boldsymbol{z}\, dt \right\|_\infty \\
&\overset{(A.42)}{\leq} \int_0^1 \|\nabla\boldsymbol{g}(\boldsymbol{w}+t\boldsymbol{z}) - \nabla\boldsymbol{g}(\boldsymbol{w})\|_{2,1}\|\boldsymbol{z}\|_2\, dt \,.
\end{aligned} \tag{A.43}$$

Note that the definition (2.9) can equivalently be stated as $\|\boldsymbol{A}\|_{2,1} = \max_{\|\boldsymbol{u}\|_1\leq 1} \|\boldsymbol{A}^\top\boldsymbol{u}\|_2$. Given $\boldsymbol{u} \in \mathbb{R}^m$, we index its entries $u_{\boldsymbol{y}}$ by $\boldsymbol{y} \in \mathcal{Y}$. Then, the matrix norm in (A.43) can be simplified as

$$\begin{aligned}
\|\nabla\boldsymbol{g}(\boldsymbol{w}+t\boldsymbol{z}) - \nabla\boldsymbol{g}(\boldsymbol{w})\|_{2,1} &= \max_{\|\boldsymbol{u}\|_1\leq 1} \left\| \sum_{\boldsymbol{y}\in\mathcal{Y}} u_{\boldsymbol{y}}(\nabla\psi(\boldsymbol{y};\boldsymbol{w}+t\boldsymbol{z}) - \nabla\psi(\boldsymbol{y};\boldsymbol{w})) \right\|_2 \\
&\leq \max_{\|\boldsymbol{u}\|_1\leq 1} \sum_{\boldsymbol{y}\in\mathcal{Y}} |u_{\boldsymbol{y}}| \|\nabla\psi(\boldsymbol{y};\boldsymbol{w}+t\boldsymbol{z}) - \nabla\psi(\boldsymbol{y};\boldsymbol{w})\|_2 \\
&\leq Lt\|\boldsymbol{z}\|_2 \,,
\end{aligned}$$

from the L-smoothness of ψ. Plugging this back into (A.43) completes the proof. The bound on the smothing approximation holds similarly by noticing that if h is 1-Lipschitz then $h_{\mu\omega}$ too since $\nabla h_{\mu\omega}(\boldsymbol{u}) \in \operatorname{dom} h^*$ for any $\boldsymbol{u} \in \operatorname{dom} h$. □

We now detail the stationarity measure used to analyze the convergence of the prox-linear in Proposition 3.5.

Proposition 3.5. *Consider ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. For any $\boldsymbol{w} \in \mathbb{R}^d$ and $\mu, \kappa \geq 2L$, if $\boldsymbol{z} \in \mathbb{R}^d$ approximately minimizes the model $C_{\mu\omega,\kappa}$ defined in* (3.11) *as*

$$\|\nabla C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w})\|_2 \leq \frac{\kappa}{2}\|\boldsymbol{z}-\boldsymbol{w}\|_2,$$

then there exists $\hat{\boldsymbol{z}} \in \mathbb{R}^d$ such that

$$\|\hat{\boldsymbol{z}} - \boldsymbol{z}\|_2 \leq \|\boldsymbol{z}-\boldsymbol{w}\|_2 + \sqrt{\mu D} \qquad \textit{and} \qquad d(0, \partial F(\hat{\boldsymbol{z}})) \leq 5\kappa\|\boldsymbol{z}-\boldsymbol{w}\|_2 + 3\sqrt{\mu\kappa D}.$$

where $d(0, \partial F(\hat{\boldsymbol{z}})) = \inf_{\boldsymbol{u}\in\partial F(\hat{\boldsymbol{z}})} \|\boldsymbol{u}\|_2$.

Proof. By κ-strong convexity of $C_{\mu\omega,\kappa}(\cdot;\boldsymbol{w})$, we have for any $\boldsymbol{u}\in\mathbb{R}^d$,

$$\begin{aligned}C_{\mu\omega,\kappa}(\boldsymbol{u};\boldsymbol{w}) &\geq C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w}) + \nabla C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w})^\top(\boldsymbol{u}-\boldsymbol{z}) + \frac{\kappa}{2}\|\boldsymbol{u}-\boldsymbol{z}\|_2^2\\ &\overset{(3.13)}{\geq} C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w}) - \frac{\kappa}{2}\|\boldsymbol{z}-\boldsymbol{w}\|_2\|\boldsymbol{u}-\boldsymbol{z}\|_2 + \frac{\kappa}{2}\|\boldsymbol{u}-\boldsymbol{z}\|_2^2\\ &\geq C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w}) - \frac{\kappa}{8}\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 \end{aligned} \tag{A.44}$$

where we used $2ab\leq a^2+b^2$ for any reals a,b. Denote then $\bar{F}:\boldsymbol{u}\to F(\boldsymbol{u})+\frac{\kappa+L}{2}\|\boldsymbol{u}-\boldsymbol{w}\|_2^2$, then for any $\boldsymbol{u}\in\operatorname{dom} f$,

$$\begin{aligned}\bar{F}(\boldsymbol{u}) &\overset{(3.18)}{\geq} C(\boldsymbol{u};\boldsymbol{w}) + \frac{\kappa}{2}\|\boldsymbol{u}-\boldsymbol{w}\|_2^2\\ &\overset{(3.17)}{\geq} C_{\mu\omega}(\boldsymbol{u};\boldsymbol{w}) + \frac{\kappa}{2}\|\boldsymbol{u}-\boldsymbol{w}\|_2^2\\ &\overset{(A.44)}{\geq} C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w}) - \frac{\kappa}{8}\|\boldsymbol{z}-\boldsymbol{w}\|_2^2.\end{aligned}$$

Therefore

$$\begin{aligned}\bar{F}(\boldsymbol{z}) - \inf_{\boldsymbol{u}}\bar{F}(\boldsymbol{u}) &\leq F(\boldsymbol{z}) + \frac{\kappa+L+\kappa/4}{2}\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 - C_{\mu\omega,\kappa}(\boldsymbol{z};\boldsymbol{w})\\ &\leq F(\boldsymbol{z}) + \frac{L+\kappa/4}{2}\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 - C_{\mu\omega}(\boldsymbol{z};\boldsymbol{w})\\ &\overset{(3.17)}{\leq} F(\boldsymbol{z}) - C(\boldsymbol{z};\boldsymbol{w}) + \frac{L+\kappa/4}{2}\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 + \mu D\\ &\overset{(3.18)}{\leq} \left(L+\frac{\kappa}{8}\right)\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 + \mu D\\ &\leq \frac{9}{8}\kappa\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 + \mu D\end{aligned}$$

By Ekeland's variational principle, for any $\rho>0$ there exists then $\hat{\boldsymbol{z}}$ such that, denoting $\theta=\frac{9}{8}\kappa\|\boldsymbol{z}-\boldsymbol{w}\|_2^2+\mu D$,

$$\|\hat{\boldsymbol{z}}-\boldsymbol{z}\|_2\leq\frac{\theta}{\rho},\qquad \bar{F}(\hat{\boldsymbol{z}})\leq\bar{F}(\boldsymbol{z}),\qquad d(0,\partial\bar{F}(\hat{\boldsymbol{z}}))\leq\rho. \tag{A.45}$$

where $d(0,\partial\bar{F}(\hat{\boldsymbol{z}}))=\inf_{\boldsymbol{u}\in\partial\bar{F}(\hat{\boldsymbol{z}})}\|\boldsymbol{u}\|_2$[1]. Choosing $\rho=\frac{9}{8}\kappa\|\boldsymbol{z}-\boldsymbol{w}\|_2+\sqrt{\kappa\mu D}$, we get

$$\|\hat{\boldsymbol{z}}-\boldsymbol{z}\|_2\leq\|\boldsymbol{z}-\boldsymbol{w}\|_2+\sqrt{\frac{\mu D}{\kappa}}.$$

Finally, using that $\partial\bar{F}(\hat{\boldsymbol{z}})=\partial F(\hat{\boldsymbol{z}})+(\kappa+L)(\hat{\boldsymbol{z}}-\boldsymbol{w})$ and $\|\hat{\boldsymbol{z}}-\boldsymbol{w}\|_2\leq\|\hat{\boldsymbol{z}}-\boldsymbol{z}\|_2+\|\boldsymbol{z}-\boldsymbol{w}\|_2\leq 2\|\boldsymbol{z}-\boldsymbol{w}\|_2+\sqrt{\mu D/\kappa}$, we get from (A.45)

$$d(0,\partial F(\hat{\boldsymbol{z}}))\leq\frac{33}{8}\kappa\|\boldsymbol{z}-\boldsymbol{w}\|_2+\frac{5\kappa}{2}\sqrt{\frac{\mu D}{\kappa}}.$$

The result follows after simplifications. □

[1]The original statement is that $\hat{\boldsymbol{z}}$ is the unique minimizer of $\boldsymbol{z}\to\bar{F}(\boldsymbol{z})+\rho\|\boldsymbol{z}-\hat{\boldsymbol{z}}\|_2$. We deduce $0\in\partial\bar{F}(\hat{\boldsymbol{z}})+\rho\partial\|0\|_2$ and so $d(0,\partial F(\hat{\boldsymbol{z}}))\leq\rho$.

We now restate and proof Theorem 3.7.

Theorem 3.7. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. Feed Algorithm 3.1 with a regularization $\kappa \geq 2L$ and non-increasing smoothing parameters $(\mu_k)_{k\geq 1}$, then it produces a sequence $(\boldsymbol{w}_k)_{k\geq 0}$ that satisfies*

$$\min_{k\in\{1,\dots,K\}} \kappa^2 \|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq \frac{4\kappa}{K}\big(F(\boldsymbol{w}_0) - F^* + \mu_1 D\big),$$

where $F^* = \inf_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w})$.

Proof. For $k \geq 1$, by κ-strong convexity of $C_{\mu_k\omega,\kappa}(\cdot; \boldsymbol{w}_{k-1})$, we have

$$\begin{aligned}
F_{\mu_k\omega}(\boldsymbol{w}_{k-1}) &= C_{\mu_k\omega,\kappa}(\boldsymbol{w}_{k-1}; \boldsymbol{w}_{k-1}) \\
&\geq C_{\mu_k\omega,\kappa}(\boldsymbol{w}_k; \boldsymbol{w}_{k-1}) + \nabla C_{\mu_k\omega,\kappa}(\boldsymbol{w}_k; \boldsymbol{w}_{k-1})^\top (\boldsymbol{w}_k - \boldsymbol{w}_{k-1}) + \frac{\kappa}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \\
&\overset{(3.13)}{\geq} C_{\mu_k\omega,\kappa}(\boldsymbol{w}_k; \boldsymbol{w}_{k-1}) \\
&\overset{(3.18)}{\geq} F_{\mu_k\omega}(\boldsymbol{w}_k) + \frac{\kappa - L}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2
\end{aligned}$$

We have then for $k \geq 2$, using (3.16),

$$\frac{\kappa - L}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq F_{\mu_{k-1}\omega}(\boldsymbol{w}_{k-1}) - F_{\mu_k\omega}(\boldsymbol{w}_k) + (\mu_{k-1} - \mu_k)D$$

and for $k = 1$,

$$\frac{\kappa - L}{2}\|\boldsymbol{w}_1 - \boldsymbol{w}_0\|_2^2 \leq F(\boldsymbol{w}_0) - F_{\mu_1\omega}(\boldsymbol{w}_1)$$

Summing along $k \geq 1$, we have

$$\begin{aligned}
\frac{\kappa - L}{2}\sum_{k=1}^{K} \|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 &\leq F(\boldsymbol{w}_0) - F_{\mu_K\omega}(\boldsymbol{w}_K) + (\mu_1 - \mu_K)D \\
&\leq F(\boldsymbol{w}_0) - F(\boldsymbol{w}_K) + \mu_1 D. \qquad \text{(A.46)}
\end{aligned}$$

Therefore for $\kappa \geq 2L$, we get

$$\min_{k\in\{1,\dots,K\}} \|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq \frac{4}{\kappa K}(F(\boldsymbol{w}_0) - F^* + \mu_1 D).$$

□

Next, we restate and prove Proposition 3.9.

Proposition 3.9. *Consider F of the form* (3.2)*, $C_{\mu\omega,\kappa}$ defined in* (3.11) *with ω satisfying* (3.15) *and suppose Assumption 3.3 such that* (3.16)*,* (3.17)*,* (3.18)*,* (3.19) *hold. Taking $\bar{\kappa} \geq 2L$ and non-increasing smoothing parameters μ_k, the iterates produced by Algorithm 3.2 satisfy*

$$\min_{k\in\{1,\dots,K\}} \bar{\kappa}^2 \|\boldsymbol{w}_k - \boldsymbol{w}_{k-1}\|_2^2 \leq \frac{4\bar{\kappa}}{K}\big(F(\boldsymbol{w}_0) - F^* + \mu_1 D\big).$$

If moreover the smooth convex models lower bound the smooth objective, i.e. for $k \geq 1$ *and* $\boldsymbol{w}^* \in \arg\min_w F(w)$

$$C_{\mu_k\omega}(\boldsymbol{w}_{k-1}; \boldsymbol{z}_{k-1}) \leq F_{\mu_k\omega}(\boldsymbol{w}_{k-1}) \quad \textit{and} \quad C_{\mu_k\omega}(\boldsymbol{w}^*; \boldsymbol{z}_{k-1}) \leq F_{\mu_k\omega}(\boldsymbol{w}^*), \tag{3.23}$$

then the algorithm converges for non-decreasing regularization parameters $\kappa_k \geq 2L$ *as*

$$F(\boldsymbol{w}_k) - F^* \leq \frac{\mathcal{A}_0^{k-1}}{\mathcal{B}_1^k}\Delta_0 + \mu_k D_\omega + \sum_{j=1}^{k} \frac{\mathcal{A}_j^{k-1}}{\mathcal{B}_j^k}\left(\mu_{j-1} - (1-\delta_j)\mu_j\right) D_\omega\,,$$

where $\mathcal{A}_i^j := \prod_{r=i}^{j}(1-\alpha_r)$, $\mathcal{B}_i^j := \prod_{r=i}^{j}(1-\delta_r)$, $\Delta_0 := F(\boldsymbol{w}_0) - F^* + \frac{\kappa_1\alpha_0^2}{2(1-\alpha_0)}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2$ *and* $\mu_0 := 2\mu_1$.

Proof. First part of the proposition stems from Theorem 3.7 and picking best of two steps in line 5. Assume now $F_{\mu_k\omega}$ to be lower bounded by the models $C_{\mu_k\omega}(\cdot; \boldsymbol{z}_{k-1})$ as in (3.23). We follow the proof of Theorem 2.17 from Chapter 2 in the convex case such that $\eta_k = \alpha_k$ and $\lambda = 0$. Denoting then for any $k \geq 0$,

$$\begin{aligned}
S_0 &= (1-\alpha_0)(F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*) + \frac{\kappa_1\alpha_0^2}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 \\
S_k &= (1-\alpha_k)(F_{\mu_k\omega}(\boldsymbol{w}_k) - F_{\mu_k\omega}(\boldsymbol{w}^*)) + \frac{\kappa_{k+1}\alpha_k^2}{2}\|\boldsymbol{z}_k - \boldsymbol{w}^*\|_2^2,
\end{aligned}$$

we want to show the following recurrence

$$\frac{S_k}{1-\alpha_k} - \frac{\kappa_{k+1}\alpha_k^2\delta_k}{2(1-\alpha_k)} \leq S_{k-1} + (\mu_{k-1} - \mu_k)D. \tag{A.47}$$

We start by detailing the decrease made by each approximate minimization of the convex models. At the k^{th} iterate of the algorithm, with $k \geq 1$, by κ_k-strong convexity of the regularized model, for any $\boldsymbol{u} \in \mathbb{R}^d$,

$$\begin{aligned}
C_{\mu_k\omega,\kappa_k}(\boldsymbol{u}; \boldsymbol{z}_{k-1}) &\geq C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{\kappa_k}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2 + \nabla C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1})^\top(\boldsymbol{u} - \boldsymbol{\zeta}_k) \\
&\overset{(3.21)}{\geq} C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{\kappa_k}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2 - \frac{\kappa_k\sqrt{\delta_k}}{\sqrt{2}}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|\|\boldsymbol{u} - \boldsymbol{\zeta}_k\| \\
&\geq C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{\kappa_k}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2 - \frac{\kappa_k}{4}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|^2 - \frac{\delta_k\kappa_k}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|^2
\end{aligned}$$

where we used $2ab \leq a^2 + b^2$ for any reals a, b. Therefore

$$C_{\mu_k\omega,\kappa_k}(\boldsymbol{u}; \boldsymbol{z}_{k-1}) + \frac{\kappa_k}{4}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|^2 \geq C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{\kappa_k(1-\delta_k)}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2 \tag{A.48}$$

Now we make in evidence the recurrence as, denoting $F_k = F_{\mu_k\omega}$ with $F_0 = F$,

$$\begin{aligned}
F_k(\boldsymbol{w}_k) \leq F_k(\boldsymbol{\zeta}_k) &\overset{(3.18)}{\leq} C_{\mu_k\omega}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{L}{2}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|_2^2 \\
&\leq C_{\mu_k\omega,\kappa_k}(\boldsymbol{\zeta}_k; \boldsymbol{z}_{k-1}) + \frac{L-\kappa_k}{2}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|_2^2 \\
&\overset{(A.48)}{\leq} C_{\mu_k\omega,\kappa_k}(\boldsymbol{u}; \boldsymbol{z}_{k-1}) + \frac{\kappa_k/2 + L - \kappa_k}{2}\|\boldsymbol{\zeta}_k - \boldsymbol{z}_{k-1}\|_2^2 + \frac{\kappa_k(\delta_k - 1)}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2 \\
&\leq C_{\mu_k\omega}(\boldsymbol{u}; \boldsymbol{z}_{k-1}) + \frac{\kappa_k}{2}\|\boldsymbol{u} - \boldsymbol{z}_{k-1}\|_2^2 + \frac{\kappa_k(\delta_k - 1)}{2}\|\boldsymbol{u} - \boldsymbol{\zeta}_k\|_2^2,
\end{aligned}$$

where we used in last inequality the choice of $\kappa_k \geq 2L$.

Taking $\boldsymbol{u} = \alpha_{k-1}\boldsymbol{w}^* + (1-\alpha_{k-1})\boldsymbol{w}_{k-1}$, where $\boldsymbol{w}^* \in \arg\min F$, using convexity of the models $C_{\mu_k\omega}(\cdot; \boldsymbol{z}_{k-1})$,

$$\begin{aligned} F_k(\boldsymbol{w}_k) \leq & \alpha_{k-1} C_{\mu_k\omega}(\boldsymbol{w}^*; \boldsymbol{z}_{k-1}) + (1-\alpha_{k-1}) C_{\mu_k\omega}(\boldsymbol{w}_{k-1}; \boldsymbol{z}_{k-1}) \\ & + \frac{\kappa_k \alpha_{k-1}^2}{2}\|\boldsymbol{z}_{k-1} - \boldsymbol{w}^*\|_2^2 + \frac{\kappa_k(\delta_k - 1)\alpha_{k-1}^2}{2}\|\boldsymbol{z}_k - \boldsymbol{w}^*\|_2^2 \\ \overset{(3.23)}{\leq} & \alpha_{k-1} F_k(\boldsymbol{w}^*) + (1-\alpha_{k-1}) F_k(\boldsymbol{w}_{k-1}) \\ & + \frac{\kappa_k \alpha_{k-1}^2}{2}\|\boldsymbol{z}_{k-1} - \boldsymbol{w}^*\|_2^2 + \frac{\kappa_k(\delta_k - 1)\alpha_{k-1}^2}{2}\|\boldsymbol{z}_k - \boldsymbol{w}^*\|_2^2. \end{aligned}$$

Therefore, rearranging the terms and using the approximation made by smoothing for decreasing μ_k, denoting $\mu_0 = 2\mu_1$, we get

$$\begin{aligned} F_k(\boldsymbol{w}_k) - F_k(\boldsymbol{w}^*) + \frac{\kappa_k(1-\delta_k)\alpha_{k-1}^2}{2}\|\boldsymbol{z}_k - \boldsymbol{w}^*\|_2^2 \leq & (1-\alpha_{k-1})\left(F_{k-1}(\boldsymbol{w}_{k-1}) - F_{k-1}(\boldsymbol{w}^*)\right) \\ & + \frac{\kappa_k \alpha_{k-1}^2}{2}\|\boldsymbol{z}_{k-1} - \boldsymbol{w}^*\|_2^2 \\ & + (1-\alpha_k)(\mu_{k-1} - \mu_k)D \end{aligned} \tag{A.49}$$

Using that $\kappa_k \alpha_{k-1}^2 = \kappa_{k+1}\alpha_k^2/(1-\alpha_k)$ and $1 - \alpha_k \leq 1$, this reads

$$\frac{S_k}{1-\alpha_k} - \frac{\kappa_{k+1}\alpha_k^2 \delta_k}{2(1-\alpha_k)} \leq S_{k-1} + (\mu_{k-1} - \mu_k)D.$$

We then retrieved the inequality (A.47) used to prove Theorem 2.17. The claim follows then the same steps with $\eta_k = \alpha_k$ and $\lambda = 0$. □

A.6.2 Information-Based Complexity of the Prox-linear Algorithm with Casimir-SVRG

We now state and prove Proposition 3.10.

Proposition 3.10. *Consider the setting of Corollary 3.8 and that the subproblem of Line 2 of Algorithm 3.1 is solved using Catalyst-SVRG. Then, total number of SVRG iterations N required to produce an iterate $\varepsilon(1/\kappa + 1/\sqrt{\kappa})$close to an ε-near stationary is bounded as*

$$\mathbb{E}[N] \leq \widetilde{O}\left(\kappa \frac{\sqrt{A_\omega n}}{\varepsilon^3}(\Delta F_0 + \mu_0 D)^{3/2}\right).$$

Proof. We only need to bound the number of iterations of each subproblem and combine it with Corollary 3.8. Denote for $k \geq 1$, $C_k(\cdot) = C_{\mu_k\omega,\kappa}(\cdot; \boldsymbol{w}_{k-1})$, $\boldsymbol{w}_k^* = \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} C_k(\boldsymbol{w})$, $C_k^* = \min_{\boldsymbol{w}\in\mathbb{R}^d} C_k(\boldsymbol{w})$ and $\boldsymbol{z}_t$ the iterates produced by a linearly convergent method such as Catalyst-SVRG with $\boldsymbol{z}_0 = \boldsymbol{w}_{k-1}$.

We will show that to get the stopping criterion $\|\nabla C_k(\boldsymbol{z}_t)\|_2 \leq \kappa\|\boldsymbol{w}_{k-1} - \boldsymbol{z}_t\|_2/2$, it suffices to decrease the objective gap $C_k(\boldsymbol{z}_t) - C_k^*$ by a constant factor. Precisely, assume that

$$C_k(\boldsymbol{z}_t) - C_k^* \leq \frac{\kappa^2}{18(A_\omega/\mu_k + \kappa)^2}(C_k(\boldsymbol{z}_0) - C_k^*) \tag{A.50}$$

Then by $(A_\omega/\mu_k+\kappa)$-smoothness of C_k, we have [2]

$$\|\nabla C_k(\boldsymbol{z}_t)\|_2^2 \leq 2(A_\omega/\mu_k+\kappa)(C_k(\boldsymbol{z}_t)-C_k^*) \leq \frac{\kappa^2}{9(A_\omega/\mu_k+\kappa)}(C_k(\boldsymbol{z}_0)-C_k^*) \leq \frac{\kappa^2\|\boldsymbol{z}_0-\boldsymbol{w}_k^*\|_2^2}{9}$$

Therefore, as $\boldsymbol{z}_0=\boldsymbol{w}_{k-1}$, $\|\nabla C_k(\boldsymbol{z}_t)\|_2 \leq \kappa\|\boldsymbol{w}_{k-1}-\boldsymbol{w}_k^*\|_2/3$. Then using that $\|\nabla C_k(\boldsymbol{z}_t)\|_2 \geq \kappa\|\boldsymbol{z}_t-\boldsymbol{w}_k^*\|_2$ by κ-strong convexity of C_k,

$$3\|\nabla C_k(\boldsymbol{z}_t)\| \leq \kappa\|\boldsymbol{w}_{k-1}-\boldsymbol{z}_t\|_2+\kappa\|\boldsymbol{z}_t-\boldsymbol{w}_k^*\|_2 \leq \kappa\|\boldsymbol{w}_{k-1}-\boldsymbol{z}_t\|_2+\|\nabla C_k(\boldsymbol{z}_t)\|$$

and so $\|\nabla C_k(\boldsymbol{z}_t)\|_2 \leq \kappa\|\boldsymbol{w}_{k-1}-\boldsymbol{z}_t\|_2/2$.

Recall now that for strongly convex objectives the complexity of Catalyst-SVRG to achieve an ε accuracy reads $N = O(\sqrt{nL_{C_k}/\mu_{C_k}}\log(\gamma_{C_k})\log((C_k(\boldsymbol{z}_0)-C_k^*)/\varepsilon)$ where γ_{C_k} is a constant depending on global properties of C_k (Lin et al., 2018, Proposition 17). The total complexity of Catalyst-SVRG at the k^{th} iteration to achieve (A.50) is then, ignoring the logarithmic dependencies in μ_k, A_ω, κ, of the order of $\mathbb{E}[N_k] = \tilde{O}(\sqrt{n(A_\omega/\mu_k+\kappa)/\kappa})$.

Now, using Corollary 3.8 the number of outer iterations of the prox-linear needed to achieve the ε target accuracy is, denoting $\Delta F_0 = F(\boldsymbol{w}_0)-F^*$,

$$K = \frac{4\kappa}{\varepsilon^2}\big(\Delta F_0+\mu_0 D\big)$$

Therefore collecting all constants and terms logarithmic in the parameters in $\mathcal{T}, \mathcal{T}', \mathcal{T}''$, the total number of iterations of SVRG is in

$$\begin{aligned}
\mathbb{E}[N] = \sum_{k=1}^K \mathbb{E}[N_k] &\leq \sum_{k=1}^K \left(\sqrt{n\frac{A_\omega/\mu_k+\kappa}{\kappa}}\right)\mathcal{T} \\
&\leq \left[\sqrt{\frac{A_\omega n}{\kappa}}\left(\sum_{k=1}^K \sqrt{k}\right)\right]\mathcal{T} \\
&\leq \left[\sqrt{\frac{A_\omega n}{\kappa}}\,K^{3/2}\right]\mathcal{T}' \\
&\leq \left[\sqrt{\frac{A_\omega n}{\kappa}}\left(\frac{\kappa(\Delta F_0+\mu_0 D)}{\varepsilon^2}\right)^{3/2}\right]\mathcal{T}'' \\
&= \left[\kappa\frac{\sqrt{A_\omega n}}{\varepsilon^3}(\Delta F_0+\mu_0 D)^{3/2}\right]\mathcal{T}''\,.
\end{aligned}$$

□

A.7 Local Quadratic Convergence of the Prox-Linear Method

Here, we give with a simple proof of Proposition 3.1, following standard techniques (Nesterov, 2007, Theorem 3).

[2] We use in the first inequality that a gradient step $\boldsymbol{z}_t^+ = \boldsymbol{z}_t - \frac{1}{A_\omega/\mu_k+\kappa}\nabla C_k(\boldsymbol{z}_t)$ at $\boldsymbol{z}_t$ reads by smoothness $C_k(\boldsymbol{w}_k^*) \leq C_k(\boldsymbol{z}_t^+) \leq C(\boldsymbol{z}_t) - \frac{1}{2(A_\omega/\mu_k+\kappa)}\|\nabla C_k(\boldsymbol{z}_t)\|_2^2$.

Proposition 3.1. *Consider problem* (3.2) *where h is ℓ-Lipschitz, convex, and α-sharp for some $\alpha > 0$, i.e., $h(\boldsymbol{u}) - \min h \geq \alpha \operatorname{dist}(\boldsymbol{u}, U^*)$ for any $\boldsymbol{u} \in \mathbb{R}^m$ where $\operatorname{dist}(\boldsymbol{u}, U^*)$ is the Euclidean distance of $\boldsymbol{u}$ to $U^* = \arg\min_{\boldsymbol{u}\in\mathbb{R}^m} h(\boldsymbol{u}) \neq \emptyset$. Further, suppose the function $\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{g}_1(\boldsymbol{w}); \ldots; \boldsymbol{g}_n(\boldsymbol{w})) \in \mathbb{R}^{nk}$ is L-smooth and satisfies $\sigma_{\min}(\nabla \boldsymbol{g}(\boldsymbol{w})^\top) \geq \nu > 0$ for any $\boldsymbol{w} \in \mathbb{R}^d$. Then, the sequence $(\boldsymbol{w}_k)_{k=0}^\infty$ produced by the prox-linear algorithm* (3.4) *with $\kappa = L\ell$ starting from arbitrary $\boldsymbol{w}_0 \in \mathbb{R}^d$ converges globally as $F(\boldsymbol{w}_k) \to F^* := \min F$. Furthermore, if an iterate $\boldsymbol{w}_j$ satisfies $F(\boldsymbol{w}_k) - F^* \leq (\alpha\nu n^2)^2/(L\ell)$, the subsequence $(\boldsymbol{w}_k)_{k=j}^\infty$ convergences quadratically as*

$$F(\boldsymbol{w}_{k+1}) - F^* \leq \frac{L\ell n^2}{2(\alpha\nu)^2}\big(F(\boldsymbol{w}_k) - F^*\big)^2 . \tag{3.5}$$

We prove the proposition for the case $n = 1$, i.e., $\min_{w\in\mathbb{R}^d} h \circ \boldsymbol{g}(w)$. We can generalize to $n > 1$ with the reduction $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^{mn}$ and $f : \mathbb{R}^{mn} \to \mathbb{R}$ given by

$$\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{g}_1(\boldsymbol{w}); \ldots; \boldsymbol{g}_n(\boldsymbol{w})), \quad \text{and,} \quad h\big(\boldsymbol{u}_1; \ldots; \boldsymbol{u}_n\big) = \frac{1}{n}\sum_{i=1}^n h_i(\boldsymbol{u}_i) , \tag{A.51}$$

where we use semi-colons to denote the concatenation of vectors.

We are interested primarily in the overparameterized case where $m \leq d$. Below, we denote $\nabla \boldsymbol{g}(\boldsymbol{w}) \in \mathbb{R}^{m\times d}$ as the Jacobian of $\boldsymbol{g}$ at $\boldsymbol{w}$. We impose the assumption that its minimal singular value is bounded away from 0 as $\sigma_{\min}(\nabla \boldsymbol{g}(\boldsymbol{w})^\top) \geq \nu > 0$ for any $\boldsymbol{w} \in \mathbb{R}^d$, This assumption implies the surjectivity of the Jacobian at each $\boldsymbol{w}$. That is, for every $\boldsymbol{u} \in \mathbb{R}^m$, there exists a $v \in \mathbb{R}^d$ such that $\nabla \boldsymbol{g}(\boldsymbol{w})v = \boldsymbol{u}$.

We also assume that the following minimum values are bounded from below:

$$h^* = \min_{\boldsymbol{u}\in\mathbb{R}^m} h(\boldsymbol{u}), \quad \text{and,} \quad (h \circ \boldsymbol{g})^* = \min_{\boldsymbol{w}\in\mathbb{R}^d} h\big(\boldsymbol{g}(\boldsymbol{w})\big).$$

Proposition A.17. *Consider the compositional problem $\min_{\boldsymbol{w}\in\mathbb{R}^d} h \circ \boldsymbol{g}(\boldsymbol{w})$ with following assumptions:*

(a) h is ℓ-Lipschitz continuous, convex and α-sharp, i.e., $h(\boldsymbol{u}) - h^* \geq \alpha \operatorname{dist}(\boldsymbol{u}, U^*)$ for any $\boldsymbol{u} \in \mathbb{R}^m$ with $\alpha > 0$ and $\operatorname{dist}(\boldsymbol{u}, U^*)$ the Euclidean distance of $\boldsymbol{u}$ to $U^* = \arg\min_{\boldsymbol{u}\in\mathbb{R}^m} h(\boldsymbol{u}) \neq \emptyset$.

(b) $\boldsymbol{g}$ is L-smooth and satisfies $\sigma_{\min}(\nabla \boldsymbol{g}(\boldsymbol{w})^\top) \geq \nu > 0$ for any $\boldsymbol{w} \in \mathbb{R}^d$.

The sequence $(\boldsymbol{w}_k)_{k=0}^\infty$ produced by the prox-linear algorithm (3.4) *with $M = L\ell$ starting from arbitrary $\boldsymbol{w}_0 \in \mathbb{R}^d$ converges globally as $(h \circ \boldsymbol{g})(\boldsymbol{w}_k) \to (h \circ \boldsymbol{g})^* = h^*$. Furthermore, as soon as an iterate $\boldsymbol{w}_j$ satisfies $h(\boldsymbol{g}(\boldsymbol{w}_k)) - (h \circ \boldsymbol{g})^* \leq (\alpha\nu)^2/(L\ell)$, the subsequence $(w_k)_{k=j}^\infty$ convergences quadratically as*

$$h(\boldsymbol{g}(\boldsymbol{w}_{k+1})) - (h \circ \boldsymbol{g})^* \leq \frac{L\ell}{2(\alpha\nu)^2}(h(\boldsymbol{g}(\boldsymbol{w}_k)) - (h \circ \boldsymbol{g})^*)^2 \leq \frac{1}{2}(h(\boldsymbol{g}(\boldsymbol{w}_k)) - (h \circ \boldsymbol{g})^*).$$

Proof. For an iterate $\boldsymbol{w}_k$ of the prox-linear algorithm, denote $\boldsymbol{u}_k^* = \operatorname{Proj}_{U^*}(\boldsymbol{g}(\boldsymbol{w}_k))$ the Euclidean projection of $\boldsymbol{g}(\boldsymbol{w}_k)$ onto the set of minimizers of h such that $\operatorname{dist}(\boldsymbol{g}(\boldsymbol{w}_k), U^*) = \|\boldsymbol{g}(\boldsymbol{w}_k) - \boldsymbol{u}_k^*\|_2$.

Since the Jacobian $\nabla \boldsymbol{g}(\boldsymbol{w}_k)^\top$ is surjective, there exists v_k^* be such that $\nabla \boldsymbol{g}(\boldsymbol{w}_k)v_k^* = \boldsymbol{u}_k^* - \boldsymbol{g}(\boldsymbol{w}_k)$. Furthermore, from the minimum singular value condition, there exists a choice of v_k^* such that $\|v_k^*\| \leq \|\boldsymbol{u}_k^* - \boldsymbol{g}(\boldsymbol{w})\|_2/\nu$ (see (Nesterov, 2007, Lemma 6) for a proof).

By definition of the prox-linear update in (3.4), we get

$$\begin{aligned} h(\boldsymbol{g}(\boldsymbol{w}_{k+1})) &= \min_{v\in\mathbb{R}^d}\left\{h(\boldsymbol{g}(\boldsymbol{w}_k)+\nabla\boldsymbol{g}(\boldsymbol{w}_k)v)+\frac{M}{2}\|v\|_2^2\right\} \\ &\overset{\text{(i)}}{\leq} \min_{t\in[0,1]}\left\{h(\boldsymbol{g}(\boldsymbol{w}_k)+t\nabla\boldsymbol{g}(\boldsymbol{w}_k)v_k^*)+\frac{Mt^2}{2}\|v_k^*\|_2^2\right\} \\ &\overset{\text{(ii)}}{\leq} \min_{t\in[0,1]}\left\{h(\boldsymbol{g}(\boldsymbol{w}_k)+t(\boldsymbol{u}_k^*-\boldsymbol{g}(\boldsymbol{w}_k))+\frac{Mt^2}{2\nu^2}\|\boldsymbol{u}_k^*-\boldsymbol{g}(\boldsymbol{w}_k)\|_2^2\right\} \\ &\overset{\text{(iii)}}{\leq} \min_{t\in[0,1]}\left\{th^*+(1-t)h(\boldsymbol{g}(\boldsymbol{w}_k))+\frac{Mt^2}{2(\nu\alpha)^2}(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*)^2\right\}. \end{aligned} \tag{A.52}$$

Here, we (i) restricted the domain of the minimization to $v = tv_k^*$ with $t \in [0,1]$, (ii) plugged in the definition of v_k^* and the bound on $\|v_k^*\|$, and, (iii) used the convexity and sharpness of h. Next, by subtracting h^* from both sides, we get

$$h(\boldsymbol{g}(\boldsymbol{w}_{k+1}))-h^* \leq \min_{t\in[0,1]}\left\{(1-t)(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*)+\frac{t^2M}{2(\nu\alpha)^2}(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*)^2\right\}.$$

If $h(\boldsymbol{g}(\boldsymbol{w}_k))-h^* \leq (\alpha\nu)^2/M$, the minimum in (A.52) is reached at $t=1$ and we get

$$h(\boldsymbol{g}(\boldsymbol{w}_{k+1}))-h^* \leq \frac{M}{2(\nu\alpha)^2}(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*)^2 \leq \frac{1}{2}(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*).$$

This is the quadratic convergence phase. On the other hand, if $h(\boldsymbol{g}(\boldsymbol{w}_k))-h^* \geq (\alpha\nu)^2/M$, then the minimum in (A.52) is reached at $t = (\nu\alpha)^2/\left(M\left(h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*\right)\right)$, and we have the bound

$$h(\boldsymbol{g}(\boldsymbol{w}_{k+1}))-h^* \leq h(\boldsymbol{g}(\boldsymbol{w}_k))-h^*-\frac{(\alpha\nu)^2}{2M}.$$

Since f is bounded from below, the sequence $h(\boldsymbol{g}(\boldsymbol{w}_k))$ converges to h^*. Hence, the minimum of the composite objective matches the minimum of the outer function, i.e., $h^* = (h\circ\boldsymbol{g})^*$. □

Appendix B

MAUVE: Proofs and Experimental Details

Here, we provide the full details of the proofs and experiments, as well as additional experiments in Chapter 4. We start with providing the missing proofs in Section B.1, followed by experimental details in Section B.2, and experimental results in Section B.3, followed by details of the human evaluation in Section B.4.

B.1 Proof of the Statistical Bound

We restate and give the full proof of Theorem 4.2.

Theorem 4.2. *Consider two discrete distributions $P, Q \in \Delta^{k-1}$, and i.i.d. samples $Y_1, \cdots, Y_n \sim P$ and $Y_1, \cdots, Y'_n \sim Q$. Denote $\hat{P}_n = (1/n)\sum_{i=1}^n \delta_{Y_i}$ and $\hat{Q}_n = (1/n)\sum_{i=1}^n \delta_{Y'_i}$ denote the corresponding empirical distributions. We have,*

$$\mathbb{E}\left|\mathrm{JS}(\hat{P}_n, \hat{Q}_n) - \mathrm{JS}(P, Q)\right| \leq \frac{\log n}{2}\big(\alpha_n(P) + \alpha_n(Q)\big) + \frac{1}{2}\big(\beta_n(P) + \beta_n(Q)\big)\,,$$

where $\alpha_n(P) = \sum_{i=1}^k \sqrt{n^{-1}P_i}$ and $\beta_n(P) = \mathbb{E}\big[\sum_{i:\hat{P}_{n,i}=0} P_i \log 1/P_i\big]$. We also have the distribution-free bound

$$\mathbb{E}\left|\mathrm{JS}(\hat{P}_n, \hat{Q}_n) - \mathrm{JS}(P, Q)\right| \leq \frac{\log n}{2}\left(\sqrt{\frac{k}{n}} + \frac{k}{n}\right)\,.$$

Proof. The proof relies on showing that $\left|\mathrm{JS}(\hat{P}_n, \hat{Q}_n) - \mathrm{JS}(P, Q)\right|$ is approximately Lipschitz w.r.t. $\|\cdot\|_{\mathrm{TV}}$, the total variation distance.

Notation. Define $\psi : [0,1] \times [0,1]$ as

$$\psi(p, q) = \frac{p}{2}\log\left(\frac{2p}{p+q}\right) + \frac{q}{2}\log\left(\frac{2q}{p+q}\right)\,,$$

so that $\mathrm{JS}(P, Q) = \sum_{i=1}^k \psi(P_i, Q_i)$. By the triangle inequality, we have,

$$\left|\mathrm{JS}(\hat{P}_n, \hat{Q}_n) - \mathrm{JS}(P, Q)\right| \leq \sum_{i=1}^n \underbrace{\left|\psi(\hat{P}_{n,i}, \hat{Q}_{n,i}) - \psi(P_i, \hat{Q}_{n,i})\right|}_{=:\Delta_i} + \underbrace{\left|\psi(\hat{P}_i, \hat{Q}_{n,i}) - \psi(P_i, Q_i)\right|}_{=:\Delta'_i}\,.$$

Outline. We bound Δ_i in terms of $|\hat{P}_{n,i} - P_i|$ – by summing over all coordinates, this gives a bound on the total variation distance $\|\hat{P}_n - P\|_{\mathrm{TV}} = \sum_{i=1}^k |\hat{P}_{n,i} - P_i|$. If $\hat{Q}_{n,i} = 0$, then we have $\Delta_i \leq (\log 2/2)|P_i - \hat{P}_{n,i}|$ by direct calculation. Next, we consider the case when $\hat{Q}_{n,i}$ is non-zero. A first order Taylor expansion gives the bound

$$\Delta_i \leq \max_{\lambda \in [0,1]} |\psi_p(\lambda P_i + (1-\lambda)\hat{P}_{n,i}, Q_i)| \; |P_i - \hat{P}_{n,i}| \,,$$

where ψ_p denotes the partial derivative of ψ w.r.t. its first argument. Unfortunately, if $p \leq q$ (and especially as $p \to 0$ for q fixed), we have that $|\psi_p(p,q)| \leq (1/2)\log(q/p) \to \infty$. We use two tricks to overcome this issue. We take a second order Taylor expansion so that we only depend on $\psi_p(\max\{P_i, \hat{P}_{n,i}\}, \hat{Q}_{n,i})$ and carefully bound the second order remainder term. Secondly, Because $\hat{P}_n$ is an empirical measure, we can only have two possibilities: $\hat{P}_{n,i} \geq 1/n$ or $\hat{P}_{n,i} = 0$. The first case gives an additional $\log n$ dependence, while we handle the second case with techniques from the missing mass literature.

Second Order Taylor Expansion. Assume for now that $\hat{P}_{n,i} \geq P_i$. By convexity of ψ and a second order Taylor expansion, we get,

$$\begin{aligned} 0 \leq \psi(P_i, \hat{Q}_{n,i}) - \psi(\hat{P}_{n,i}, \hat{Q}_{n,i}) - (P_i - \hat{P}_{n,i})\,\psi_p(\hat{P}_{n,i}, \hat{Q}_{n,i}) &= \frac{1}{2}\int_{\hat{P}_{n,i}}^{P_i} \psi_{pp}(s, \hat{Q}_{n,i})(p-s)\mathrm{d}s \\ &= -\frac{P_i}{2}\int_{P_i}^{\hat{P}_{n,i}} \psi_{pp}(s, \hat{Q}_{n,i})\,\mathrm{d}s + \frac{1}{2}\int_{P_i}^{\hat{P}_{n,i}} s\,\psi_{pp}(s, \hat{Q}_{n,i})\,\mathrm{d}s \\ &\leq 0 + \frac{1}{4}(\hat{P}_{n,i} - P_i)\,, \end{aligned} \tag{B.1}$$

where the last step used (a) the fact that the second partial derivative ψ_{pp} is non-negative by convexity of $\psi(\cdot, q)$ and (b) the following bound by direct calculation

$$p\,\psi_{pp}(p,q) = \frac{q}{2(p+q)} \leq \frac{1}{2} \quad \text{for all} \quad p > 0,\; q \geq 0\,.$$

We now bound the first order term in two cases:

(a) If $\hat{P}_{n,i} \geq \hat{Q}_{n,i}$, then $\psi_p(\hat{P}_{n,i}, \hat{P}_{n,i}) \geq 0$ because $\psi(\cdot, q)$ is convex with its global minimum at q. Therefore, we get by direct calculation,

$$0 \leq (\hat{P}_{n,i} - P_i)\,\psi_p(\hat{P}_{n,i}, \hat{Q}_{n,i}) = \frac{1}{2}(\hat{P}_{n,i} - P_i)\,\log\left(\frac{2\hat{P}_{n,i}}{\hat{P}_{n,i} + \hat{Q}_{n,i}}\right) \leq \frac{\log 2}{2}(\hat{P}_{n,i} - P_i)\,.$$

(b) If $\hat{P}_{n,i} \leq \hat{Q}_{n,i}$, then $\psi_p(\hat{P}_{n,i}, \hat{Q}_{n,i}) \leq 0$, we get

$$\psi_p(\hat{P}_{n,i}, \hat{Q}_{n,i}) = \frac{1}{2}\,\log\left(\frac{2\hat{P}_{n,i}}{\hat{P}_{n,i} + \hat{Q}_{n,i}}\right) \geq -\frac{1}{2}\log\frac{1}{2\hat{P}_{n,i}}\,.$$

Plugging this into (B.1) gives us, and repeating the argument with P_i and $\hat{P}_{n,i}$ swapped for $P_i \geq \hat{P}_{n,i}$ gives

$$\Delta_i \leq |\hat{P}_{n,i} - P_i|\,\max\left\{\frac{1}{4} + \frac{1}{2}\log\frac{n}{2}, \frac{1}{4}, \frac{\log 2}{2}\right\} \leq \frac{1}{2}\log\frac{1}{\max\{P_i, \hat{P}_{n,i}\}}\,|\hat{P}_{n,i} - P_i|\,. \tag{B.2}$$

Two cases based on whether i appears in the sample. If i appears in the sample, then $\hat{P}_i \geq 1/n$, so that $\log 1/\max\{\hat{P}_{n,i}, P_i\} \leq \log n$. That gives,

$$\sum_{i=1}^{k} \Delta_i \leq \frac{\log n}{2} \sum_{i\,:\,\hat{P}_{n,i}>0} |\hat{P}_{n,i} - P_i| + \frac{1}{2} \sum_{i\,:\,\hat{P}_{n,i}=0} P_i \log \frac{1}{P_i}\,.$$

Since $\psi(p,q) = \psi(q,p)$, we get an analogous bound on Δ_i'. This gives,

$$\mathbb{E}\left|\mathrm{JS}(\hat{P}_n, \hat{Q}_n) - \mathrm{JS}(P,Q)\right| \leq \frac{\log n}{2}\mathbb{E}\big[\|\hat{P}_n - P\|_{\mathrm{TV}} + \|\hat{Q}_n - Q\|_{\mathrm{TV}}\big] + \frac{1}{2}\big(\beta_n(P) + \beta_n(Q)\big)\,.$$

Final Bounds. We now get the first part of the theorem by bounding $\|\hat{P}_n - P\|_{\mathrm{TV}}$ with Jensen's inequality:

$$\mathbb{E}\sum_{i=1}^{k} |\hat{P}_{n,i} - P_i| \leq \sum_{i=1}^{k} \sqrt{\mathbb{E}(\hat{P}_{n,i} - P_i)^2} = \sum_{i=1}^{k} \sqrt{\frac{P_i(1-P_i)}{n}} \leq \alpha_n(P)\,.$$

The distribution-free bound on α_n again follows from Jensen's inequality applied to $\sqrt{\cdot}$ as

$$\frac{1}{k}\sum_{i=1}^{k} \sqrt{a_k} \leq \sqrt{\frac{1}{k}\sum_{i=1}^{k} a_k}\,.$$

The distribution-free bound on β_n comes from $\mathbb{P}(\hat{P}_{n,i} = 0) = (1-P_i)^n$ as

$$\beta_n(P) = \sum_{i=1}^{k} (1-P_i)^n P_i \log \frac{1}{P_i} \leq \frac{k \log n}{n}\,,$$

from Lemma B.1. □

We need the following technical lemma (Liu et al., 2021a, Lemma 31).

Lemma B.1. *For all $x \in (0,1)$ and $n \geq 3$, we have*

$$0 \leq (1-x)^n x \log \frac{1}{x} \leq \frac{\log n}{n}\,.$$

B.2 Experimental Details

The outline of this section is as follows.

- Section B.2.1: the three task domains considered in the experiments.
- Section B.2.2: training and decoding hyperparameters for each of these tasks.
- Section B.2.3: hyperparameters of MAUVE.
- Section B.2.4: details of other automatic comparison measures we consider.
- Section B.2.5: other details (software, hardware, running time, etc.).

B.2.1 Task Domains

We consider an open-ended text generation task under three domains: web text, news and stories. As summarized in Table 4.2, we follow a slightly different setting for the task in each domain:

Web text Generation. The goal of this task is to generate articles from the publicly available analogue of the Webtext dataset[1] using pretrained GPT-2 models for various sizes. At generation time, we use as prompts the first 35 tokens of each of the 5000 articles from the Webtext test set, keeping maximum generation length to 1024 tokens (which corresponds, on average, to around 750 words). For comparison with human text, we use the corresponding human-written continuations from the test set (up to a maximum length of 1024 tokens).

News Generation. Under this task, the goal is to generate the body of a news article, given the title and metadata (publication domain, date, author names). We use a Transformer-based (Vaswani et al., 2017) causal language model, Grover (Zellers et al., 2019), which is similar to GPT-2, but tailored to generating news by conditioning on the metadata of the article as well. Our generations rely on pretrained Grover architectures of various sizes. The generation prompt comprises the headline and metadata of 5000 randomly chosen articles from the April2019 set of the RealNews dataset (Zellers et al., 2019), and the maximum article length was 1024 tokens. We reuse the publicly available Grover generations[2] for our evaluation.

Story Continuation. Given a situation and a (human-written) starting of the story as a prompt, the goal of this task is to continue the story. Here, we use a GPT-2 medium model fine-tuned for one epoch on the WritingPrompts dataset (Fan et al., 2018). We use as generation prompts the first 50 tokens of 5000 randomly chosen samples of the test set of WritingPrompts. The machine generations are allowed to be up to 512 tokens long. The corresponding test examples, truncated at 512, tokens are used as human-written continuations.

B.2.2 Training and Decoding Hyperparameters

We use size-based variants of Transformer language models (Vaswani et al., 2017) for training each task (domain). At decoding time, we explore a text continuation setting, conditioned on a prompt containing human-written text. All experiments were built using pretrained (and if applicable, finetuned) models implemented in the HuggingFace Transformers library (Wolf et al., 2020). The tasks are summarized in Table 4.2.

Story Continuation Finetuning. We finetune GPT-2 medium on the training set of the WritingPrompts dataset using the cross entropy loss for one epoch over the training set with an effective batch size of 32 and a block size of 512. We use the default optimizer and learning rate schedules of the HuggingFace Transformers library, i.e., the Adam optimizer with a learning rate of 5×10^{-5}.

Decoding Hyperparameters. We consider pure sampling (i.e., ancestral sampling from the model distribution), greedy decoding (i.e., choosing the argmax token recursively), and nucleus sampling (Holtzman

[1] `https://github.com/openai/gpt-2-output-dataset`

[2] available at `https://github.com/rowanz/grover/tree/master/generation_examples`

et al., 2020) with parameter $p \in \{0.9, 0.92, 0.95, 0.99\}$ for web text generation and story continuation, and $p \in \{0.9, 0.92, 0.94, 0.96, 0.98\}$ for news generation.

B.2.3 MAUVE Hyperparameters

MAUVE's hyperparameters are the scaling constant c, the embedding model M, and the quantization algorithm (including the size of the quantized distribution).

Scaling Constant

Note that MAUVE's dependence on c is order-preserving since the map $x \mapsto \exp(-cx)$ is strictly monotonic in x. That is, if $\text{MAUVE}_{c_1}(P, Q_1) > \text{MAUVE}_{c_1}(P, Q_2)$, then it holds that $\text{MAUVE}_{c_2}(P, Q_2) > \text{MAUVE}_{c_2}(P, Q_2)$ for all scaling constants $c_1, c_2 > 0$. In other words, the choice of the scaling constant affects the numerical value of MAUVE but leaves the relative ordering between different models unchanged. We choose $c = 5$ throughout because it allows for a meaning comparison between the numerical values of MAUVE; Appendix B.3.3 gives the values of MAUVE for various values of c.

Embedding Model

We compute text embeddings from the GPT-2 large model. We find in Appendix B.3.3 that feature representations obtained from other large transformer models such as RoBERTA (Liu et al., 2019) also achieves similar results.

Quantization

We experiment with three quantization algorithms.

MAUVE-k-means. We first run PCA on the data matrix obtained from concatenating the hidden state representations of the human text and model text. We keep 90% of the explained variance and normalize each datapoint to have unit ℓ_2 norm. We then run k-means with FAISS for a maximum of 500 iterations for 5 repetitions; the repetition with the best objective value is used for the quantization. We quantize the human text distribution and the model text distribution by a histogram obtained from cluster memberships. We vary the number of clusters in $\{100, 250, 500, 1000\}$. Too few clusters makes the distributions seem closer than they actually are while too many clusters leads to many empty clusters (which makes all distributions seem equally far away). Yet, we find in Appendix B.3.3 that MAUVE with all these values of k correlate strongly with each other; we use as default $k = 500$ clusters as it is neither too small nor too large.

MAUVE-DRMM. We use the code released by the authors of (Hämäläinen and Solin, 2020).[3] We take 10 components per layer and 3 layers for a total of 1000 components. We train the DRMM for 20 epochs using the hyperparameters suggested by the authors, i.e., a batch size of 64 with a learning rate

$$\gamma_t = \gamma_0 \min\{1, (2 - 2t/T)^2\},$$

where T is the total number of updates and the initial learning $\gamma_0 = 0.005$. That is, the learning rate is set to a constant for the first half of the updates and then annealed quadratically. For more details, see (Hämäläinen and Solin, 2020, Appendix C).

[3] https://github.com/PerttuHamalainen/DRMM

MAUVE-Lattice. We use the code provided by the authors of (Sablayrolles et al., 2019).[4] We train a 4-dimensional feature representation of the hidden states for for 200 epochs using the triplet loss of (Sablayrolles et al., 2019), so that the learnt feature representations are nearly uniformly distributed. We use a 2-layer multilayer perceptron with batch normalization to learn a feature representation. We train this MLP for 200 epochs with hyperparameters suggested by the authors, i.e., a batch size of 64 and an initial learning rate of 0.1. The learning rate is cut to 0.05 after half the training and 0.01 after 75% of the training.

The learnt feature representations are then quantized using the lattice spherical quantizer into 744 bins. This work as follows: let S_r denote the integral points of the unit sphere of radius $r = \sqrt{50}$ in $\mathbb{R}^4$. A hidden state vector x is run through the trained MLP f to get its feature representation $f(x)$. Next, $f(x)$ is quantized to $\arg\min_{u \in S_r} \|f(x) - u/r\|_2^2$.

B.2.4 Automatic Comparison Measures: Details and Hyperparameters

We now describe the other automatic comparison measures we compared MAUVE to, as well as their hyperparameters.

- **Generation Perplexity (Gen. PPL.)**: We compute the perplexity of the generated text under the GPT-2 large model.
- **Zipf Coefficient**: we report the slope of the best-fit line on log-log plot of a rank versus unigram frequency plot. Note that the Zipf coefficient only depends on unigram count statistics and is invariant to, for instance, permuting the generations. We use the publicly available implementation of (Holtzman et al., 2020).[5]
- **Repetition Frequency (Rep.)**: The fraction of generations which devolved into repetitions. Any generation which contains at least two contiguous copies of the same phrase of any length appearing at the end of a phrase is considered a repetition. We consider repetitions at the token level.
- **Distinct-n**: The fraction of distinct n-grams from all possible n-grams across all generations. We use $n = 4$.
- **Self-BLEU**: Self-BLEU is calculated by computing the BLEU score of each generations against all other generations as references. We report the Self-BLEU using 4-grams. This operation is extremely expensive, so we follow the protocol of (Holtzman et al., 2020): sample 1000 generations and compute the BLEU against all other 4999 generations. A lower Self-BLEU score implies higher diversity. This operation takes around 7 hours to compute on a single core of an Intel i9 chip (see hardware details in the next subsection).
- **Discriminator Accuracy**: We train a binary classifier to classify text as human or not. A smaller discrimination accuracy means that model text is harder to distinguish from human text. A separate classifier is trained for each model and decoding algorithm pair. For the story continuation task, we train a classification head on a frozen GPT-2 large model using the logistic loss. We use 25% of the data as a test set and the rest for training; a regularization parameter is selected with 5-fold cross validation. For the news dataset, we follow the protocol of (Zellers et al., 2019), i.e., a Grover mega model finetuned with a binary classification head. Results with other discriminators are reported in Appendix B.3.

[4] https://github.com/facebookresearch/spreadingvectors

[5] https://github.com/ari-holtzman/degen/blob/master/metrics/zipf.py

B.2.5 Miscellaneous Details

Software. We used Python 3.8, PyTorch 1.7 and HuggingFace Transformers 4.3.2.

Hardware. All the experiments requiring a GPU (finetuning, sampling generations and computing embeddings) were performed on a machine with 8 Nvidia Quadro RTX GPUs (24G memory each) running CUDA 10.1. Each only used one GPU at a time. On the other hand, non-GPU jobs such as computation of MAUVE and self-BLEU were run on a workstation with Intel i9 processor (clock speed: 2.80GHz) with 32 virtual cores and 126G of memory.

Evaluation time for MAUVE. Computation of MAUVE using k-means with 5000 generations takes $1-3$ minutes on *a single core* of an Intel i9 CPU (clock speed: 2.80GHz), using cached hidden state representations from a GPT-2 large (which are available during generation). On the other hand, MAUVE-DRMM takes 1.75 hours on a single CPU core while MAUVE-Lattice runs in about 5 minutes on a single TITAN Xp GPU. MAUVE-k-means and MAUVE-DRMM can also run much faster on multiple CPU cores and can leverge GPUs although we did not use these features.

B.3 Additional Experimental Results

The outline of this section is as follows.

- Section B.3.1: full results across model size and decoding (elaborating on Section 4.4.1).
- Section B.3.2: full results across text length (elaborating on Section 4.4.1).
- Section B.3.3: study of approximations in MAUVE (elaborating on Section 4.4.2).
- Section B.3.4: some miscellaneous plots such use of MAUVE for hyperparameter tuning.

B.3.1 Comparison of Measures Across Model Size and Decoding

Full versions of Table 4.3 and Table 4.4 can be found between Table B.1 for statistics-based measures and Table B.4 for the language modeling measures. The corresponding tables for the news and story domains are Tables B.2 and B.3 respectively.

Note: The main paper and the appendix treat the statistics-based measures differently (Gen. PPL., Zipf, Self-BLEU, etc). For each statistic T, the main paper (Tables 4.3 and 4.4) gives the difference $|T(Q) - T(P)|$ between the statistic on model text and human text, while in Tables B.1, B.2, B.3 of the supplement, we show $T(Q)$ in the row corresponding to Q and $T(P)$ in the row corresponding to human.

Results. From Table B.1, we observe that among the decoding approaches, nucleus sampling achieves the best MAUVE followed by sampling and lastly by greedy decoding. This trend is consistent with the fraction of distinct 4-grams. On the other hand, in comparison with the perplexity of human text, Gen. PPL is too high for sampling and too low for greedy decoding; it does not give us a way to directly compare which of these two is better. MAUVE, however, rates greedy decoding as far worse than ancestral sampling. This is consistent with the empirical observation that greedy decoding produces extremely degenerate text (Welleck et al., 2020b). Adversarial perplexity sampling produces unintelligible text which nevertheless has perfect Gen. PPL, thus demonstrating its unsuitability for as a comparison measure.

GPT-2 Size	Decoding	Gen. PPL	Zipf Coef.	Rep.	Distinct-4	Self-BLEU	Human/BT(↑)	MAUVE (↑)
small	Sampling	$101.880_{0.627}$	$0.926_{0.001}$	$0.001_{0.000}$	$0.941_{0.001}$	$0.327_{0.003}$	−27.52	$0.589_{0.018}$
	Greedy	1.224	1.037	0.942	0.072	$0.465_{0.000}$	–	0.008
	Nucleus, 0.9	$23.788_{0.144}$	$1.012_{0.002}$	$0.010_{0.001}$	$0.859_{0.002}$	$0.436_{0.004}$	−15.78	$0.878_{0.006}$
	Adversarial	**12.554**	1.073	0.006	0.365	0.525	–	0.043
medium	Sampling	$129.263_{0.798}$	$0.872_{0.001}$	$0.001_{0.000}$	$0.953_{0.001}$	$0.281_{0.002}$	−30.77	$0.373_{0.010}$
	Greedy	1.241	0.978	0.903	0.091	0.415	–	0.012
	Nucleus, 0.9	$21.073_{0.134}$	$\mathbf{0.957}_{0.001}$	$0.005_{0.001}$	$\mathbf{0.884}_{0.001}$	$\mathbf{0.402}_{0.003}$	−3.43	$0.915_{0.006}$
	Adversarial	**12.554**	1.006	0.005	0.381	0.444	–	0.044
large	Sampling	$30.080_{0.196}$	$0.930_{0.002}$	$\mathbf{0.002}_{0.001}$	$0.916_{0.001}$	$0.358_{0.001}$	−6.93	$0.845_{0.010}$
	Greedy	1.232	0.983	0.881	0.100	0.413	–	0.012
	Nucleus, 0.95	$13.499_{0.058}$	$0.967_{0.002}$	$0.006_{0.001}$	$0.870_{0.001}$	$0.412_{0.002}$	12.55	$0.936_{0.003}$
	Adversarial	**12.554**	0.965	0.005	0.395	0.429	–	0.035
xl	Sampling	$31.886_{0.447}$	$0.930_{0.001}$	$0.002_{0.001}$	$0.913_{0.001}$	$0.360_{0.003}$	8.97	$0.882_{0.006}$
	Greedy	1.278	0.975	0.859	0.115	0.417	–	0.016
	Nucleus, 0.95	$14.143_{0.043}$	$0.966_{0.002}$	$0.005_{0.000}$	$0.868_{0.001}$	$0.413_{0.002}$	**15.66**	$\mathbf{0.940}_{0.006}$
	Adversarial	**12.554**	0.986	0.005	0.397	0.448	–	0.057
Human	n/a	12.602	0.952	0.002	0.878	0.382	47.25	–

Table B.1: Comparison measures across different model sizes, and decoding approaches for web text generations. Subscripts indicate the s.d. across 5 runs for the sampling-based methods; greedy decoding, being deterministic, always returns the same value for a given model. For nucleus sampling, we show the best hyperparameter value from $\{0.9, 0.92, 0.95, 0.99\}$ as per MAUVE. The column "Human/BT" gives the Bradley-Terry score obtained from a pairwise human evaluation (Section 4.4.3). Boldfaced numbers indicate best performance according to the measure, or closest to the human reference, when applicable. MAUVE shows that larger models perform better, across decoding approaches; moreover, nucleus sampling is the best decoding algorithm as per MAUVE.

Grover Size	Decoding	Gen. PPL	Zipf Coef.	Rep.	Distinct-4	Self-BLEU	% Disc. Acc.(↓)	MAUVE(↑)
base	Sampling	37.505	0.942	0.002	0.882	0.419	99.925	0.700
	Greedy	1.413	1.038	0.518	0.081	0.548	100.000	0.005
	Nucleus, 0.96	23.064	0.974	0.006	**0.847**	0.462	99.950	0.701
large	Sampling	27.796	0.946	**0.002**	0.878	0.429	99.450	0.794
	Greedy	1.575	1.012	0.366	0.124	0.504	100.000	0.005
	Nucleus, 0.98	20.792	**0.962**	0.002	0.859	0.450	98.475	0.750
mega	Sampling	22.656	0.950	0.001	0.879	0.427	97.300	0.808
	Greedy	1.796	1.003	0.316	0.176	0.500	100.000	0.005
	Nucleus, 0.96	**14.834**	0.972	0.003	0.848	**0.469**	**88.675**	**0.813**
Human	n/a	15.356	0.956	0.002	0.842	0.473	–	–

Table B.2: News generation evaluation across different Grover model sizes, and decoding approaches. For nucleus sampling, we show the best hyperparameter value from $\{0.9, 0.92, 0.94, 0.96, 0.98\}$ as per MAUVE. Disc. Acc. denotes the discrimination accuracy (%) of a Grover mega model trained to distinguish human text from machine text generated with the model and decoding algorithm of each row. Boldfaced numbers indicate performance closest to the human reference when applicable, or the best performance according to the measure. MAUVE favors nucleus sampling over ancestral sampling and greedy decoding.

The results in Tables B.2 and B.3 for the news and story domains are qualitatively similar to the webtext

Decoding	Gen. PPL	Zipf Coef.	REP	Distinct-4	Self-BLEU	% Disc. Acc. (↓)	MAUVE(↑)
Sampling	$38.983_{0.143}$	$\mathbf{1.066}_{0.002}$	$\mathbf{0.001}_{0.000}$	$0.833_{0.001}$	$0.518_{0.003}$	$0.781_{0.004}$	$0.905_{0.010}$
Nucleus, 0.9	$15.433_{0.042}$	$1.201_{0.002}$	$0.006_{0.001}$	$0.719_{0.001}$	$0.637_{0.002}$	$0.752_{0.004}$	$0.887_{0.008}$
Nucleus, 0.92	$17.422_{0.060}$	$1.179_{0.002}$	$0.004_{0.001}$	$0.742_{0.001}$	$0.620_{0.003}$	$0.720_{0.006}$	$0.901_{0.005}$
Nucleus, 0.95	$\mathbf{21.599}_{0.127}$	$1.147_{0.002}$	$0.003_{0.000}$	$\mathbf{0.775}_{0.002}$	$0.589_{0.005}$	$\mathbf{0.686}_{0.006}$	$\mathbf{0.920}_{0.004}$
Top-100	$16.527_{0.041}$	$1.252_{0.001}$	$0.002_{0.000}$	$0.743_{0.001}$	$0.631_{0.001}$	$0.782_{0.002}$	$0.884_{0.007}$
Top-500	$23.833_{0.076}$	$1.153_{0.001}$	$0.001_{0.000}$	$0.794_{0.001}$	$\mathbf{0.576}_{0.002}$	$0.697_{0.005}$	$0.919_{0.005}$
Greedy	1.739	1.362	0.988	0.101	0.742	0.997	0.005
Human	19.704	1.101	0.001	0.783	0.571		

Table B.3: Story continuation evaluation across different and decoding approaches with GPT-2 medium. Disc. Acc. denotes the discrimination accuracy (%) of a classifier (a frozen GPT-2 large model with classification head) trained to distinguish human text from machine text generated with the decoding algorithm of each row. Boldfaced numbers indicate performance closest to the human reference when applicable, or the best performance according to the measure. MAUVE favors nucleus and top-K sampling over ancestral sampling and greedy decoding.

GPT-2 Size	Decoding	SP(↑)	JS(↓)	ε-PPL(↓)	Human/BT(↑)	MAUVE (↑)
small	Greedy	0.431	0.394	1049.589	–	0.008
	Sampling	0.653	0.425	19.401	−27.52	$0.589_{0.018}$
	Nucleus, 0.9	0.652	0.414	25.938	−15.78	$0.878_{0.006}$
medium	Greedy	0.465	0.371	708.057	–	0.012
	Sampling	0.670	0.402	14.631	−30.77	$0.373_{0.010}$
	Nucleus, 0.9	0.670	0.391	18.821	−3.43	$0.915_{0.006}$
large	Greedy	0.483	0.359	580.020	–	0.012
	Sampling	0.679	0.381	12.658	−6.93	$0.845_{0.010}$
	Nucleus, 0.95	0.679	0.374	14.938	12.55	$0.936_{0.003}$
xl	Greedy	0.496	**0.349**	497.696	–	0.016
	Sampling	**0.686**	0.369	**11.412**	8.97	$0.882_{0.006}$
	Nucleus, 0.95	0.686	0.363	13.677	**15.66**	$\mathbf{0.940}_{0.006}$
	Adversarial	n/a	n/a	n/a	–	0.057

Table B.4: MAUVE versus comparison measures based on language modeling (SP, JS and ε-PPL) across different model sizes, and decoding approaches for web text generations. SP, JS and ε-PPL are deterministic because they do not require generations from a decoding algorithm; moreover they cannot measure the quality of the adversarial decoding. The column "Human/BT" gives the Bradley-Terry score obtained from a pairwise human evaluation (Section 4.4.3). Boldfaced numbers indicate best performance according to the measure.

Discriminator	BERT		GPT-2			Grover		
	Base	Large	Small	Medium	Large	Base	Large	Mega
Correlation	0.803	0.817	0.831	0.829	0.822	0.928	0.956	0.925

Table B.5: Spearman rank correlation between the discrimination accuracy for various discriminators and MAUVE for news generation. All entries have a p-value of $< 2 \times 10^{-6}$.

Decoding	Greedy	Beam $b = 4$	Beam $b = 4$ + no 4-gram repeat	Beam $b = 8$	Beam $b = 8$ + no 4-gram repeat	Ancestral	Nucleus
Mauve	0.008	0.021	0.026	0.366	0.341	$0.589_{0.02}$	$\mathbf{0.878_{0.007}}$

Table B.6: MAUVE and beam search: we compare beam search with beam sizes $b = 4, 8$ (with and without allowing 4-gram repetitions) with other decoding algorithms of Table B.1 for web text generation with GPT-2 small. The subscript denotes the standard deviation over 5 random seeds, and is omitted for the deterministic greedy decoding and beam search.

GPT-2 size	Decoding	RoBERTa	GPT-2
small	Sampling	0.174	0.589
	Greedy	0.056	0.008
	Nucleus, 0.9	0.723	0.878
medium	Sampling	0.292	0.372
	Greedy	0.114	0.011
	Nucleus, 0.9	0.891	0.915
large	Sampling	0.684	0.845
	Greedy	0.125	0.012
	Nucleus, 0.95	0.920	0.936
xl	Sampling	0.780	0.881
	Greedy	0.170	0.016
	Nucleus, 0.95	**0.947**	**0.940**

Table B.7: Comparison of MAUVE computed with dense embeddings from RoBERTa (Liu et al., 2019) large with the default GPT-2 large. Boldfaced numbers indicate best performance according to the measure. The two feature representations have a Spearman rank correlation of 0.993. See Figure 4.7 for a visual representation of a subset of this table.

domain. MAUVE, like discrimination accuracy, rates larger models as better and nucleus sampling as better than ancetral sampling and greedy decoding. An exception to this rule is Grover large, where MAUVE thinks ancestral sampling is better than nucleus sampling. The statistics-based measures Zipf coefficient, Repetition and the fraction of distinct 4-grams all prefer smaller Grover sizes.

Next we turn to the language modeling comparison measures in Table B.4. JS consistently favors greedy decoding, which produces far worse text than other decoding algorithms. Likewise, ε-PPL favors ancestral sampling, which also produces somewhat degenerate text (Holtzman et al., 2020), while SP appears to be unable to distinguish between ancestral sampling and nucleus sampling. This makes SP, JS and ε-PPL unsuitable to compare generated text to human text.

While most measures behave nearly as expected across model architectures (larger models produce better generations for the same decoding algorithm), Self-BLEU prefers generations from GPT-2 medium over GPT-2 large or xl. This indicates that while measures based on word/token statistics are important diagnostic tools, they do not capture the quality of generated text entirely.

Discriminator Accuracy: Choice of Discriminator. We show the Spearman rank correlation between the discriminator accuracy for various choices of the discriminator in Table B.5. The results show that MAUVE has a strong correlation with the discrimination accuracy for a variety of discriminators, including

one based on a masked language model, BERT (Devlin et al., 2019). This correlation is particular strong for the Grover-based discriminators. We note that evaluating any one model and decoding algorithm pair requires fine-tuning a separate model. This can be particularly expensive for the larger models such as Grover mega. MAUVE, on the other hand, is inexpensive in comparison.

Beam Search. We also calculate MAUVE for beam search in Table B.6. MAUVE is able to quantify the qualitative observations of Holtzman et al. (2020): beam search produces extremely degenerate text, but slightly better than greedy decoding. Disallowing repetition of 4-grams substantially improves the quality of the produced text, since the most glaring flaw of beam search is that the text is highly repetitive. However, the quality of the resulting text is still far worse than produced by ancestral sampling, and hence also nucleus sampling.

B.3.2 Behavior Across Text Length

We now turn to the plot of comparison measures versus text length in Figure B.1. We expect the quality of the generation to degrade as the maximum length of the text (both machine and human-written) increases.

Comparison Measures. Figure B.1 plots MAUVE, Gen. PPL. and the Sparsemax score (Martins et al., 2020). In addition we also plot the Fréchet distance, a variant of the Fréchet Inception Distance (FID) (Heusel et al., 2017) which is the de facto standard evaluation metric for GANs in computer vision. The FID is computed as the Wasserstein-2 distance between Gaussians fit to the feature representation from using an Inception network; we adapt it to our setting by using embeddings from GPT-2 large instead. For Gen. PPL., we plot the difference of Gen. PPL., i.e., $|T_{\text{ppl}}(Q_{\leq \ell}) - T_{\text{ppl}}(P_{\leq \ell})|$, $T_{\text{ppl}}(P_{\leq \ell})$ denotes the perplexity of the text $\boldsymbol{x} \sim P$ truncated at a length of ℓ. The perplexity is measured using GPT-2 large model as the external language model.

Results. MAUVE indeed shows this expected behavior. However, the Fréchet distance (Heusel et al., 2017) actually decreases for nucleus sampling for all GPT-2 sizes and ancestral sampling for GPT-2 xl. This shows that it is not suitable as an evaluation metric for text. While Gen. PPL. mostly agrees with MAUVE about quality versus text length, we observe non-monotonic behavior for nucleus sampling with GPT-2 small and large. Finally, sparsemax score (Martins et al., 2020) does not depend on the samples generated and is therefore independent of the maximum text length.

B.3.3 Effect of Approximations of MAUVE

We expand upon the approximation results from the main paper in §4.4.2.

Embedding Model. Table B.7 shows MAUVE compute with RoBERTa large in addition to the default GPT-2 large. We restrict the maximum text length of the RoBERTa model to 256 BPE tokens, since RoBERTa cannot handle sequences of length 1024 tokens. We observe similar trends with both: larger models are rated higher and nucleus sampling is preferred over ancestral sampling while greedy decoding is rated very low. The Spearman rank correlation between MAUVE computed with the two feature representations is 0.993, indicating that MAUVE is robust to feature representations. We observe that RoBERTa penalizes ancestral sampling more while rating greedy decoding higher across all model sizes. We leave a study of the biases induced by different feature representations to future work.

Quantization Algorithm. We compare different choices of the quantization to k-means with $k = 500$, which is our default. The Spearman rank correlation between MAUVE computed with k-means for k ranging from 100 to 5000 correlates nearly perfectly with that of $k = 500$. In particular, the Spearman correlation is exactly 0.99 or 1.00. Likewise, MAUVE computed with DRMM or lattice quantization has a near-perfect Spearman correlation of at least 0.99 with k-means. While the actual numerical value of MAUVE could vary with the quantization algorithm, these results show that the *rankings induced by various variants of* MAUVE *are nearly identical.*

See Figure B.3 (Left) for how MAUVE-k-means depends on the number of clusters, k. If k is too small ($k < 100$), all methods are scored close to 1. If k is too large $k > 2000$), all methods are scored close to 0. There is a large region between these two extremes where MAUVE-k-means is effective.

Effect of Number of Generations. Figure B.2 plots the value of MAUVE versus the sample size n, with the number of clusters in k-means chosen as $k = n/10$. We observe that a smaller sample size gives an optimistic estimate of MAUVE; this is consistent with (Djolonga et al., 2020, Prop. 8). We also note that a smaller sample size leads to a larger variance in MAUVE.

B.3.4 Miscellaneous Plots

Figure B.3 plots MAUVE for nucleus and top-K sampling for various values of the hyperparameters p and K.

B.4 Human Evaluation Details

B.5 Human Evaluation: Protocol and Full Results

Here, we describe the human evaluation protocol and results of §4.4.3 in detail. The outline for this section is

- Section B.5.1: Overview of the human evaluation setup.
- Section B.5.2: Details of the statistical model we fit to the raw data.
- Section B.5.3: Full results of the human evaluation.
- Section B.5.4: Additional details of the human evaluation protocol.

B.5.1 Overview

We performed a human evaluation for web text generations where human annotators are instructed to select one from a pair of texts. The pairs might come from human and machine text, or different sources of machine text; each is based on the same prompt for generation (recall that we obtained the prompt as a prefix from the human text).

The annotators were presented with a pairs of continuations of the same prompt and were instructed to choose which one is (a) more interesting, (b) more sensible, and, (c) more likely to be written by a human. Each question could have a different answer.

We considered all four GPT-2 model sizes with pure sampling and nucleus sampling. We collected 90 annotations for each of the 8 model-human pairs and $\binom{8}{2}$ model-model pairs on the Amazon Mechanical

Turk platform using the interface shown in Figure B.4. We fit a Bradley-Terry model to obtain a ranking from the pairwise preferences of the crowd-workers. We report the correlation of MAUVE with obtained Bradley-Terry scores.

B.5.2 From Pairwise Preferences to Ranking: the Bradley-Terry Model

We compute the Bradley-Terry (BT) scores from the pairwise preferences obtained from the human evaluation along each of the three axes interesting, sensible and more likely to be written by a human.

Bradley-Terry Model Review. Given n players with scores $w_1, \cdots, w_n$, the the Bradley-Terry model (Marden, 1995) models the outcome of a head-to-head comparison of any two players using a sigmoid[6]

$$\text{Prob}(i \text{ beats } j) = \frac{1}{1 + e^{-(w_i - w_j)/100}} \,.$$

The model also assumes the outcome of each head-to-head comparison of any pair of players is independent of all other comparisons. Note that the model is invariant to additive shifts of the scores, i.e., the model probabilities induced by scores $w_1 + C, \cdots, w_n + C$ is same as the that induced by $w_1, \cdots, w_n$ for any constant C. For uniqueness, we normalize the scores so that their mean is 0.

Fitting the Model. The Bradley-Terry model can be fit to data using Zermelo's algorithm (Hunter, 2004). Suppose that we are given a dataset of head-to-head comparisons summarized by numbers N_{ij} denoting the number of times player i has defeated player j. Then, the negative log-likelihood $\ell(w_1, \cdots w_n)$ of the data under the Bradley-Terry model can be written as

$$\ell(w_1, \cdots, w_n) = -\sum_{i=1}^{n} \sum_{j=1}^{n} N_{ij} \log(1 + e^{-(w_i - w_j)/100}) \,.$$

This is convex in the parameters $w_1, \cdots, w_n$ since the log-sum-exp function is convex. Zermelo's algorithm (Hunter, 2004) can be used to compute the maximum likelihood estimate. Denote $\widetilde{w}_i = w_i/100$. Starting from an initial estimate $\widetilde{w}_1^{(0)}, \cdots, \widetilde{w}_n^{(0)}$, each iteration of Zermelo's algorithm performs the update

$$u_i^{(t)} = \log\left(\sum_{j \neq i} N_{ij}\right) - \log\left(\sum_{j \neq i} \frac{N_{ij} + N_{ji}}{\exp(\widetilde{w}_i^{(t)}) + \exp(\widetilde{w}_j^{(t)})}\right)$$

followed by the mean normalization

$$\widetilde{w}_i^{(t+1)} = u_i^{(t)} - \frac{1}{n}\sum_{j=1}^{n} u_j^{(t)} \,.$$

Processing Raw Data. We collect the result of a head-to-head comparison using 5 options: Definitely A/B, Slightly A/B or a Tie. We combine Definitely A and Slightly A into a single category denoting that A wins, while ties were assigned to either A or B uniformly at random.

[6] the scaling factor 100 is arbitrary and does not change the model

		BT/Human-like	BT/Interesting	BT/Sensible
Human		47.251	25.503	43.229
xl	Nucleus, $p = 0.95$	**15.664**	**23.046**	**31.888**
	Sampling	8.966	9.529	7.753
large	Nucleus, $p = 0.95$	12.553	6.785	8.781
	Sampling	-6.935	-1.532	-7.106
medium	Nucleus, $p = 0.9$	-3.429	-12.824	-7.293
	Sampling	-30.769	-34.323	-32.004
small	Nucleus, $p = 0.9$	-15.783	-0.697	-7.442
	Sampling	-27.518	-15.487	-37.805

Table B.8: Fitted Bradley-Terry (BT) scores for each of the three axes rated by human annotators: "Human-like" denotes measures how likely the text is to be written by a human, while "Interesting" and "Sensible" quantify how interesting or sensible the text is. The Spearman rank correlations between each of these scores are (p-value $\leq 5 \times 10^{-4}$ for each): Human-like and Interesting: 0.917, Human-like and Sensible: 0.917, Interesting and Sensible: 0.967.

	Gen. PPL	Zipf Coef.	REP	Distinct-4	Self-BLEU	MAUVE
BT/Human-like	0.810	0.833	-0.167	0.738	0.595	**0.952**
BT/Interesting	0.643	0.524	-0.143	0.524	0.405	**0.810**
BT/Sensible	0.738	0.690	-0.071	0.595	0.524	**0.857**

Table B.9: Spearman rank correlation between the Bradley-Terry scores from the human evaluation and the various automatic comparison measures.

B.5.3 Full Results of the Human Evaluation

BT Model for Human Eval. In our setting, each "player" is a source of text, i.e., one human, plus, eight model and decoding algorithm pairs (four model sizes GPT-2 small/medium/large/xl coupled with pure sampling or nucleus sampling). We compute the BT score of each player as the maximum likelihood estimate of corresponding the parameters $w_1, \cdots, w_n$ based on head-to-head human evaluation data.

A higher BT score indicate a stronger preference from human annotators. The BT scores are reported in Table B.8. The Spearman rank correlations between each of these scores are (p-value $\leq 5 \times 10^{-4}$ for each):

- Human-like and Interesting: 0.917,
- Human-like and Sensible: 0.917,
- Interesting and Sensible: 0.967.

Interpreting BT scores. The BT scores reported in Table B.8 give us predictions from the sigmoid model above. For example, consider the column "BT/Human-like". The best model-generated text, GPT-2 xl with nucleus sampling, will lose to human text with probability 0.578. At the other end, GPT-2 small with nucleus sampling will lose to human text with probability 0.679. This shows that there is still much room for improvement in machine generated text.

Discussion. In general, the BT scores from human evaluations and MAUVE both indicate that (a) nucleus

sampling is better than pure sampling for the same model size, and, (b) larger model sizes are better for the same decoding algorithm. There is one exceptions to this rule, as per both the human evaluations and MAUVE: GPT-2 small is better than GPT-2 medium for pure sampling.

Correlation Between Comparison Measures. We compare the Spearman rank correlation between the various automatic comparison measures and the BT scores from human evaluations in Table B.9. In terms of being human-like, we observe that MAUVE correlates the best (0.95) with human evaluations. While this is also the case for Zipf coefficient, we note that it is based purely on unigram statistics; it is invariant to the permutation of tokens, which makes it unsuitable to evaluate generations.

We note that MAUVE does disagree with human evaluations on specific comparisons. For instance, MAUVE rates nucleus sampling with GPT-2 medium as being better than pure sampling from GPT-2 large and xl. The same is also the case with Gen. PPL. We leave a detailed study of this phenomenon to future work.

B.5.4 Additional Details

We describe more details for the human evaluation. The terminology below is taken from (Shimorina and Belz, 2021).

Number of Outputs Evaluated. We compare 9 players: one player is "human", representing human-written text, whereas the other 8 are text generated by the model using the first 35 tokens of the corresponding human generation as a prompt. Each of the 8 non-human players come from a GPT-2 model of different sizes (small, medium, large, xl) and two decoding algorithms (pure sampling and nucleus sampling). We perform 90 comparisons between each pair of players, so each player is evaluated $90 \times 8 = 720$ times.

Prompt Filtering. We manually selected 1831 out of 5000 prompts which are well-formed English sentences from the webtext test set[7]. For every head-to-head comparison, we sample 90 prompt without replacement and then sample the corresponding completions (for human-generated text, we use the test set of webtext). We only consider a pair of players for human evaluation if the generation from each player is at least 200 BPE tokens long (and we truncate each generation at a maximum length of 256 BPE tokens).

Number of Evaluators. 214 unique evaluators participated in the evaluation. Of these, 11 evaluators supplied at least 50 annotations 95 evaluators supplied at least 10 annotations.

Evaluator Selection and Pay. We conduct our human evaluation on Amazon Mechanical Turk. Since the task only requires elementary reading and understanding skills in English, we open the evaluations to non-experts. Each crowd-worker was paid 0.40 per annotation. The pay was estimated based on a $\$16$/hour wage for the 85^{th} percentile of response times from a pilot study (which was approx. 98 seconds per annotation). There evaluators are not previously known to the authors.

[7]The webtext dataset is scraped from the internet and is *not* curated. It contains poor prompts such as headers of webpages or error message, such as: "Having trouble viewing the video? Try disabling any ad blocking extensions currently running on your browser" or "Front Page Torrents Favorites My Home My Galleries Toplists Bounties News Forums Wiki". We exclude such prompts as they are unsuitable for human evaluation.

Training and Instructions. The evaluators were given instructions about the task and two detailed examples. No other training was provided due to the elementary nature of the task. The screenshots of these examples are given in Figure B.5 while the instructions read:

> **Task Info**: We are studying how good AI models are at generating text on the internet. You are given a snippet of text from a random document on the internet, called the "prompt" or the "context", as well as and two continuations, A and B. One or both of these is written by an AI. You must choose (a) which of two continuations is more interesting, (b) which makes more sense given the prompt, and, (c) which is more likely to have been written by a human, as per your assessment.
>
> **Guidelines**:
>
> - There are five choices for each question: Definitely A/B, Slightly A/B, or Tie. Please use the "Tie" option extremely sparingly! (No more than one in every ten pairs should be chosen as a tie along any of the three questions).
> - The questions can have different answers! Some text is very creative or interesting, but it doesn't quite fit the prompt or make sense.
> - Try to focus on quality over quantity. The text can be long but contain rambly gibberish.
> - Don't worry if the text ends abruptly, or has other artifacts of the website downloading process (text like 'Advertisement' for instance).
> - Please do your best, some of these are pretty challenging!
> - Answering each question should take around 1.5 minutes on average, as per our estimation. We have calibrated the pay to be $16 per hour with this speed.

Quality Control. All annotations made in under 25 seconds were excluded for quality control (the mean response time per annotation was 47 seconds).

Quality Criteria. We use three quality criteria. The questions asked to the evaluators are (verbatim):

1. Interestingness: "Which continuation is more interesting or creative, given the context?"
2. Sensible: "Which continuation makes more sense, given the context?"
3. Human-like: "Which continuation is more likely to be written by a human?"

Note that we do explicitly name the criteria in the evaluation form, although those names could be inferred from the definitions. We use these names only in the paper.

Further Details:

- Each of the criteria is a "Goodness" criteria as per the classification of (Belz et al., 2020). Goodness refers to the setting where there is no single, general mechanism for deciding when outputs are maximally good, only for deciding for two outputs which is better and which is worse. E.g. for Fluency, even if outputs contain no disfluencies, there may be other ways in which any given output could be more fluent.
- Each criterion assesses outputs as a whole, not just form or just content.
- The output quality is assessed without referring to anything other than the output itself, i.e. no system-internal or external frame of reference.
- Each criterion involves a subjective assessments of preferences by evaluators.
- The quality of outputs is assessed *without* considering their *effect* on something external to the system, e.g. the performance of an embedding system or of a user at a task.
- For each criteria, we provide 5 options: "Definitely/Slightly A/B" and "Tie (Use sparingly!)"

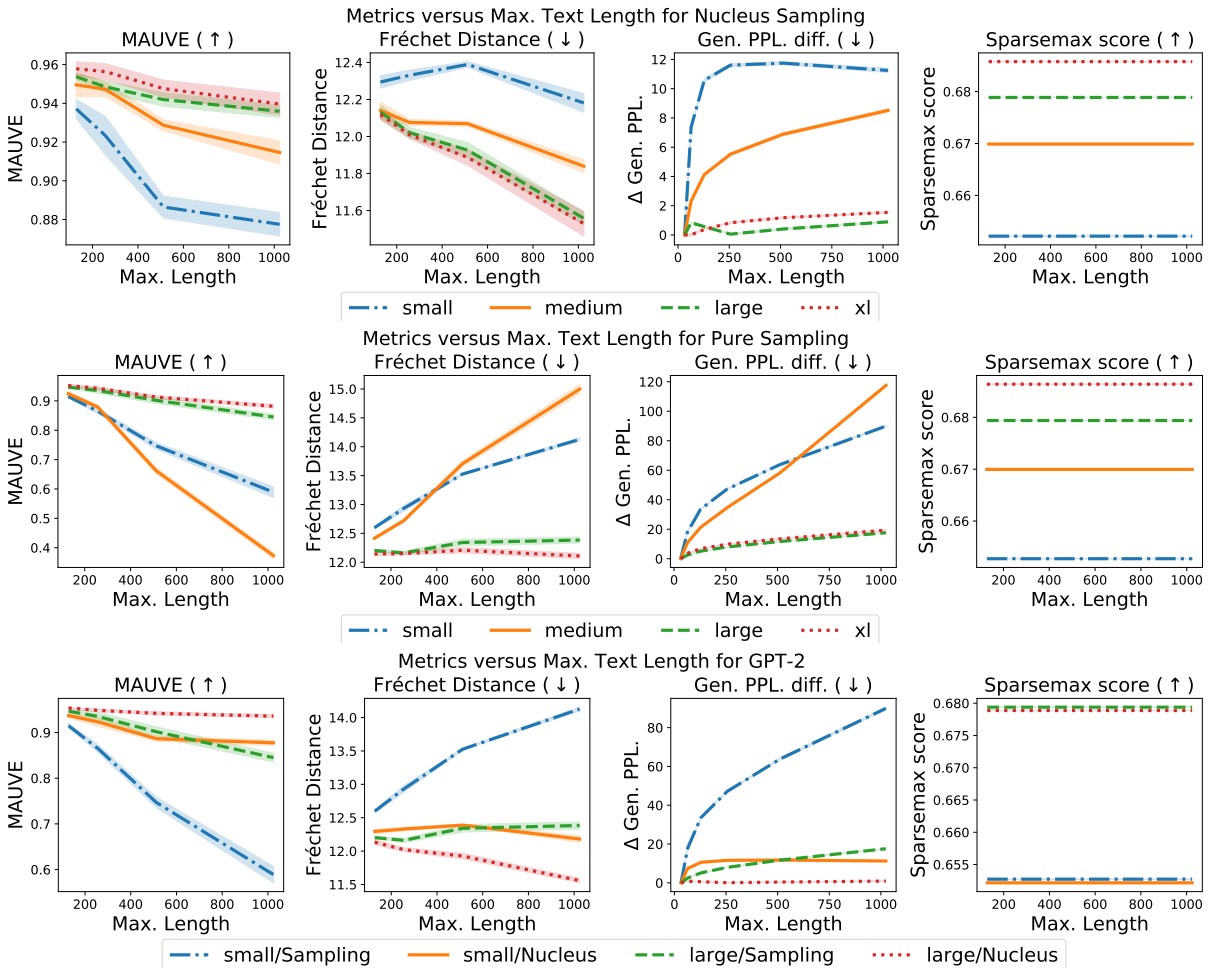


Figure B.1: Generation quality versus maximum generation length as per various comparison measures for web text generation with GPT-2. We expect the quality of the generation to degrade as the maximum length of the text (both machine and human-written) increases. MAUVE is the only comparison measure which correctly shows this behavior across all models and decoding algorithms. The shaded area denotes one standard deviation over generations from 5 random seeds.

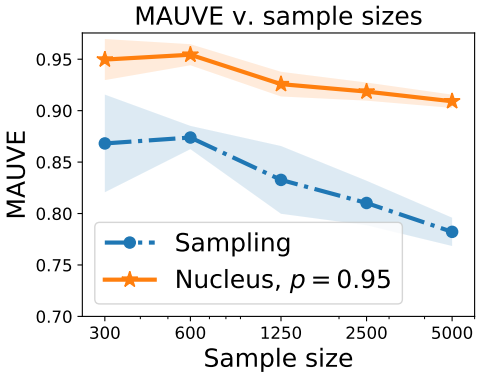


Figure B.2: Effect of the sample size on MAUVE.

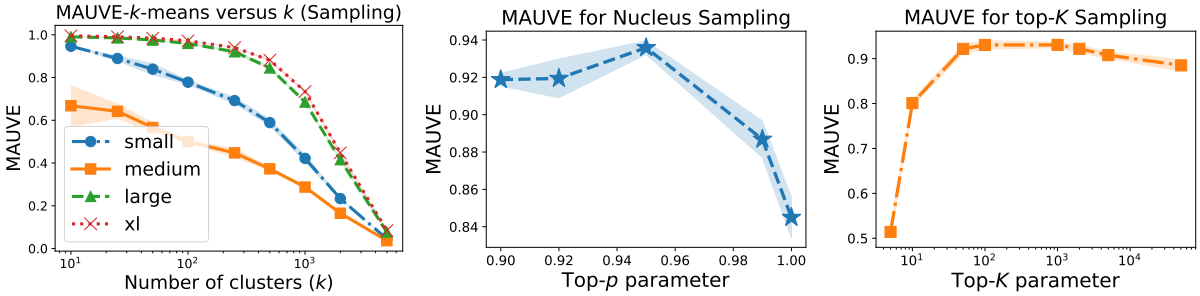


Figure B.3: Left: MAUVE-k-means for various values of the number of clusters k. We use $k = 500$ as our default because it is neither too small (every method is scored close to 1) nor too large (every method is scored close to 0). **Center & Right**: MAUVE for nucleus and top-K sampling for different values of p and K for GPT-2 large. MAUVE rates nucleus sampling with $p = 0.95$ and top-K sampling with $100 \leq K \leq 1000$ as the best choices. The shaded area denotes one s.d. over generations from 5 random seeds.

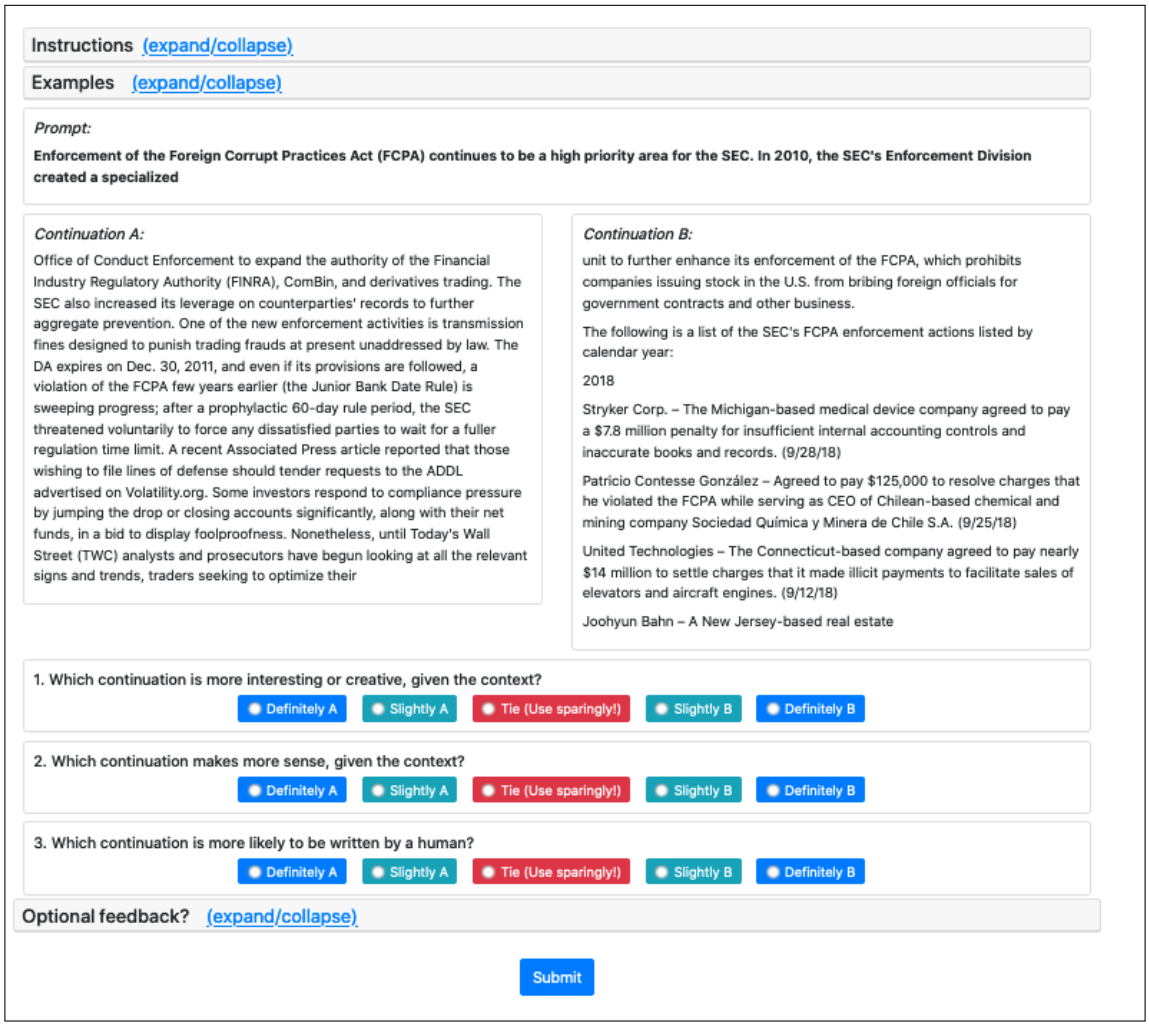


Figure B.4: Mechanical Turk interface for human evaluation.

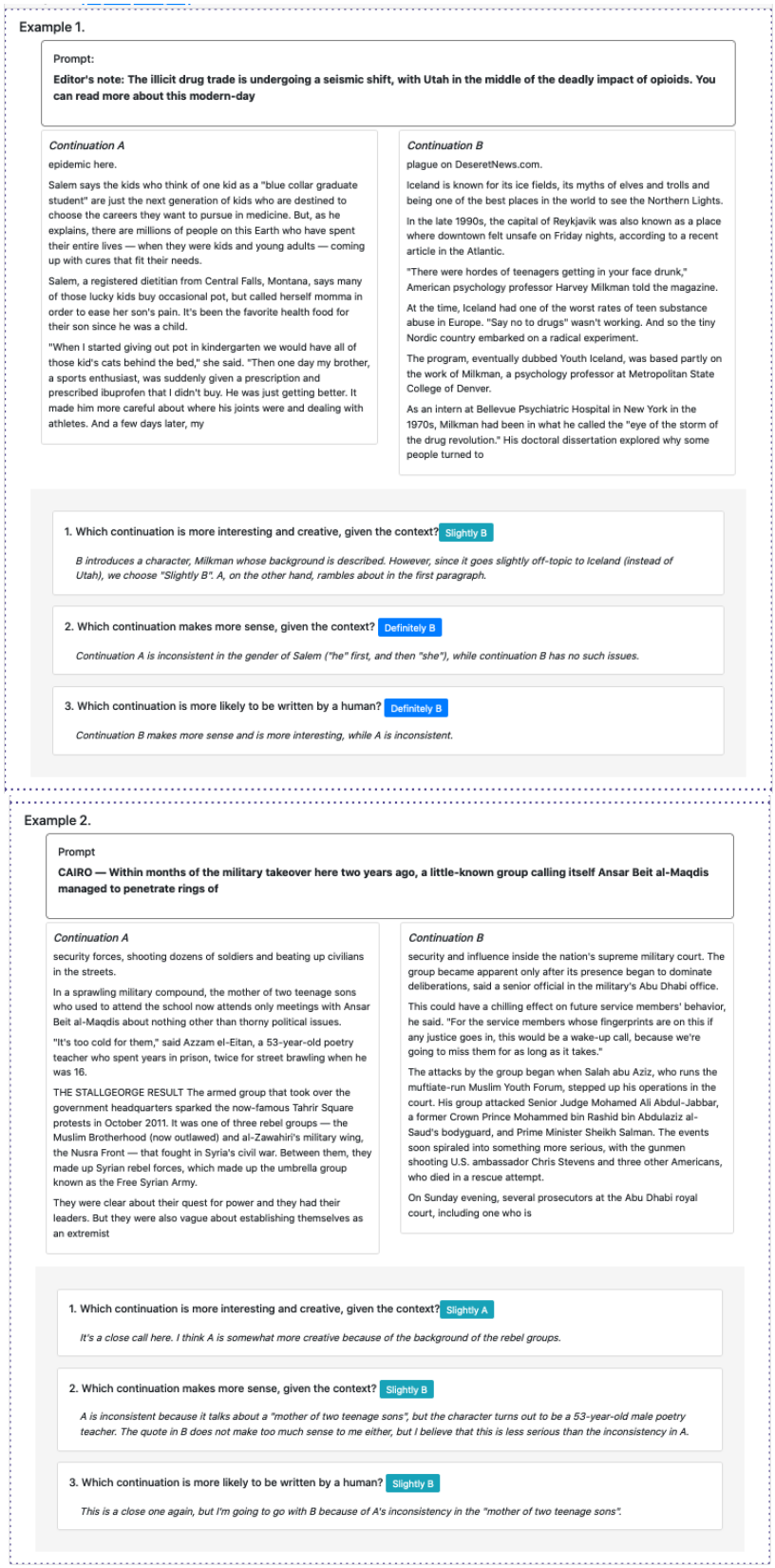


Figure B.5: Annotated examples shown to the evaluators.

Appendix C

RFA: Experimental Details

In this chapter, we we give complete experiments details in Section C.1 and additional numerical results in Section C.2. The numerical study of the Weiszfeld algorithm is given in Section C.3.

C.1 Experimental Details

The outline of this section is:

- Section C.1.1 gives the dataset and task description.
- Section C.1.2 discusses the hyperparameter tuning.
- Section C.1.3 describes the evaluation methodology.

C.1.1 Datasets and Task Description

We experiment with three tasks, (1) handwritten-letter recognition, (2) character-level language modeling, and, (3) sentiment analysis. As discussed in Section 5.1, we take the weight $\alpha_i \propto N_i$, which is the number of data points available on device i.

Handwritten-Letter Recognition

The first dataset is the EMNIST dataset (Cohen et al., 2017) for handwritten letter recognition.

Data. Each inpt x is a gray-scale image resized to 28×28. Each output y is categorical variable which takes 62 different values, one per class of letter (0-9, a-z, A-Z).

Formulation. The task of handwritten letter recognition is cast as a multi-class classification problem with 62 classes.

Distribution of Data. The handwritten characters in the images are annotated by the writer of the character as well. We use a non-i.i.d. split of the data grouped by a writer of a given image. We discard devices with less than 100 total input-output pairs (both train and test), leaving a total of 3461 devices. Of these, we sample 1000 devices to use for our simulations, corresponding to about 30% of the data. This selection held constant throughout the simulations. The number of training examples across these devices

summarized in the following statistics: median 160, mean 202, standard deviation 77, maximum 418 and minimum 92. This preprocessing was performed using LEAF (Caldas et al., 2018).

Models. For the model φ, we consider two options: a linear model and a convolutional neural network.

- Linear Model: The linear model maintains parameters $w_1, \cdots, w_{62} \in \mathbb{R}^{28\times 28}$. For a given image x, class l is assigned score $\langle w_l, x\rangle$, which is then converted to a probability using a softmax operation as $p_l = \exp(\langle w_l, x\rangle)/\sum_{l'} \exp(\langle w_{l'}, x\rangle)$. For a new input image x, the prediction is made as $\arg\max_l \langle w_l, x\rangle$.
- Convolutional Neural Network (ConvNet): The ConvNet (LeCun et al., 1998) we consider contains two convolutional layers with max-pooling, followed by a fully connected hidden layer, and another fully connected (F.C.) layer with 62 outputs. When given an input image x, the output of this network is assigned as the scores of each of the classes. Probabilities are assigned similar to the linear model with a softmax operation on the scores. The schema of network is given below:

$$\begin{array}{c}\text{Input} \\ 28\times 28\end{array} \longrightarrow \begin{array}{c}\text{Conv2D} \\ \text{filters} = 32 \\ \text{kernel} = 5\times 5\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool} \\ \text{kernel } = 2\times 2 \\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{Conv2D} \\ \text{filters} = 64 \\ \text{kernel} = 5\times 5\end{array} \longrightarrow$$

$$\text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool} \\ \text{kernel } = 2\times 2 \\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{F.C.} \\ \text{units} = 2048\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{F.C.} \\ \text{units} = 62\end{array} \longrightarrow \text{score}$$

Loss Function. We use the multinomial logistic loss $\ell(y, p) = -\log p_y$, for probabilities $p = (p_1, \cdots, p_{62})$ and $y \in \{1, \cdots, 62\}$. In the linear model case, it is equivalent to the classical softmax regression.

Evaluation Metric. The model is evaluated based on the classification accuracy on the test set.

Character-Level Language Modeling

We learn a character-level language model over the Complete Works of Shakespeare (Shakespeare). The goal is to read a few characters and predict the next character which appears.

Data. The dataset consists of text from the Complete Works of William Shakespeare as raw text.

Formulation. We formulate the task as a multi-class classification problem with 53 classes (a-z, A-Z, other) as follows. At each point, we consider the previous $H = 20$ characters, and build $x \in \{0, 1\}^{H\times 53}$ as a one-hot encoding of these H characters. The goal is then try to predict the next character, which can belong to 53 classes. In this manner, a text with l total characters gives l input-output pairs.

Distribution of Data. We use a non-i.i.d. split of the data. Each role in a given play (e.g., Brutus from The Tragedy of Julius Caesar) is assigned as a separate device. All devices with less than 100 total examples are discarded, leaving 628 devices. The training set is assigned a random 90% of the input-output pairs, and the other rest are held out for testing. This distribution of training examples is extremely skewed, with the following statistics: median 1170, mean 3579, standard deviation 6367, maximum 70600 and minimum 90. This preprocessing was performed using LEAF (Caldas et al., 2018).

Models. We use a long-short term memory model (LSTM) (Hochreiter and Schmidhuber, 1997) with 128 hidden units for this purpose. This is followed by a fully connected layer with 53 outputs, the output of which is used as the score for each character. As previously, probabilities are obtained using the softmax operation.

Loss Function. We use the multinomial logistic loss.

Evaluation Metric. The model is evaluated based on the accuracy of next-character prediction on the test set.

Sentiment Analysis

The third task is analyze the sentiment of tweets as positive or negative.

Data. Sent140 (Go et al., 2009) is a text dataset of 1,600,498 tweets produced by 660,120 Twitter accounts. Each tweet is represented by a character string with emojis redacted. Each tweet is labeled with a binary sentiment reaction (i.e., positive or negative), which is inferred based on the emojis in the original tweet.

Formulation. The task is a binary classification problem, with the output being a positive or negative sentiment, while the input is the raw text of the tweet.

Distribution of Data. We use a non-i.i.d. split of the data. Each client device represents a Twitter user and contains tweets from this user. We discarded all clients containing less that 50 tweets, leaving only 877 clients. The training set is assigned a random 80% of the input-output pairs, and the other rest are held out for testing. This distribution of training examples across client devices is skewed, with the following statistics: median 55, mean 65.3, standard deviation 32.4, maximum 439 and minimum 40. This preprocessing was performed using LEAF (Caldas et al., 2018).

Models. We use a linear model $\varphi(x; w) = w^\top \phi(x)$, where the feature representation $\phi(x) \in \mathbb{R}^{50}$ of text x is obtained as the average of the GloVe embeddings (Pennington et al., 2014) $G(\cdot)$ of each word in the tweet, i.e.,

$$\phi(x) = \frac{1}{|x|} \sum_{i=1}^{|x|} G(x_i) \, .$$

Loss Function. We use the binary logistic loss.

Evaluation Metric. We use the binary classification accuracy.

C.1.2 Methods, Hyperparameters and Variants

We first describe the corruption model, followed by various methods tested.



Figure C.1: Illustration of the data corruption introduced in the Shakespeare dataset. The first line denotes the original text. The second line shows the effective x when predicting the "m" of the word "median". The second line shows the corresponding $\widetilde{x}$ after the introduction of the corruption. Note that $\widetilde{x}$ is the string "edian's ro" reversed.

Corruption Model

Since the goal of this work to test the robustness of federated learning models in the setting of high corruption, we artificially corrupt updates while controlling the level of corruption. We use the following corruption models.

Data Corruption. This is an example of static data poisoning. The model training procedure is not modified, but the data fed into the model is modified. In particular, we take a modification $\tilde{D}_i$ of the local dataset D_i of client i and run the training algorithm on this different dataset. The exact nature of the modification depends on the dataset:

- EMNIST: We take the negative of the image x. Mathematically, $\tilde{D}_i(x, y) = D_i(1 - x, y)$, assuming the pixels of x are normalized to lie in $[0, 1]$. The labels are left unmodified.
- Shakespeare: We reverse the original text. Mathematically, $\tilde{D}_i(c_1 \cdots c_{20}, c_{21}) = D_i(c_{21} \cdots c_2, c_1)$ This is illustrated in Fig. C.1. The labels are left unmodified.
- Sent140: We flip the label, i.e., $\tilde{D}_i(x, y) = D_i(x, -y)$. The text in the tweet remains unchanged.

Gaussian corruption. This is an example of update poisoning. The data is not modified here but the update of a client device is directly replaced by a Gaussian random variable, with standard deviation σ equal to the standard deviation of the original update across its components. Note that we corrupt the *update* to the model parameters transmitted by the device, which is typically much smaller in norm than the model parameters themselves.

Omniscient corruption. This is an example of update poisoning. The data is not modified here but the parameters of a device are directly modified. In particular, $w_i^{(t+1)}$ for $i \in \mathcal{C}$ is set to be

$$w_i^{(t+1)} = -\frac{1}{\sum_{j \in S_t \cap \mathcal{C}} \alpha_j} \left(2 \sum_{j \in S_t \setminus \mathcal{C}} \alpha_j w_{j,\tau}^{(t)} + \sum_{j \in S_t \cap \mathcal{C}} \alpha_j w_{j,\tau}^{(t)} \right),$$

such that

$$\sum_{i \in S_t} \alpha_i w_i^{(t+1)} = -\sum_{i \in S_t} \alpha_i w_{i,\tau}^{(t)}.$$

In other words, the weighted arithmetic mean of the model parameter is set to be the negative of what it would have other been without the corruption. This corruption model requires full knowledge of the data and server state, and is adversarial in nature.

Implementation details. Given a corruption level ρ, the set of devices which return corrupted updates are selected as follows:
- Start with $\mathcal{C} = \varnothing$.
- Sample device i uniformly without replacement and add to $\mathcal{C}$. Stop when $\sum_{i \in C} \alpha_i$ just exceeds ρ.

Methods

We compare the following algorithms:
- the FedAvg algorithm (McMahan et al., 2017),
- the RFA algorithm proposed here in Algorithm 5.1,
- the minibatch stochastic gradient descent (SGD) algorithm.

Hyperparameters

The hyperparameters for each of these algorithms are detailed below.

FedAvg. The FedAvg algorithm requires the following hyperparameters.
- Devices per round m: We use 100 for EMNIST and 50 for both the Shakespeare and Sent140 datasets.
- Batch Size and Number of Local Epochs: Instead of running τ local updates, we run for n_e local epochs following (McMahan et al., 2017) with a batch size of b. For the EMNIST dataset, we use $b = 50, n_e = 5$, and for Shakespeare and Sent140, we use $b = 10, n_e = 1$.
- Learning Rate (γ_t): We use a learning a learning rate scheme $\gamma_t = \gamma_0 C^{\lfloor t/t_0 \rfloor}$, where γ_0 and C were tuned using grid search on validation set (20% held out from the training set) for a fixed time horizon on the uncorrupted data. The values which gave the highest validation accuracy were used *for all settings* - both corrupted and uncorrupted. The time horizon used was 2000 iterations for the EMNIST linear model, 1000 iterations for the EMNIST ConvNet 200 iterations for Shakespeare LSTM.
- Initial Iterate $w^{(0)}$: Each element of $w^{(0)}$ is initialized to a uniform random variable whose range is determined according to TensorFlow's "glorot_uniform_initializer".

RFA. RFA's hyperparameters, in addition to those of FedAvg, are:
- Algorithm: We use the smoothed Weiszfeld algorithm, as discussed in Sec. 5.3.
- Smoothing parameter ν: Based on the interpretation that ν guards against division by small numbers, we simply use $\nu = 10^{-6}$ throughout.
- Robust Aggregation Stopping Criterion: This concerns the stopping criterion used to terminate the smoothed Weiszfeld algorithm. We use two criteria: an iteration budget and a relative improvement condition - we terminate if a given iteration budget has been extinguished, or if the relative improvement in objective value $|g_\nu(v^{(r)}) - g_\nu(v^{(r+1)})|/g_\nu(v^{(r)}) \leq 10^{-6}$ is small.

C.1.3 Evaluation Methodology and Other Details

We specify here the quantities appearing on the x and y axes on the plots, as well as other details.

x **Axis.** As mentioned in Section 5.1, the goal of federated learning is to learn the model with as few rounds of communication as possible. Therefore, we evaluate various methods against the number of rounds of communication, which we measure via the number of calls to a secure average oracle.

Note that FedAvg and SGD require one call to the secure average oracle per outer iteration, while RFA could require several. Hence, we also evaluate performance against the number of outer iterations.

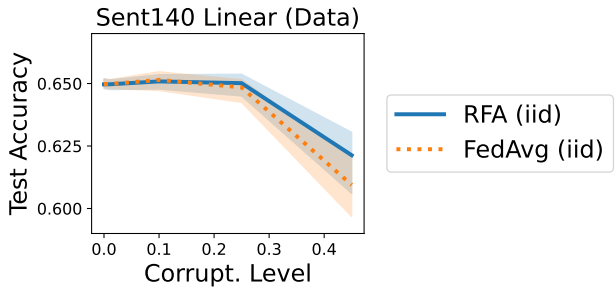


Figure C.2: Robustness of RFA and FedAvg for an i.i.d. data split on Sent140 with data corruption.

y **Axis.** We are primarily interested in the test accuracy, which measures the performance on unseen data. We also plot the function value F, which is the quantity our optimization algorithm aims to minimize. We call this the train loss.

Evaluation with Data Corruption. In simulations with data corruption, while the training is performed on corrupted data, we evaluate train and test progress using the corruption-free data.

Software. We use the package LEAF (Caldas et al., 2018) to simulate the federated learning setting. The models used are implemented in TensorFlow.

Hardware. Each simulation was run in a simulation as a single process. The EMNIST linear model simulations were run on two workstations with 126GB of memory, with one equipped with Intel i9 processor running at 2.80GHz, and the other with Intel Xeon processors running at 2.40GHz. Simulations involving neural networks were run either on a 1080Ti or a Titan Xp GPU.

Random runs. Each simulation is repeated 5 times with different random seeds, and the solid lines in the plots here represents the mean over these runs, while the shaded areas show the maximum and minimum values obtained in these runs.

C.2 Additional Numerical Results

Effect of non-identical data distributions. Here, we plot the analogue of Figure 5.3 for the Sent140 dataset with data corruption in the setting where the dataset was split in an i.i.d. manner across devices. Recall that we had a small gap of 0.3% between the performance of RFA and FedAvg in the setting of no corruption. Consistent with the theory, this gap completely vanishes in the i.i.d. case, as shown in Figure C.2.

Effect of iteration budget of smoothed Weiszfeld. We study the effect of the iteration budget of the smoothed Weiszfeld algorithm in RFA. in Figure C.3. We observe that a low communication budget is faster in the regime of low corruption, while more iterations work better in the high corruption regime. We used a budget of 3 calls to the secure average oracle throughout to trade-off between these two scenarios.

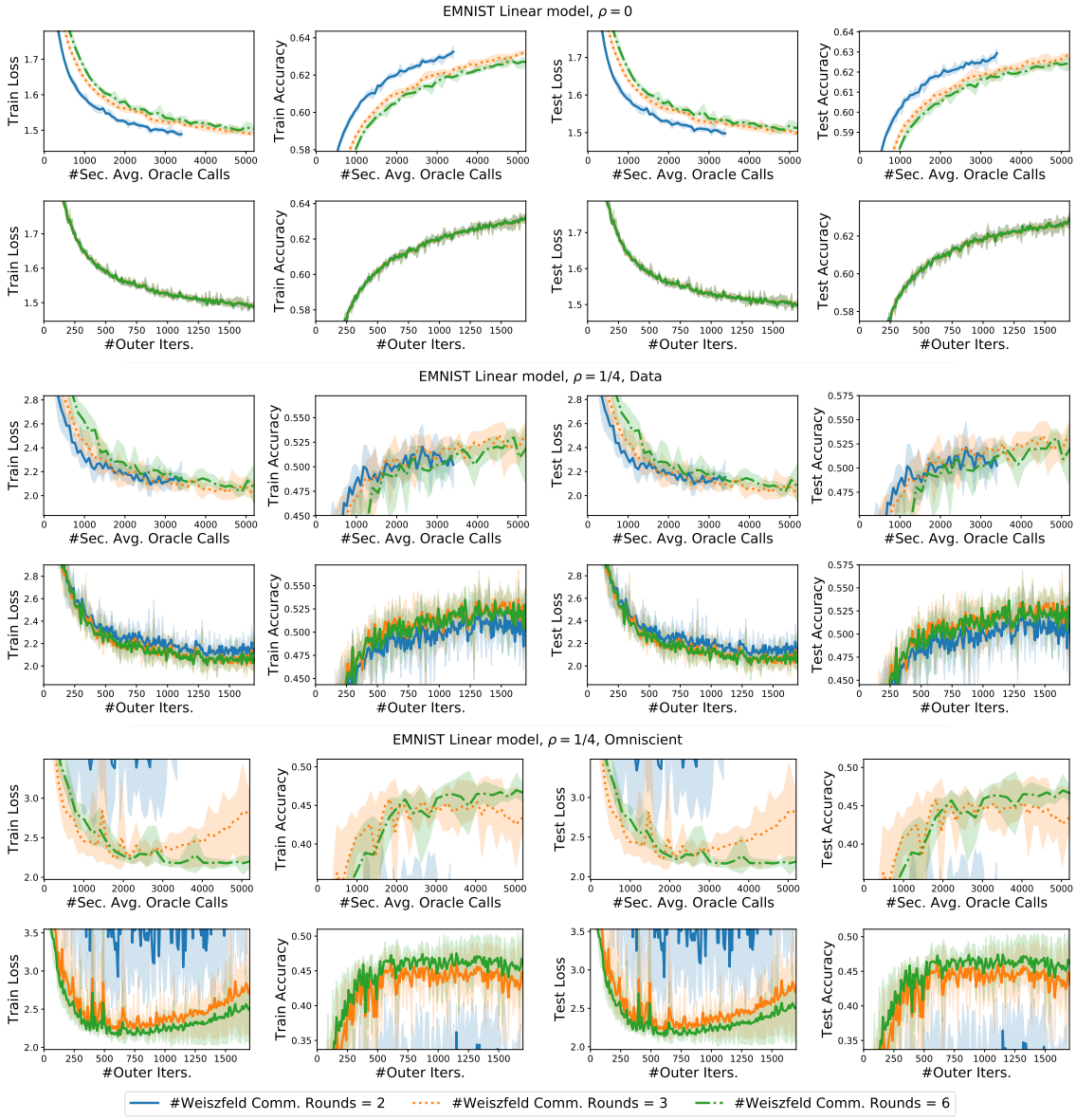


Figure C.3: Hyperparameter study, effect of the maximum number of the communication budget on the smoothed Weiszfeld algorithm in RFA on the EMNIST dataset with a linear model.

Effect of number of devices per iteration round. Figure C.4 plots the performance of RFA against the number m of devices chosen per round. We observe the following: in the regime of low corruption, good performance is achieved by selecting 50 devices per round (5%), where as 10 devices per round (1%) is not enough. On the other hand, in high corruption regimes, we see the benefit of choosing more devices per round, as a few runs with 10 or 50 devices per round with omniscient corruption at 25% diverged. This is consistent with Theorem 5.9, which requires the number of devices per round to increase with the level of corruption (cf. Eq. (5.17)).

Effect of local computation. Figure C.5 plots the performance of FedAvg and RFA versus the amount

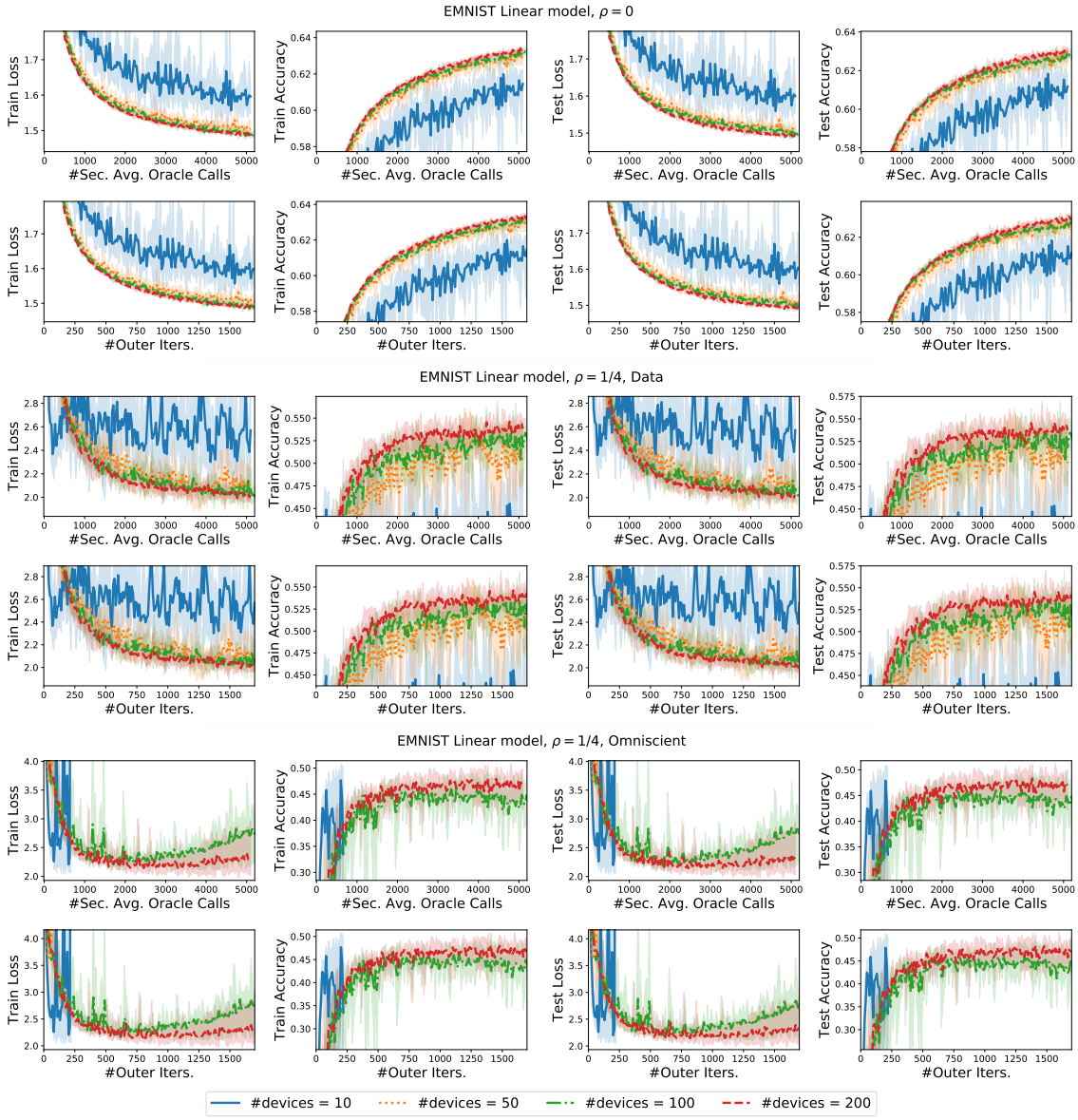
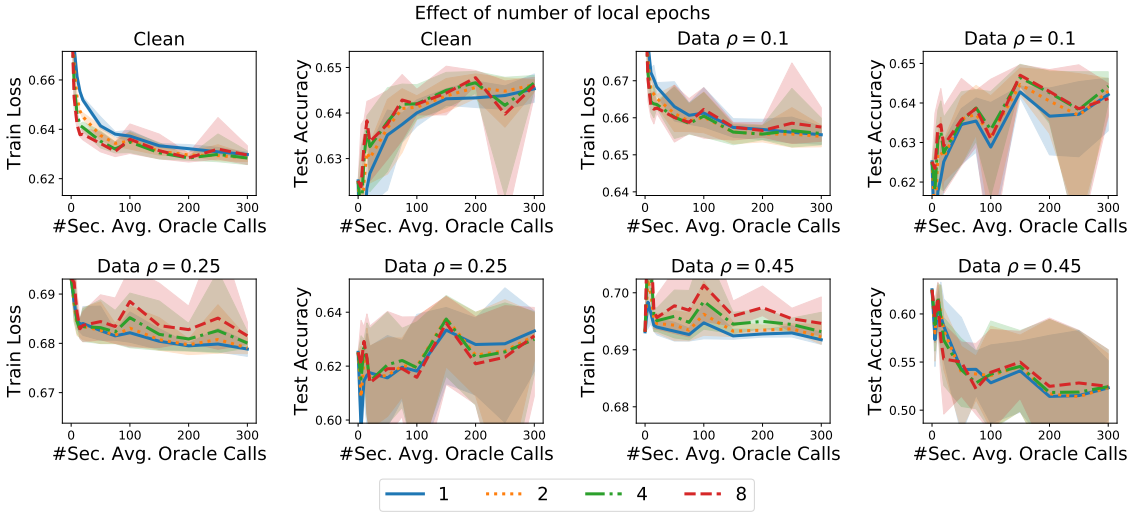
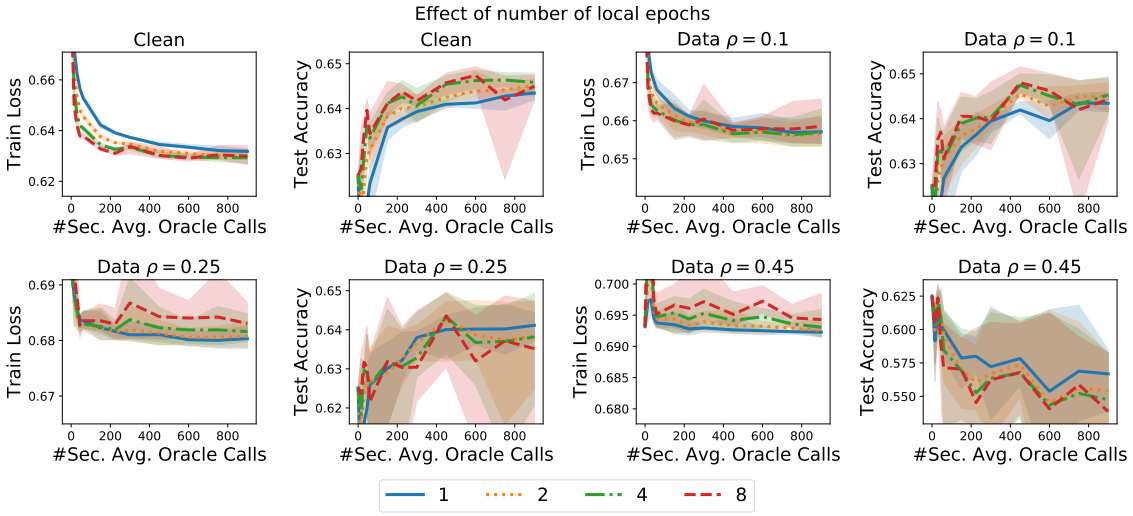


Figure C.4: Hyperparameter study, effect of the number of selected client devices per round in RFA on the EMNIST dataset with a linear model.

of local computation. We see that the performance is always within one standard deviation of each other irrespective of the amount of local computation. However, we also note that RFA with a single local epoch is obtains a slightly lower test accuracy in the no-corruption regime than using more local computation.



(a) FedAvg.



(b) RFA.

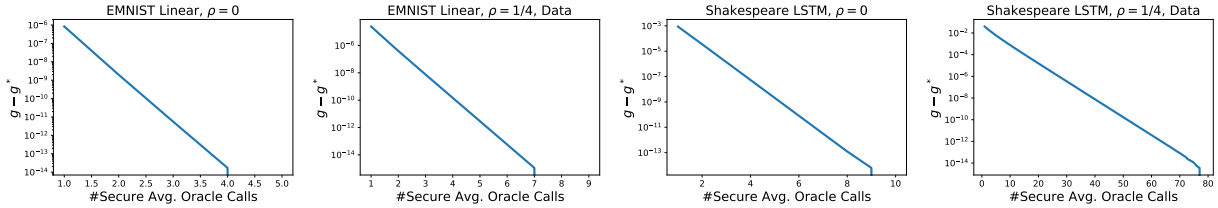
Figure C.5: Effect of the number of epochs on FedAvg and RFA for the Sent140 dataset in the presence of data corruption.

C.3 Numerical Results: Convergence of The Smoothed Weiszfeld Algorithm

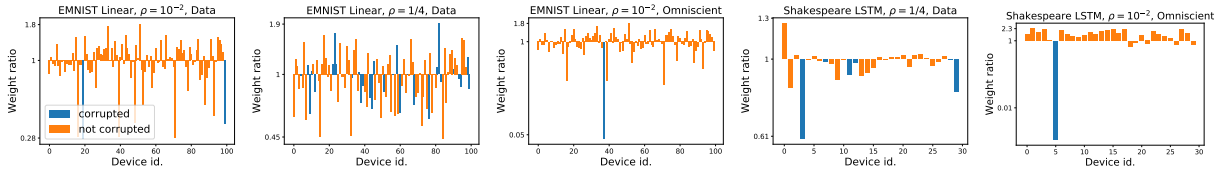
For each of these models, we freeze FedAvg at a certain iteration and experiment with different robust aggregation algorithms.

We find that the smoothed Weiszfeld algorithm enjoys a fast convergence behavior, converging exactly to the smoothed geometric median in a few passes. In fact, the smoothed Weiszfeld algorithm displays (local) linear convergence, as evidenced by the straight line in log scale. Further, we also maintain a strict iteration budget of 3 iterations. This choice is also justified in hindsight by the results of Figure C.3.

Next, we visualize the weights assigned by the geometric median to the corrupted updates. Note that



(a) Convergence of the smoothed Weiszfeld algorithm and for robust aggregation.



(b) Visualization of the re-weighting of points in the robust aggregate.

Figure C.6: Performance of robust aggregation algorithms.

the smoothed geometric median $w_1, \cdots, w_m$ is some convex combination $\sum_{i=1}^{m} \beta_i w_i$. This weight β_i of w_i is a measure of the influence of w_i on the aggregate. We plot in Figure C.6b the ratio β_i/α_i for each device i, where α_i is its weight in the arithmetic mean and β_i is obtained by running the smoothed Weiszfeld algorithm to convergence. We expect this ratio to be smaller for worse corruptions and ideally zero for obvious corruptions. We find that the smoothed geometric median does indeed assign lower weights to the corruptions, while only accessing the points via a secure average oracle.

Appendix D

$\triangle$-FL: Proofs and Experimental Details

In this chapter, we give the full proofs from Chapter 6 and provide additional experimental details and results. In particular, we give the full proofs of convergence in Section D.1, followed by the privacy analysis in Section D.2. Finally, we turn to the experiments in Section D.3.

D.1 Convergence Analysis

Below, we restate and prove Theorem 6.5 as Theorem D.1 in Section D.1.1 and Theorem 6.6 as Theorem D.2 in Section D.1.2,

D.1.1 Convergence Analysis: Non-convex Case

We review some definitions of subdifferentials and weak convexity before we get to the main theorem.

Nonconvex Subdifferentials. We start by recalling the definition of subgradients for nonsmooth functions (in finite dimension), following the terminology of (Rockafellar and Wets, 2009). Consider a function $\psi\colon \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ and a point $\bar{w}$ such that $\psi(\bar{w}) < +\infty$. The regular (or Fréchet) subdifferential of ψ at $\bar{w}$ is defined by

$$\partial\psi(\bar{w}) = \left\{ s \in \mathbb{R}^d : \ \psi(w) \geq \psi(\bar{w}) + \langle s, w - \bar{w}) \rangle + o(\|w - \bar{w}\|) \right\}.$$

The regular subdifferential thus corresponds to the set of gradients of smooth functions that are below ψ and coincide with it at $\bar{w}$. These notions generalize (sub)gradients of both smooth functions and convex functions: it reduces to the singleton $\{\nabla\psi(\bar{w})\}$ when ψ is smooth and to the standard subdifferential from convex analysis when ψ is convex.

Weak Convexity. We recall the notion of weak convexity, which is one way of characterizing functions which are "close" to convex. A function $\psi : \mathbb{R}^d \to \mathbb{R}$ is said to be η-weakly convex if the function $w \mapsto \psi(w) + (\eta/2)\|w\|^2$ is convex (Nurminskii, 1973). The class of weakly convex functions includes all convex functions (with $\eta = 0$) and all L-smooth functions (with $\eta = L$).

Weak convexity also admits an equivalent first-order condition: for any $w, v \in \mathbb{R}^d$ and $s \in \partial\psi(w)$, we have,

$$\psi(z) \geq \psi(w) + \langle s, v - w \rangle - \frac{\eta}{2}\|v - w\|^2 . \tag{D.1}$$

Weak convexity will feature in our developments in two ways:

- In our case, both F_θ as well as $F_{\theta,S}$ are L-weakly convex, since each can be written as the maximum of a family of L-smooth functions (Drusvyatskiy and Paquette, 2019, Lemma 4.2).
- The prox operator for weakly convex function is well-defined. Let ψ be a η-weakly convex function. Its proximal or prox operator, with parameter $\mu > 0$ is defined as

$$\operatorname{prox}_{\psi/\mu}(w) = \arg\min_v \left\{ \psi(v) + \frac{\mu}{2}\|v - w\|^2 \right\} .$$

  It is well-defined (i.e., the argmin exists and is unique) for $\mu > \eta$, since the function inside the argmin is $(\mu - \eta)$-strongly convex.

In nonsmooth and nonconvex optimization of weakly convex functions, we are interested in finding stationary points w.r.t. the regular subdifferential, i.e., points w satisfying $0 \in \partial\psi(w)$. A natural measure of near-stationarity is, therefore,

$$\operatorname{dist}(0, \partial\psi(w)) = \inf_{s \in \partial\psi(w)} \|s\| .$$

Moreau Envelope. Given a parameter $\mu > 0$, we define the Moreau envelope of $\overline{F}_\theta$ as

$$\overline{\Phi}^\mu_\theta(w) = \inf_v \left\{ \overline{F}_\theta(v) + \frac{\mu}{2}\|v - w\|^2 \right\} .$$

The Moreau envelope is well-defined since $\overline{F}_\theta$ is bounded from below by our assumptions. We will use two standard properties of the Moreau envelope:

- Since $\overline{F}_{\theta,S}$ is L-weakly convex, we have that its Moreau envelope $\overline{\Phi}^\mu_\theta(w)$ is continuously differentiable for $\mu > L$ with

$$\nabla\overline{\Phi}^\mu_\theta(w) = \mu \left(w - \operatorname{prox}_{\overline{F}_\theta/\mu}(w) \right) .$$

- The stationary points of $\overline{\Phi}^\mu_\theta$ and $\overline{F}_\theta$ coincide and $\inf \overline{\Phi}^\mu_\theta = \inf \overline{F}_\theta$ for $\mu > L$.
- We have for all $\mu > 0$ that $\overline{\Phi}^\mu_\theta(w) \leq \overline{F}_\theta(w)$.

Notation. Let $S = S^{(t)}$ denote the random set of clients selected in round t of Algorithm 6.3. We define

$$\tilde{\nabla} F_{\theta,S}(w^{(t)}) = \sum_{i \in S} \pi_i^{(t)} \nabla F_i(w^{(t)}) , \tag{D.2}$$

where $\pi_i^{(t)} \in \arg\max_{\pi \in \mathcal{P}_{\theta,S}} \sum_{i \in S} \pi_i F_i(w^{(t)})$ is selected as in line 3 of Algorithm 6.3. A key consequence of the chain rule (Rockafellar and Wets, 2009, Thm. 10.6) is

$$\tilde{\nabla} F_{\theta,S}(w^{(t)}) \in \partial F_{\theta,S}(w^{(t)}) . \tag{D.3}$$

Convergence Analysis. We now state and prove the convergence result in the nonconvex case.

Theorem D.1. *Fix and the number of rounds T, fix $\mu = 2L$ and set the learning rate*

$$\gamma = \min \left\{ \frac{1}{4\tau L}, \frac{1}{\tau\sqrt{T}} \sqrt{\frac{\Delta F_0}{LG^2}}, \frac{1}{\tau T^{1/3}} \left(\frac{\Delta F_0}{32L^2G^2(1 - \tau^{-1})} \right)^{1/3} \right\} ,$$

where we denote $\Delta F_0 = \overline{\Phi}_\theta^\mu(w^{(0)}) - \inf \overline{\Phi}_\theta^\mu \leq \overline{F}_\theta(w^{(0)}) - \inf \overline{F}_\theta$. *Let* $\hat{w}$ *be sampled uniformly at random from* $\{w^{(0)}, \cdots, w^{(T-1)}\}$. *Ignoring absolute constants, we have the bound,*

$$\mathbb{E}\left\|\nabla \overline{\Phi}_\theta^\mu(\hat{w})\right\|^2 \leq \sqrt{\frac{\Delta F_0 L G^2}{T}} + \left(\frac{\Delta F_0 L G (1-\tau^{-1})^{1/2}}{T}\right)^{2/3} + \frac{\Delta F_0 L}{T}\,.$$

Proof. We start with some notation. Throughout, we denote $v^{(t)}$ as the proximal point of $w^{(t)}$:

$$v^{(t)} = \operatorname{prox}_{\overline{F}_\theta/\mu}(w^{(t)}) = \arg\min_v \left\{\overline{F}_\theta(z) + \frac{\mu}{2}\left\|v - w^{(t)}\right\|^2\right\}\,.$$

Let $\mathcal{F}^{(t)}$ denote the sigma algebra generated by $w^{(t)}$ and define $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot \mid \mathcal{F}^{(t)}]$. By definition, we have that $v^{(t)}$ is also $\mathcal{F}^{(t)}$-measurable.

We use the update $w^{(t+1)} = w^{(t)} - \gamma \sum_{i\in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla F_i(w_{i,k}^{(t)})$ to get

$$\begin{aligned}
\overline{\Phi}_\theta^\mu(w^{(t+1)}) &= \min_v \left\{\overline{F}_\theta(v) + \frac{\mu}{2}\left\|v - w^{(t+1)}\right\|^2\right\} \\
&\leq \overline{F}_\theta(v^{(t)}) + \frac{\mu}{2}\left\|v^{(t)} - w^{(t+1)}\right\|^2 \\
&= \overline{F}_\theta(v^{(t)}) + \frac{\mu}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + \mu\gamma\left\langle v^{(t)} - w^{(t)}, \sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla F_i(w_{i,k}^{(t)})\right\rangle \\
&\qquad + \frac{\mu\gamma^2}{2}\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla F_i(w_{i,k}^{(t)})\right\|^2 \\
&= \overline{\Phi}_\theta^\mu(w^{(t)}) + \underbrace{\mu\gamma\left\langle v^{(t)} - w^{(t)}, \sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla F_i(w_{i,k}^{(t)})\right\rangle}_{=:\mathcal{T}_1} \\
&\qquad + \underbrace{\frac{\mu\gamma^2}{2}\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla F_i(w_{i,k}^{(t)})\right\|^2}_{=:\mathcal{T}_2}\,.
\end{aligned} \tag{D.4}$$

We handle both $\mathcal{T}_1$ and $\mathcal{T}_2$ separately. We start with $\mathcal{T}_1$ by defining

$$C_k := \left\langle v^{(t)} - w^{(t)}, \sum_{i\in S}\pi_i^{(t)}\left(\nabla F_i(w_{i,k}^{(t)}) - \nabla F_i(w^{(t)})\right)\right\rangle.$$

We use the weak convexity of $F_{\theta,S}$, in particular (D.1), to bound

$$\begin{aligned}
\left\langle v^{(t)} - w^{(t)}, \sum_{i\in S}\pi_i^{(t)}\nabla F_i(w_{i,k}^{(t)})\right\rangle &= \left\langle v^{(t)} - w^{(t)}, \sum_{i\in S}\pi_i^{(t)}\nabla F_i(w^{(t)})\right\rangle + C_k \\
&\overset{\text{(D.2)}}{=} \left\langle v^{(t)} - w^{(t)}, \tilde{\nabla} F_{\theta,S}(w^{(t)})\right\rangle + C_k \\
&\overset{\text{(D.1),(D.3)}}{\leq} F_{\theta,S}(v^{(t)}) - F_{\theta,S}(w^{(t)}) + \frac{L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + C_k\,.
\end{aligned}$$

Taking an expectation conditioned on $\mathcal{F}^{(t)}$ and noting that $v^{(t)}$ is $\mathcal{F}^{(t)}$-measurable (so the expectation is only over the randomness in S), we get

$$\begin{aligned}
&\mathbb{E}_t\left\langle v^{(t)} - w^{(t)}, \sum_{i\in S} \pi_i^{(t)} \nabla F_i(w_{i,k}^{(t)}) \right\rangle \\
&\quad \leq \left(\overline{F}_\theta(v^{(t)}) + \frac{\mu}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 \right) - \overline{F}_\theta(w^{(t)}) - \frac{\mu - L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + \mathbb{E}_t[C_k]\,.
\end{aligned}$$

Note that the function

$$h(v) := \overline{F}_\theta(v) + \frac{\mu}{2}\left\|v - w^{(t)}\right\|^2$$

is $(\mu - L)$-strongly convex and $v^{(t)}$ is its minimizer. This gives,

$$h(w^{(t)}) - h(v^{(t)}) \geq \frac{\mu - L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2.$$

This implies,

$$\mathbb{E}_t\left\langle v^{(t)} - w^{(t)}, \sum_{i\in S} \pi_i^{(t)} \nabla F_i(w_{i,k}^{(t)}) \right\rangle \leq -(\mu - L)\left\|v^{(t)} - w^{(t)}\right\|^2 + \mathbb{E}_t[C_k]\,.$$

Next, we bound the term C_k as follows:

$$\begin{aligned}
|C_k| &= \left| \left\langle v^{(t)} - w^{(t)}, \sum_{i\in S} \pi_i^{(t)} \left(\nabla F_i(w_{i,k}^{(t)}) - \nabla F_i(w^{(t)}) \right) \right\rangle \right| \\
&\leq \frac{\mu - L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + \frac{1}{2(\mu - L)}\left\| \sum_{i\in S} \pi_i^{(t)} \left(\nabla F_i(w_{i,k}^{(t)}) - \nabla F_i(w^{(t)}) \right) \right\|^2 \\
&\leq \frac{\mu - L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + \frac{1}{2(\mu - L)} \sum_{i\in S} \pi_i^{(t)} \left\| \left(\nabla F_i(w_{i,k}^{(t)}) - \nabla F_i(w^{(t)}) \right) \right\|^2 \\
&\leq \frac{\mu - L}{2}\left\|v^{(t)} - w^{(t)}\right\|^2 + \frac{L^2}{2(\mu - L)} \sum_{i\in S} \pi_i^{(t)} \left\| w_{i,k}^{(t)} - w^{(t)} \right\|^2.
\end{aligned}$$

Together with the previous inequality and the equality $\nabla \overline{\Phi}_\theta^\mu(w^{(t)}) = \mu(w^{(t)} - v^{(t)})$, we get a bound on $\mathcal{T}_1$ as

$$\mathbb{E}_t[\mathcal{T}_1] \leq -\frac{\gamma\tau(\mu - L)}{2\mu}\left\| \nabla \overline{\Phi}_\theta^\mu(w^{(t)}) \right\|^2 + \frac{\mu\gamma L^2}{2(\mu - L)} d^{(t)}\,, \tag{D.5}$$

where $d^{(t)} = \sum_{i\in S} \sum_{k=0}^{\tau-1} \pi_i^{(t)} \left\| w_{i,k}^{(t)} - w^{(t)} \right\|^2$ is the client drift. Next, we bound $\mathcal{T}_2$ as

$$\begin{aligned}
\mathcal{T}_2 &= \frac{\mu\gamma^2}{2}\left\| \sum_{i\in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla F_i(w_{i,k}^{(t)}) \right\|^2 \leq \frac{\mu\gamma^2\tau}{2} \sum_{i\in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \left\| \nabla F_i(w_{i,k}^{(t)}) \right\|^2 \\
&\leq \frac{\mu\gamma^2\tau^2 G^2}{2}\,,
\end{aligned} \tag{D.6}$$

where we used Jensen's inequality and $\left\|\nabla F_i(w_{i,k}^{(t)})\right\|^2 \leq G^2$ since F_i is G-Lipschitz.

Next, we plug in (D.5) and (D.6) into (D.4) and invoke Proposition D.7 to bound the client drift $d^{(t)}$ to get

$$\mathbb{E}_t\left[\overline{\Phi}_\theta^\mu(w^{(t+1)})\right] \leq \overline{\Phi}_\theta^\mu(w^{(t)}) - \frac{\gamma\tau(\mu-L)}{2}\left\|\nabla\overline{\Phi}_\theta^\mu(w^{(t)})\right\|^2 + \frac{\mu\gamma^2\tau^2G^2}{2}\left(1+\frac{8L^2\gamma}{\mu-L}(\tau-1)\right).$$

Finally, taking an unconditional expectation, summing this up over $t=0$ to $T-1$ and rearranging gives us the bound

$$\mathbb{E}\left\|\nabla\overline{\Phi}_\theta^\mu(\hat{w})\right\|^2 \leq \frac{2\Delta F_0}{\gamma\tau T} + 2\gamma\tau LG^2\left(1+8L\gamma(\tau-1)\right),$$

where we plugged in $\mu = 2L$. Plugging in the choice of γ (cf. Lemma D.10) completes the proof. □

D.1.2 Convergence Analysis: Strongly Convex Case

The main result is the following.

Theorem D.2 (Convergence rate, Strongly Convex Case)**.** *Suppose that each F_i is convex and the regularization parameter satisfies $0 < \lambda < L$. Define notation $\kappa = (L+\lambda)/\lambda$, $w^\star = \arg\min_w F_\theta(w)$ and $\Delta_0 = \|w^{(0)} - w^\star\|^2$. Assume also that the number of rounds is $T \geq \sqrt{2\kappa^3}$. Fix a smoothing parameter*

$$\nu = \begin{cases} \frac{G^2}{\lambda\kappa}, & \text{if } T \leq \sqrt{2\kappa^3}\log\left(1\vee\frac{CT^2}{\kappa^2}\right), \\ \frac{2G^2\kappa^2}{\lambda T^2}\log\left(1\vee\frac{CT^2}{\kappa^2}\right), & \text{if } \sqrt{2\kappa^3}\log\left(1\vee\frac{CT^2}{\kappa^2}\right) \leq T \leq \kappa^2\log\left(1\vee\frac{CT^2}{\kappa^2}\right), \\ \frac{2G^2}{\lambda T}\log\left(1\vee CT\right), & \text{else,} \end{cases}$$

where $C = \lambda^2\Delta_0/(2G^2\log m)$, and a learning rate

$$\gamma = \min\left\{\frac{\sqrt{\lambda}}{18\tau(L+\lambda)\sqrt{L'}}, \frac{1}{4\tau L'}, \frac{1}{\lambda\tau T}\log\left(1\vee\frac{\lambda^2\Delta_0\theta m}{G^2}T\right), \frac{1}{\lambda\tau T}\log^2\left(1\vee\frac{\lambda^2\Delta_0T^2}{G^2\kappa^2(1-\tau^{-1})}T\right)\right\},$$

where $L' = L+\lambda+G^2/\nu$. Consider the sequence $(w^{(t)})_{t=0}^T$ produced by Algorithm 6.3 run with smoothing parameter ν and learning rate γ, and the corresponding averaged iterate

$$\overline{w}^{(T)} := \frac{\sum_{t=0}^T w^{(t)}\left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+t)}}{\sum_{r=0}^T\left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+r)}}.$$

Then, ignoring absolute constants, we have the bound,

$$\mathbb{E}\left[F_\theta(\overline{w}^{(T)}) - F_\theta(w^\star)\right] \leq \lambda\|w^{(0)}-w^\star\|^2\exp\left(-T/\sqrt{2\kappa^3}\right) + \frac{B}{\sqrt{\theta m}} + \frac{G^2}{\lambda T}\left(\frac{1}{\theta m}+\log m\right)\log\left(1\vee\frac{\lambda^2\Delta_0T}{G^2}\right) + \frac{G^2\kappa^2}{\lambda T^2}\left(1-\tau^{-1}+\log m\right)\log^2\left(1\vee\frac{\lambda^2\Delta_0T^2}{G^2\kappa^2}\right).$$

We give the proof in a sequence of results. We start with some notation.

Notation. Analogous to the smoothing F_θ^ν of F_θ, we define the smoothing of the sample version $F_{\theta,S}$ as

$$F_{\theta,S}^\nu(w) = \max_{\pi \in \mathcal{P}_{\theta,S}} \left\{ \sum_{i \in S} \pi_i F_i(w) - \nu D_S(\pi) \right\} + \frac{\lambda}{2}\|w\|^2 \,, \tag{D.7}$$

From Danskin's theorem (Bertsekas, 1999, Proposition B.25), we get the expression of its gradient as

$$\nabla F_{\theta,S}(w^{(t)}) = \sum_{i \in S} \pi_i^{(t)} \nabla \tilde{F}_i(w^{(t)}) \,, \tag{D.8}$$

where $\pi^{(t)}$ attains the unique argmax in (D.7) (see also (6.13) for the definition).

We define the averaged superquantile as

$$\overline{F}_\theta^\nu(w) = \mathbb{E}_{S \sim U_m}[F_{\theta,S}^\nu(w)] \,, \tag{D.9}$$

where U_m is the uniform distribution over subsets of $[n]$ of size m. Finally, let $\overline{w}^\star = \arg\min_w \overline{F}_\theta^\nu(w)$.

We also define the notion of *client drift* as

$$d^{(t)} := \mathbb{E}_{S \sim U_m} \left[\sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \|w_{i,k}^{(t)} - w^{(t)}\|^2 \,\middle|\, \mathcal{F}_t \right] . \tag{D.10}$$

Effect of One Round. The crux of the proof of Theorem D.2 is the following statement.

Proposition D.3. *Consider the setting of Theorem D.2. Let $(w^{(t)})_{t \geq 0}$ the sequence of global models generated by Algorithm 6.3. For any $t \geq 0$, we have:*

$$\begin{aligned} \overline{F}_\theta^\nu(w^{(t)}) - \overline{F}_\theta^\nu(\overline{w}^\star) \leq \frac{1}{\gamma\tau}\left(1 - \frac{\lambda\gamma\tau}{2}\right)\|w^{(t)} - \overline{w}^\star\|^2 - \frac{1}{\gamma\tau}\|w^{(t+1)} - \overline{w}^\star\|^2 \\ + \frac{16\tau G^2\gamma}{\theta m} + \frac{9\,(L+\lambda)^2}{\tau\lambda} d^{(t)} \,, \end{aligned}$$

where $d^{(t)}$ denotes the client drift, defined in (D.10).

Proof. We denote $\mathbb{E}_t[\cdot] := \mathbb{E}[\cdot \mid \mathcal{F}_t]$. We expand the update $w^{(t+1)} = w^{(t)} - \gamma \sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla \tilde{F}_i(w_{i,k}^{(t)})$ to get

$$\begin{aligned} \mathbb{E}_t\|w^{(t+1)} - \overline{w}^\star\|^2 = \|w^{(t)} - \overline{w}^\star\|^2 \underbrace{- 2\gamma \mathbb{E}_t \left[\sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \left\langle \nabla \tilde{F}_i(w_{i,k}^{(t)}), w^{(t)} - \overline{w}^\star \right\rangle \right]}_{=:A} \\ \underbrace{+ \gamma^2 \mathbb{E}_t \left\| \sum_{i \in S} \pi_i^{(t)} \sum_{k=0}^{\tau-1} \nabla \tilde{F}_i(w_{i,k}^{(t)}) \right\|^2}_{=:B} . \end{aligned}$$

In order to bound A, we use the expression (D.8) of the gradient $\nabla F_{\theta,S}(w^{(t)})$ together with the λ-strong convexity (cf. (D.16)) of $F_{\theta,S}$ to get

$$\begin{aligned}\left\langle \sum_{i\in S} \pi_i^{(t)} \nabla \tilde{F}_i(w^{(t)}), w^{(t)} - \overline{w}^\star \right\rangle &= \left\langle \nabla F_{\theta,S}(w^{(t)}), w^{(t)} - \overline{w}^\star \right\rangle \\ &\geq F^\nu_{\theta,S}(w^{(t)}) - F^\nu_{\theta,S}(\overline{w}^\star) + \frac{\lambda}{2}\|w^{(t)} - \overline{w}^\star\|^2 .\end{aligned}$$

This allows us to bound A as

$$\begin{aligned}A = &\left\langle \sum_{i\in S} \pi_i^{(t)} \nabla \tilde{F}_i(w^{(t)}), w^{(t)} - \overline{w}^\star \right\rangle \\ &+ \left\langle \sum_{i\in S} \pi_i^{(t)} \left(\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)}) \right), w^{(t)} - \overline{w}^\star \right\rangle \\ \geq &F^\nu_{\theta,S}(w^{(t)}) - F^\nu_{\theta,S}(\overline{w}^\star) + \frac{\lambda}{2}\|w^{(t)} - \overline{w}^\star\|^2 \\ &- \left| \left\langle \sum_{i\in S} \pi_i^{(t)} \left(\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)}) \right), w^{(t)} - \overline{w}^\star \right\rangle \right| .\end{aligned}$$

Next, using successively the triangle inequality, the Cauchy-Schwartz inequality and $(L+\lambda)$-smoothness of the $\tilde{F}_i$ yields:

$$\begin{aligned}&\left| \left\langle \sum_{i\in S} \pi_i^{(t)} \left(\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)}) \right), w^{(t)} - \overline{w}^\star \right\rangle \right| \\ &\leq \sum_{i\in S} \pi_i^{(t)} \left| \left\langle \nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)}), w^{(t)} - \overline{w}^\star \right\rangle \right| \\ &\leq \sum_{i\in S} \pi_i^{(t)} \|\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)})\| \|w^{(t)} - \overline{w}^\star\| \\ &\leq \sum_{i\in S} \pi_i^{(t)} (L+\lambda) \|w_{i,k}^{(t)} - w^{(t)}\| \|w^{(t)} - \overline{w}^\star\| .\end{aligned}$$

Finally, using $2|ab| \leq a^2/c^2 + c^2 b^2$ and the convexity of $t \mapsto t^2$,

$$\begin{aligned}&\left| \left\langle \sum_{i\in S} \pi_i^{(t)} \left(\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)}) \right), w^{(t)} - \overline{w}^\star \right\rangle \right| \\ &\leq \frac{4}{\lambda} \left(\sum_{i\in S} \pi_i^{(t)} (L+\lambda) \|w_{i,k}^{(t)} - w^{(t)}\| \right)^2 + \frac{\lambda}{4}\|w^{(t)} - \overline{w}^\star\|^2 \\ &\leq \frac{\lambda}{4}\|w^{(t)} - \overline{w}^\star\|^2 + \frac{4(L+\lambda)^2}{\lambda} \sum_{i\in S} \pi_i^{(t)} \|w_{i,k}^{(t)} - w^{(t)}\|^2 .\end{aligned}$$

Overall, we bound A as

$$\begin{aligned} A &\geq 2\gamma\,\mathbb{E}_t\Bigg[\sum_{k=0}^{\tau-1}\Big(F^{\nu}_{\theta,S}(w^{(t)}) - F^{\nu}_{\theta,S}(\overline{w}^{\star}) + \frac{\lambda}{4}\|w^{(t)} - \overline{w}^{\star}\|^2 \\ &\qquad\qquad - \frac{4\,(L+\lambda)^2}{\lambda}\sum_{i\in S}\pi_i^{(t)}\|w_{i,k}^{(t)} - w^{(t)}\|^2\Big)\Bigg] \\ &\geq 2\gamma\tau\left(\overline{F}^{\nu}_{\theta}(w^{(t)}) - \overline{F}^{\nu}_{\theta}(\overline{w}^{\star})\right) + \frac{\lambda\gamma\tau}{2}\|w^{(t)} - \overline{w}^{\star}\|^2 - \frac{8\gamma\,(L+\lambda)^2}{\lambda}d^{(t)}\,, \end{aligned}$$

where we use the definition of $d^{(t)}$ from (D.10). We bound B using Proposition D.8. Putting these together, we get,

$$\begin{aligned} \mathbb{E}_t\|w^{(t+1)} - \overline{w}^{\star}\|^2 \leq &\left(1 - \frac{\lambda\gamma\tau}{2}\right)\|w^{(t)} - \overline{w}^{\star}\|^2 \\ &- (2\gamma\tau - 4\gamma^2\tau^2L')(\overline{F}^{\nu}_{\theta}(w^{(t)}) - \overline{F}^{\nu}_{\theta}(\overline{w}^{\star})) \\ &+ \frac{16\tau^2G^2\gamma^2}{\theta m} + 2\left(\gamma^2\tau\,(L+\lambda)^2 + 4\gamma\frac{(L+\lambda)^2}{\lambda}\right)d^{(t)}\,. \end{aligned}$$

With $\gamma \leq (4\tau L')^{-1}$ we have $2\gamma\tau - 4\gamma^2\tau^2L' \geq \gamma\tau$. Likewise, the same condition on γ also implies $2(\gamma\,(L+\lambda)^2 + 4(L+\lambda)^2/(\tau\lambda)) \leq 9\,(L+\lambda)^2/(\tau\lambda)$. Rearranging completes the proof. □

Proof of Theorem D.2. We are now ready to prove the theorem.

Proof of Theorem D.2. Plugging in the client drift bound of Proposition D.7 into the bound of Proposition D.3 and rearranging, we get

$$\begin{aligned} &\left(1 - \frac{18L'\,(L+\lambda)^2\,\tau^2\gamma^2e^2}{\lambda}\right)\left(\overline{F}^{\nu}_{\theta}(w^{(t)}) - \overline{F}^{\nu}_{\theta}(\overline{w}^{\star})\right) \\ &\qquad\leq \frac{1}{\gamma\tau}\left(1 - \frac{\lambda\gamma\tau}{2}\right)\|w^{(t)} - \overline{w}^{\star}\|^2 - \frac{1}{\gamma\tau}\mathbb{E}_t\|w^{(t+1)} - \overline{w}^{\star}\|^2 \\ &\qquad\quad + \frac{16\tau G^2\gamma}{\theta m} + \frac{9G^2\,(L+\lambda)^2\,\tau^2\gamma^2e^2}{\lambda}\left(4 + \frac{8}{\theta m}\right)\,. \end{aligned}$$

The constraint $\gamma \leq \sqrt{\lambda}(18\tau\,(L+\lambda)\sqrt{L'})^{-1}$ together with the numerical bound $36e^2 \leq 18^2$ implies $18L'\,(L+\lambda)^2\,\tau^2\gamma^2e^2/\lambda \leq \frac{1}{2}$. This gives

$$\begin{aligned} \overline{F}^{\nu}_{\theta}(w^{(t)}) - \overline{F}^{\nu}_{\theta}(\overline{w}^{\star}) \leq &\frac{2}{\gamma\tau}\left(1 - \frac{\lambda\gamma\tau}{2}\right)\|w^{(t)} - \overline{w}^{\star}\|^2 - \frac{2}{\gamma\tau}\mathbb{E}_t\|w^{(t+1)} - \overline{w}^{\star}\|^2 \\ &+ \underbrace{\frac{32\tau G^2\gamma}{\theta m} + \frac{18G^2\,(L+\lambda)^2\,\tau^2\gamma^2e^2}{\lambda}\left(4 + \frac{8}{\theta m}\right)}_{=:\mathcal{T}_1}\,. \end{aligned}$$

Next, we use convexity to get

$$\mathbb{E}\left[\overline{F}_\theta^\nu(\overline{w}^{(T)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right] \\ \leq \frac{1}{\sum_{t=0}^T \left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+t)}} \sum_{t=0}^T \left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+t)} \mathbb{E}\left[\overline{F}_\theta^\nu(w^{(t)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right]$$

so that telescoping the sum yields

$$\mathbb{E}\left[\overline{F}_\theta^\nu(\overline{w}^{(T)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right] \leq \frac{2\left\|w^{(0)} - \overline{w}^\star\right\|^2}{\gamma\tau\sum_{t=0}^T \left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+t)}} + \mathcal{T}_1 \ .$$

Now, we can lower bound the denominator with

$$\sum_{t=0}^T \left(1-\frac{\lambda\gamma\tau}{2}\right)^{-(1+t)} \geq \frac{1}{\gamma\tau\lambda} e^{T\gamma\tau\lambda} \,,$$

to get the bound

$$\mathbb{E}\left[\overline{F}_\theta^\nu(\overline{w}^{(T)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right] \leq 2\lambda e^{-T\gamma\tau\lambda}\|w^{(0)} - \overline{w}^\star\|^2 + \mathcal{T}_1 \ . \tag{D.11}$$

It remains to translate the results on $\overline{F}_\theta^\nu$ into F_θ. For the left hand side, we use the bias bound of Property D.4. For the right hand side, we use the λ-strong convexity of F_θ and Property D.4 we have:

$$\begin{aligned}
\|w^{(0)} - \overline{w}^\star\|^2 &\leq 2\|w^{(0)} - w^\star\|^2 + 2\|\overline{w}^\star - w^\star\|^2 \\
&\leq 2\|w^{(0)} - w^\star\|^2 + \frac{4}{\lambda}\left(F_\theta(\overline{w}^\star) - F_\theta(w^\star)\right) \\
&\leq 2\|w^{(0)} - w^\star\|^2 \\
&\quad + \frac{4}{\lambda}\left(F_\theta(\overline{w}^\star) - \overline{F}_\theta^\nu(\overline{w}^\star) + \overline{F}_\theta^\nu(\overline{w}^\star) - \overline{F}_\theta^\nu(w^\star) + \overline{F}_\theta^\nu(w^\star) - F_\theta(w^\star)\right) \\
&\leq 2\|w^{(0)} - w^\star\|^2 + \frac{4}{\lambda}\left(\frac{2B}{\sqrt{\theta m}} + 4\,\nu\log m\right) \quad ,
\end{aligned}$$

since $\overline{F}_\theta^\nu(\overline{w}^\star) - \overline{F}_\theta^\nu(w^\star) \leq 0$. Plugging this into (D.11) gives us the bound

$$\begin{aligned}
\mathbb{E}\left[F_\theta(\overline{w}^{(T)}) - F_\theta(w^\star)\right] \leq\ & 4\lambda\|w^{(0)} - w^\star\|^2 e^{-T\gamma\tau\lambda} + \frac{32\tau G^2\gamma}{\theta m} + \\
& \frac{18G^2\left(L+\lambda\right)^2\tau^2\gamma^2e^2}{\lambda}\left(4 + \frac{8}{\theta m}\right) + \\
& \left(\frac{2B}{\sqrt{\theta m}} + 2\,\nu\log m\right)\left(1 + 8e^{-T\gamma\tau\lambda}\right) \,.
\end{aligned}$$

Hyperparameter Optimization. To complete the proof from here, it remains to optimize the learning rate γ and the smoothing parameter ν. We invoke Lemma D.9 to optimize for γ. Ignoring absolute

constants, this gives us the bound

$$\begin{aligned}\mathbb{E}\left[F_\theta(\overline{w}^{(T)})\right] - F_\theta(w^\star) \le\ & \lambda\Delta_0 \exp(-\lambda\tau\Gamma T) + \frac{G^2}{\theta m \lambda T}\log\left(1 \vee \frac{\lambda^2\Delta_0 T\theta m}{G^2}\right) \\ & + \frac{G^2\kappa^2}{\lambda T^2}(1-\tau^{-1})\log^2\left(1 \vee \frac{\lambda^2\Delta_0 T^2}{G^2\kappa^2}\right) \\ & + \frac{B}{\sqrt{\theta m}} + \nu\log m\,,\end{aligned}$$

where we take

$$\Gamma = \min\left\{\frac{\sqrt{\lambda}}{18\tau(L+\lambda)\sqrt{L'}}, \frac{1}{4\tau L'},\right\}$$

Next, we set ν. The two terms that depend on ν are

$$\begin{aligned}\lambda\Delta_0\exp(-\lambda\tau\Gamma T) + \nu\log m &= \lambda\Delta_0\exp\left(-\frac{T}{\left(\kappa + \frac{G^2}{\lambda\nu}\right) \vee \kappa\sqrt{\kappa + \frac{G^2}{\lambda\nu}}}\right) + \nu\log m \\ &\le \lambda\Delta_0\max\left\{\exp\left(-\frac{\lambda\nu T}{G^2}\right), \exp\left(-\frac{T}{\kappa}\sqrt{\frac{\lambda\nu}{2G^2}}\right)\right\} + \nu\log m\,.\end{aligned}$$

Assume now that $\nu \le 2G^2/(\lambda\kappa^2)$, so the first term in the max is active. The conditions of Lemma D.9 are met since $T \ge 2\kappa$ is assumed; that gives us the choice

$$\nu = \frac{G^2}{\lambda\kappa} \wedge \frac{2G^2}{\lambda T}\log\left(1 \vee \frac{\lambda^2\Delta_0 T}{2G^2\log m}\right)\,,$$

so that the error is bounded by

$$\lambda\Delta_0\exp\left(-\frac{T}{2\kappa}\right) + \frac{2G^2}{\lambda T}\log(m)\log\left(1 \vee \frac{\lambda^2\Delta_0 T}{2G^2\log m}\right)\,.$$

Likewise, if $\nu > 2G^2/(\lambda\kappa^2)$, the second term inside the max is active. The conditions of Lemma D.9 are met since $T \ge \sqrt{2\kappa^3}$ is assumed. That gives us the choice

$$\nu \le \frac{G^2}{\lambda\kappa} \wedge \frac{2G^2\kappa^2}{\lambda T^2}\log^2\left(1 \vee \frac{\lambda^2\Delta_0 T^2}{2G^2\kappa^2\log m}\right)\,,$$

so that the error is bounded by

$$\lambda\Delta_0\exp\left(-\frac{T}{\sqrt{2\kappa^3}}\right) + \frac{2G^2\kappa^2}{\lambda T^2}\log(m)\log^2\left(1 \vee \frac{\lambda^2\Delta_0 T^2}{2G^2\kappa^2\log m}\right)\,.$$

Plugging in these choices completes the proof. □

D.1.3 Intermediate Results

We present some prerequisites and some intermediate results which are required in the convergence proofs of both the convex and nonconvex cases.

Note that for any $S \subset [n]$ of size m, the partial superquantile is differentiable at w with :

$$\nabla F^{\nu}_{\theta,S}(w) = \sum_{i \in S} \pi^{\star}_i \nabla \tilde{F}_i(w) \tag{D.12}$$

where $\pi^{\star}$ denotes solution to the maximization

$$F^{\nu}_{\theta,S}(w) = \max_{\pi \in \mathcal{P}_{\theta,S}} \sum_{i \in S} \pi_i \tilde{F}_i(w) - \nu \, D_S(\pi)$$

Bias and variance of the partial superquantile. We use the partial superquantile defined on a subset $S \subset [n]$ to approximate the full superquantile. We start with the quality of this approximation.

Property D.4. *Let U_m denote the uniform distribution over all subsets of $[n]$ of size m. For any $w \in \mathbb{R}^d$, we have*

$$\left|\overline{F}^{\nu}_{\theta}(w) - F_{\theta}(w)\right| \leq \frac{B}{\sqrt{\theta m}} + 2\,\nu \log m\,,$$
$$\mathbb{E}_{S \sim U_m} \left\|\nabla F^{\nu}_{\theta,S}(w) - \nabla \overline{F}^{\nu}_{\theta}(w)\right\|^2 \leq \frac{8G^2}{\theta m}\,.$$

Smoothing and smoothness constants. The following result is standard (Beck and Teboulle, 2012, Theorem 4.1, Lemma 4.2).

Property D.5. *For every $\nu > 0$, we have that $F^{\nu}_{\theta,S}$ and $\overline{F}^{\nu}_{\theta,S}$ are L'-smooth with $L' = L + \lambda + \frac{G^2}{\nu}$.*

Bounding Gradient Dissimilarity. Bounding of the variance of gradient estimators is a key assumption in the analysis of stochastic gradients methods (see e.g. the textbook (Bottou et al., 2018)). In the centralized setting, when a stochastic objective $\mathbb{E}_w i[f(w, wi)]$, it is standard to assume for a given estimator g_w of $\nabla_w \mathbb{E}\left[f(w, wi\right]$ that there exists some constants $M_1, M_2 > 0$ such that for all $w \in \mathbb{R}^d$,

$$\left\|\mathbb{E}\left[g_w\right]\right\|^2 \leq M_1 \quad \text{or} \quad \left\|\mathbb{E}\left[g_w\right]\right\|^2 \leq M_1 + M_2 \left\|\nabla_w \mathbb{E}\left[f(w, wi\right]\right\|^2 .$$

In the federated setting, the use of a subset $S \subset [n]$ of clients in each round induces noise on the estimation of the average gradient over the whole network. Thus, such assumption translates into a *bound on the gradient dissimilarity* among the clients (Karimireddy et al., 2020; Wang et al., 2019):

$$\frac{1}{n} \sum_{i \in [n]} \left\|\nabla \tilde{F}_i(w)\right\|^2 \leq M_1 + M_2 \left\|\frac{1}{n} \sum_{i \in [n]} \nabla \tilde{F}_i(w)\right\|^2 .$$

In this work, we also consider the minimization of the global loss F^{ν}_{θ} by a stochastic algorithm based on a partial participation of the clients in the network, with the additional difficulties that we only have access to a biased estimator $\overline{F}^{\nu}_{\theta}$ of the loss F^{ν}_{θ} and its gradient. In particular, the adaptive reweighting of the clients selected at each round does not permit the direct use of such assumption. We show instead in the next lemma that the variance of stochastic gradient estimator can also be bounded, thanks to the Lipschitz assumption.

Proposition D.6 (Gradient Dissimilarity). *Consider the quantities $\pi^{(t)}, w^{(t)}$ from Algorithm 6.3. We have,*

$$\mathbb{E}\left[\sum_{i\in S}\pi_i^{(t)}\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2\middle|\mathcal{F}_t\right]\leq\left(4+\frac{8}{\theta m}\right)G^2+\left\|\nabla\overline{F}_\theta^\nu(w^{(t)})\right\|^2.$$

Proof. We drop the superscript t throughout this proof. By centering the second moment (cf. (D.15)), we have:

$$\begin{aligned}\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_j(w)\right\|^2&=\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_i(w)-\nabla F_{\theta,S}^\nu(w)\right\|^2+\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2\\&=\sum_{i\in S}\pi_i\left\|\nabla F_i(w)-\sum_{j\in S}\pi_j\nabla F_j(w)\right\|^2+\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2.\end{aligned}$$

Now since the weights π_i sum to one, we may use the convexity of $\|\cdot\|^2$ to get:

$$\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_j(w)\right\|^2\leq\sum_{i,j\in S}\pi_i\pi_j\left\|\nabla F_j(w)-\nabla F_i(w)\right\|^2+\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2.$$

The squared triangle inequality (cf. (D.14)) together with the Lipschitz assumption on the functions F_i yields:

$$\begin{aligned}\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_i(w)\right\|^2&\leq 2\sum_{i,j\in S}\pi_i\pi_i\left(\|\nabla F_i(w)\|^2+\|\nabla F_j(w)\|^2\right)+\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2\\&\leq 4\,G^2+\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2.\end{aligned}$$

Thus, taking an expectation over $S\sim U_m$ gives

$$\mathbb{E}\left[\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_j(w)\right\|^2\middle|\mathcal{F}_t\right]\leq 4\,G^2+\mathbb{E}_{S\sim U_m}\left[\left\|\nabla F_{\theta,S}^\nu(w)\right\|^2\right].$$

By centering (cf. (D.15)), we get,

$$\begin{aligned}\mathbb{E}\left[\sum_{i\in S}\pi_i\left\|\nabla\tilde{F}_i(w)\right\|^2\middle|\mathcal{F}_t\right]\leq&\,4\,G^2+\left\|\nabla\overline{F}_\theta^\nu(w)\right\|^2\\&+\mathbb{E}\left[\left\|\nabla F_{\theta,S}^\nu(w)-\nabla\overline{F}_\theta^\nu(w)\right\|^2\middle|\mathcal{F}_t\right].\end{aligned}\tag{D.13}$$

Finally, substituting the variance bound from Property D.4 into (D.13) yields the stated result. □

Bounding the Client Drift. During federated learning, each client takes multiple local steps. This causes the resulting update to be a biased estimator of a descent direction for the global objective. This phenomenon has been referred to as "client drift" (Li et al., 2020e; Karimireddy et al., 2020). Current proof techniques rely on treating this as a "noise" term which is to be controlled. In the context of this work, the reweighting by $\pi^{(t)}$ requires us to adapt this typical definition of client drift to our setting. In particular, recall that we define the client drift $d^{(t)}$ in outer iteration t of the algorithm as

$$d^{(t)}:=\mathbb{E}_{S\sim U_m}\left[\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\|w_{i,k}^{(t)}-w^{(t)}\|^2\middle|\mathcal{F}_t\right].$$

Proposition D.7 (Client Drift). *If $\gamma \leq \frac{1}{4\tau(L+\lambda)}$, we have the following bounds for any $t \geq 0$:*

$$d^{(t)} \leq \tau^2(\tau-1)\gamma^2 e^2 \left(\left(4+\frac{8}{\theta m}\right) G^2 + \|\nabla \overline{F}_\theta^\nu(w^{(t)})\|^2\right) \text{ and,}$$
$$d^{(t)} \leq \tau^2(\tau-1)\gamma^2 e^2 \left(\left(4+\frac{8}{\theta m}\right) G^2 + 2L'\left(\overline{F}_\theta^\nu(w^{(t)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right)\right) .$$

Furthermore, if $\lambda = 0$, we have the bound

$$d^{(t)} \leq 8\tau^2(\tau-1)\gamma^2 G^2 .$$

The last bound also works without smoothing, i.e., $\nu = 0$.

Proof. If $\tau = 1$, there is nothing to prove as both sides of the inequality are 0. We assume now that $\tau > 1$. Let us first fix $S \subset [n]$ of size $|S| = m$. For any $k \in S$ and $j \in \{1, \ldots, \tau-1\}$, by the squared triangle inequality (cf. (D.14)), we have:

$$\begin{aligned}\left\|w_{i,k}^{(t)} - w^{(t)}\right\|^2 &= \left\|w_{i,k-1}^{(t)} - \gamma\nabla\tilde{F}_i(w_{i,k-1}^{(t)}) - w^{(t)}\right\|^2 \\ &\leq \left(1+\frac{1}{\tau-1}\right)\left\|w_{i,k-1}^{(t)} - w^{(t)}\right\|^2 + \tau\gamma^2\left\|\nabla\tilde{F}_i(w_{i,k-1}^{(t)})\right\|^2 .\end{aligned}$$

The squared triangle inequality (cf. (D.14)) together with the smoothness of the local losses gives:

$$\begin{aligned}\left\|w_{i,k}^{(t)} - w^{(t)}\right\|^2 &\leq \left(1+\frac{1}{\tau-1}\right)\left\|w_{i,k-1}^{(t)} - w^{(t)}\right\|^2 \\ &\quad + 2\tau\gamma^2\left(\left\|\nabla\tilde{F}_i(w_{i,k-1}^{(t)}) - \nabla\tilde{F}_i(w^{(t)})\right\|^2 + \left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2\right) \\ &\leq \left(1+\frac{1}{\tau-1}\right)\left\|w_{i,k-1}^{(t)} - w^{(t)}\right\|^2 \\ &\quad + 2\tau\gamma^2(L+\lambda)^2\left\|w_{i,k-1}^{(t)} - w^{(t)}\right\|^2 + 2\tau\gamma^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2 .\end{aligned}$$

Hence, for $\gamma \leq \frac{1}{4\tau(L+\lambda)}$, we get:

$$\left\|w_{i,k}^{(t)} - w^{(t)}\right\|^2 \leq \left(1+\frac{2}{\tau-1}\right)\left\|w_{i,k-1}^{(t)} - w^{(t)}\right\|^2 + 2\tau\gamma^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2 .$$

Unrolling this recursion yields for any $j \leq \tau - 1$

$$\begin{aligned}\left\|w_{i,k}^{(t)} - w^{(t)}\right\|^2 &\leq \sum_{i=0}^{j-1}\left(1+\frac{2}{\tau-1}\right)^i\left(2\tau\gamma^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2\right) \\ &\leq \frac{\tau-1}{2}\left(1+\frac{2}{\tau-1}\right)^j\left(2\tau\gamma^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2\right) \\ &\leq \frac{\tau-1}{2}\left(1+\frac{2}{\tau-1}\right)^{\tau-1}\left(2\tau\gamma^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2\right) \\ &\leq \tau(\tau-1)\gamma^2 e^2\left\|\nabla\tilde{F}_i(w^{(t)})\right\|^2 ,\end{aligned}$$

where we use $(1+1/x)^x \leq e$ for any $x > 0$. If $\lambda = 0$ we have that $\left\|\nabla \tilde{F}_i(w^{(t)})\right\|^2 = \left\|\nabla F_i(w^{(t)})\right\|^2 \leq G^2$ since F_i is G-Lipschitz; this gives us the final bound in the statement. When $\lambda \neq 0$, this does not hold. In this case, we apply Proposition D.6 to get

$$\begin{aligned} d^{(t)} &\leq \tau^2(\tau-1)\gamma^2 e^2\, \mathbb{E}_{S\sim U_m}\left[\sum_{i\in S}\pi_i^{(t)}\left\|\nabla \tilde{F}_i(w^{(t)})\right\|^2 \,\middle|\, \mathcal{F}^{(t)}\right] \\ &\leq \tau^2(\tau-1)\gamma^2 e^2\left(\left(4+\frac{8}{\theta m}\right)G^2 + \left\|\nabla \overline{F}_\theta^\nu(w^{(t)})\right\|^2\right). \end{aligned}$$

This gives the first bound. The second bound follows from the first by smoothness (cf. (D.17)). □

Bound on the Norm of Each Update. We bound the expected squared norm of each update $w^{(t+1)} - w^{(t)}$, which has the closed form expression:

$$w^{(t+1)} - w^{(t)} = -\gamma \sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla \tilde{F}_i(w_{i,k}^{(t)}).$$

Proposition D.8. *We have the bounds,*

$$\begin{aligned} &\gamma^2\mathbb{E}\left[\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla \tilde{F}_i(w_{i,k}^{(t)})\right\|^2 \,\middle|\, \mathcal{F}_t\right] \\ &\quad\leq 2\gamma^2\tau\,(L+\lambda)^2\; d^{(t)} + \frac{16\tau^2\gamma^2G^2}{\theta m} + 2\tau^2\gamma^2\left\|\nabla \overline{F}_\theta^\nu(w^{(t)})\right\|^2 \\ &\quad\leq 2\gamma^2\tau\,(L+\lambda)^2\; d^{(t)} + \frac{16\tau^2\gamma^2G^2}{\theta m} + 4\tau^2\gamma^2L'\left(\overline{F}_\theta^\nu(w^{(t)}) - \overline{F}_\theta^\nu(\overline{w}^\star)\right), \end{aligned}$$

where $d^{(t)}$ is the client drift term defined in (D.10).

Proof. Using the the squared triangle inequality (cf. Eq. (D.14)) together with the gradient formula (D.12), we get:

$$\begin{aligned} &\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\nabla \tilde{F}_i(w_{i,k}^{(t)})\right\|^2 \\ &\leq 2\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{j=0}^{\tau-1}\left(\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)})\right)\right\|^2 + 2\left\|\sum_{i\in S}\pi_i^{(t)}\sum_{j=0}^{\tau-1}\nabla \tilde{F}_i(w^{(t)})\right\|^2 \\ &\leq 2\tau\sum_{i\in S}\pi_i^{(t)}\sum_{k=0}^{\tau-1}\left\|\nabla \tilde{F}_i(w_{i,k}^{(t)}) - \nabla \tilde{F}_i(w^{(t)})\right\|^2 + 2\tau^2\left\|\nabla F_{\theta,S}^\nu(w^{(t)})\right\|^2. \end{aligned}$$

For the first term, we invoke $(L+\lambda)$-smoothness of $\tilde{F}_i$ and take an expectation to get $2\tau(L+\lambda)^2 d^{(t)}$. For the second term, we use centering (cf. Eq. (D.15)) followed by the variance bound of Property D.4 to get:

$$\begin{aligned}\mathbb{E}_t\left[\left\|\nabla F^{\nu}_{\theta,S}(w^{(t)})\right\|^2\right] &= \mathbb{E}_t\left[\left\|\sum_{i\in S}\pi_i^{(t)}\nabla\tilde{F}_i(w^{(t)}) - \nabla\overline{F}^{\nu}_{\theta}(w^{(t)})\right\|^2\right] + \left\|\nabla\overline{F}^{\nu}_{\theta}(w^{(t)})\right\|^2 \\ &\leq \frac{8G^2}{\theta m} + \left\|\nabla\overline{F}^{\nu}_{\theta}(w^{(t)})\right\|^2 .\end{aligned}$$

This gives the first bound. The second bound follows from the first by smoothness (cf. Eq. (D.17)). □

D.1.4 Useful Inequalities and Technical Results

We recall a few standard inequalities:

- Squared Triangle inequality: For any $x, y \in \mathbb{R}^d$ and $\alpha > 0$ we have:

$$\|x+y\|^2 \leq (1+\alpha)\|x\|^2 + \left(1+\frac{1}{\alpha}\right)\|y\|^2 . \tag{D.14}$$

- Centering the second moment: For any $\mathbb{R}^d$-valued random vector X such that $\mathbb{E}\|X\|^2 < \infty$,

$$\mathbb{E}\|X\|^2 = \mathbb{E}\|X - \mathbb{E}[X]\|^2 + \|\mathbb{E}[X]\|^2 \tag{D.15}$$

- Strong convexity: Let $F : \mathbb{R}^d \to \mathbb{R}$ be μ-strongly convex. Then for any $x, y \in \mathbb{R}^d$, we have:

$$\langle \nabla F(x), x-y\rangle \geq F(x) - F(y) + \frac{\mu}{2}\|x-y\|^2 \tag{D.16}$$

- Smoothness: Let $F : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and let $F^\star$ be the minimum value of F (assuming it exists). Then for any $x \in \mathbb{R}^d$, we have:

$$\|\nabla F(x)\|^2 \leq 2L\,(F(x) - F^\star) \tag{D.17}$$

Lemma D.9. *Consider the map $\varphi : (0, \Gamma] \to \mathbb{R}_+$ given by*

$$\varphi(\gamma) = A\exp(-\gamma T) + B\gamma + C\gamma^2 ,$$

where $\Gamma, A, B, C > 0$ are given. If $T > 1/\Gamma$, then, we have,

$$\varphi(\gamma^\star) \leq A\exp(-\Gamma T) + \frac{B}{T} + \frac{B}{T}\log\left(1 \vee \frac{AT}{B}\right) + \frac{C}{T^2} + \frac{C}{T^2}\log^2\left(1 \vee \frac{AT^2}{C}\right) ,$$

where $\gamma^\star$ is given by

$$\gamma^\star = \min\left\{\Gamma, \frac{1}{T}\log\left(1 \vee \frac{AT}{B}\right), \frac{1}{T}\log\left(1 \vee \frac{AT^2}{C}\right)\right\} .$$

Proof. Define $\gamma_1 = T^{-1}\log(1 \vee AT/B)$ and $\gamma_2 = T^{-1}\log(1 \vee AT^2/C)$. If $\gamma^\star = \Gamma$, we have that $\Gamma \le \gamma_1$ and $\Gamma \le \gamma_2$ so that

$$\varphi(\gamma^\star) = A\exp(-\Gamma T) + B\Gamma + C\Gamma^2 \le A\exp(-\Gamma T) + B\gamma_1 + C\gamma_2^2\,.$$

Now suppose that $\gamma^\star = \gamma_1$ so that $\gamma_1 \le \gamma_2$. Then, we have,

$$\varphi(\gamma^\star) = A\exp(-\gamma_1 T) + B\gamma_1 + C\gamma_1^2 \le \frac{B}{T} + \frac{B}{T}\log(1 \vee AT/B) + C\gamma_2^2\,.$$

The third case is identical to the second. □

The proof of the next lemma is elementary and is omitted.

Lemma D.10. *Consider the map $\varphi : (0, \Gamma] \to \mathbb{R}_+$ given by*

$$\varphi(\gamma) = \frac{A}{\gamma T} + B\gamma + C\gamma^2\,,$$

where $\Gamma, A, B, C > 0$ are given. Then, we have,

$$\varphi(\gamma^\star) \le \frac{A}{\Gamma T} + 2\left(\frac{AB}{T}\right)^{1/2} + 2C^{1/3}\left(\frac{A}{T}\right)^{2/3}\,,$$

where $\gamma^\star$ is given by

$$\gamma^\star = \min\left\{\Gamma, \sqrt{\frac{A}{BT}}, \left(\frac{A}{CT}\right)^{1/3}\right\}\,.$$

D.2 Privacy Analysis

The discrete Gaussian mechanism was introduced in (Canonne et al., 2020) as an extension of the Gaussian mechanism to integer data. A random variable ξ is said to satisfy the discrete Gaussian distribution with mean μ and variance proxy σ^2 if

$$\mathbb{P}(\xi = n) = C\,\exp\left(-\frac{(n-\mu)^2}{2\sigma^2}\right) \quad \text{for all } n \in \mathbb{Z}\,,$$

where C is an appropriate normalizing constant. We denote it by $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$. We need the following property of the discrete Gaussian.

Property D.11. *Let ξ be distributed according to $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$. Then, $\mathbb{E}[\xi] = \mu$. Furthermore, if $\mu = 0$, then ξ is sub-Gaussian with variance proxy σ^2, i.e., $\mathbb{E}[\exp(\lambda\xi)] \le \exp(\lambda^2\sigma^2/2)$ for all $\lambda > 0$.*

We now give the full proof of Theorem 6.8

Proof of Theorem 6.8. We start by defining and controlling the probabilities of some events. Throughout, let $\delta > 0$ be fixed. Define the event

$$E_{\mathrm{mod}} = \bigcap_{i=1}^{n}\bigcap_{j=1}^{b}\left\{-\frac{M-2}{2n} \le cx_{i,j} + \xi_{i,j} \le \frac{M-2}{2n}\right\}\,. \tag{D.18}$$

Note that under E_{mod}, no modular wraparound occurs in the algorithm, i.e., $\tilde{x}_i = cx_i + \xi_i$ and

$$\hat{h} = \sum_{i=1}^{n}\frac{\tilde{x}_i}{c} = \sum_{i=1}^{n}\left(x_i + \frac{\xi_i}{c}\right)\,.$$

We will show later that E_{mod} holds with high probability; for now, we assume that it holds.

Privacy Analysis. We start by establishing the sensitivity of the sum query over x_i's as 1. Define the input space $\mathcal{X}$ to be the canonical basis vectors in $\mathbb{R}^b$, i.e., the set of all vectors in $\{0,1\}^b$ with only one 1, and let $\mathcal{X}^* = \cup_{r=1}^{\infty} \mathcal{X}^r$ denote the set of all sequences of elements of $\mathcal{X}$. We consider the rescaled sum query $A((x_1, \cdots, x_N)) = \sum_{i=1}^{n} c x_i$. The L_2 sensitivity $S(A)$ of this query A is supremum over all $X \in \mathcal{X}^*$ and X' which is obtained by concatenating x' to X:

$$S(A) = \sup_{X, X'} \|A(X) - A(X')\|_2 = \sup_{x' \in \mathcal{X}} c\|x'\|_2 = c\,.$$

We invoke the privacy bound of sums of discrete Gaussians (Lemma D.14) to claim that an algorithm $\mathcal{A}$ returning $A(x) + \sum_{i=1}^{n} \xi_i$ satisfies $(1/2)\varepsilon^2$-concentrated DP where ε is as in the theorem statement. The fact that the quantile and all further functions of it remains private follows from the post-processing property of DP (also known as the data-processing inequality).

Utility Analysis. Define $\hat{n} = \sum_{j=1}^{b} \hat{h}_j$, as the noisy analogue to $n = \sum_{j=1}^{b} h_j$. Below, we use shorthand $\rho = 1 - \theta$. We bound the quantile error as

$$\begin{aligned}
\Delta_\theta\left(\hat{h}, h\right) = R_\theta\left(h, j_\theta^*(\hat{h})\right) &= \left|\frac{1}{n}\sum_{j=1}^{j_\theta^*(\hat{h})} h_j - \rho\right| \\
&\le \frac{1}{n}\left|\sum_{j=1}^{j_\theta^*(\hat{h})} h_j - \hat{h}_j\right| + \frac{1}{n}\left|\sum_{j=1}^{j_\theta^*(\hat{h})} \hat{h}_j - \hat{n}\rho\right| + \frac{\rho}{n}\left|\hat{n} - n\right| \\
&\le \max_{j' \in [b]} \frac{1}{cn}\left|\sum_{j=1}^{j'}\sum_{i=1}^{n} \xi_{i,j}\right| + \left(1 + \frac{|\hat{n} - n|}{n}\right) R_\theta^*(\hat{h}) + \frac{\rho}{n}\left|\hat{n} - n\right|\,.
\end{aligned}$$

Let us define an event E_{sum} under which the first term and last terms are bounded:

$$E_{\text{sum}} = \left\{\max_{j \in [b]}\left|\sum_{j'=1}^{j}\sum_{i=1}^{n} \xi_{i,j'}\right| \le \sqrt{2\sigma^2 nb \log(4/\delta)}\right\}\,. \tag{D.19}$$

Under E_{sum}, we also have

$$|n - \hat{n}| = \frac{1}{c}\left|\sum_{j=1}^{b}\sum_{i=1}^{n} \xi_{i,j}\right| \le \sqrt{\frac{2\sigma^2 nb}{c^2}\log\frac{4}{\delta}}\,.$$

Plugging this back into $\Delta_\rho(\hat{h}, h)$ gives us the desired bound, provided E_{sum} holds.

Bounding the Failure Probability. The algorithm fails when at least one of E_{mod} or E_{sum} fail to hold. We have from Claim D.12 that $\mathbb{P}(E_{\text{mod}}) \ge 1 - \delta/2$ and from Claim D.13 that $\mathbb{P}(E_{\text{sum}}) \ge 1 - \delta/2$. With a union bound, we get that $\mathbb{P}(E_{\text{sum}} \cap E_{\text{prod}}) \ge 1 - \delta$, i.e., the algorithm succeeds with probability at least $1 - \delta$. □

We state and prove bounds on probabilities of the events $E_{\text{mod}}, E_{\text{sum}}$ defined above.

Claim D.12. *If* $M \ge 2 + 2cn + 2n\sqrt{2\sigma^2 \log(4nb/\delta)}$, *then* $\mathbb{P}(E_{\text{mod}}) \ge 1 - \delta/2$.

Proof. Each discrete Gaussian random variable $\xi_{i,j}$ is centered and sub-Gaussian with variance proxy σ^2 (cf. Property D.11). A Cramér-Chernoff bound (cf. Lemma D.15) gives us the exponential tail bound

$$\mathbb{P}\left(|\xi_{i,j}| > \sqrt{2\sigma^2 \log(4nb/\delta)}\right) \leq \frac{\delta}{2nb}\,.$$

Using a union bound for $i \in [n], j \in [b]$ and $x_{i,j} \in \{0,1\}$ completes the proof. □

Claim D.13. *We have that* $\mathbb{P}(E_{\text{sum}}) \geq 1 - \delta/2$.

Proof. Each discrete Gaussian random variable $\xi_{i,j}$ is centered and sub-Gaussian with variance proxy σ^2, i.e., $\mathbb{E}[\xi_{i,j}] = 0$ and $\mathbb{E}[\exp(\lambda\xi_{i,j})] \leq \exp(\lambda^2\sigma^2/2)$ for all $\lambda \in \mathbb{R}$ (cf. Property D.11). Therefore, $\zeta_j := \sum_{i=1}^n \xi_{i,j}$ is centered and sub-Gaussian with variance proxy $n\sigma^2$, since $\mathbb{E}[\zeta_j] = 0$, and

$$\mathbb{E}[\exp(\lambda\zeta_j)] = \prod_{i=1}^n \mathbb{E}[\exp(\lambda\xi_{i,j})] \leq \exp(\lambda^2\sigma^2 n/2)$$

by independence. We get a bound on the partial sums from Lemma D.16; this involves constructing a martingale $(\sum_{j'=1}^j \zeta_{j'})_{j=1}^b$ and applying the maximal inequality. The bound we get is

$$\mathbb{P}\left(\max_{j\in[b]} \left|\sum_{j'=1}^j \zeta_{j'}\right| > t\right) \leq \exp\left(-\frac{t^2}{2\sigma^2 nb}\right)\,.$$

Plugging in $t = \sqrt{2\sigma^2 nb\log(2/\delta)}$ completes the proof. □

D.2.1 Useful Results

The distributed discrete Gaussian mechanism gets privacy guarantees by adding a sum of discrete Gaussian random variables. We give a bound on its privacy. The following lemma is due to (Kairouz et al., 2021a).

Lemma D.14 (Privacy of Sum of Discrete Gaussians)**.** *Fix $\sigma \geq 1/2$. Let A be a deterministic algorithm with ℓ_2-sensitivity S. Define a randomized algorithm $\mathcal{A}$, which when given an input x, samples $\xi_1, \cdots, \xi_n \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma^2 I_d)$ and returns $A(x) + \sum_{i=1}^n \xi_i$. Then, $\mathcal{A}$ satisfies $\varepsilon^2/2$-concentrated DP with*

$$\varepsilon = \min\left\{\sqrt{\frac{S^2}{n\sigma^2} + \frac{\psi d}{2}}, \frac{S}{\sqrt{n}\sigma} + \psi\sqrt{d}\right\}\,,$$

where $\psi = 10\sum_{i=1}^{n-1} \exp\big(-2\pi^2\sigma^2 i/(k+1)\big) \leq 10(n-1)\exp(-2\pi^2\sigma^2)$.

Next, we record two standard concentration results.

Lemma D.15 (Cramér-Chernoff)**.** *Let ξ be a real-valued and centered sub-Gaussian random variable with variance proxy σ^2, i.e., $\mathbb{E}[\xi] = 0$ and $\mathbb{E}[\exp(\lambda\xi)] \leq \exp(\lambda^2\sigma^2/2)$ for all $\lambda > 0$. Then, we have for any $t > 0$,*

$$\mathbb{P}(|\xi| > t) \leq 2\exp\left(-\frac{t^2}{2\sigma^2}\right)\,.$$

Lemma D.16 (Maximal Inequality)**.** *Let $\xi_1, \xi_2, \cdots$ be i.i.d. centered sub-Gaussian random variables with variance proxy σ^2, i.e., $\mathbb{E}[\xi_j] = 0$ and $\mathbb{E}[\exp(\lambda\xi_j)] \leq \exp(\lambda^2\sigma^2/2)$ for all $\lambda \in \mathbb{R}$ and $j = 1, 2, \cdots$. Then, it holds for any $t > 0$ and integer $n \geq 1$ that*

$$\mathbb{P}\left(\max_{i\in[n]} \left|\sum_{j=1}^k \xi_j\right| > t\right) \leq 2\exp\left(-\frac{t^2}{2\sigma^2 n}\right)\,.$$

D.3 Numerical Experiments: Complete Results

We conduct our experiments on two datasets from computer vision and natural language processing. These datasets contain a natural, non-iid split of data which is reflective of data heterogeneity encountered in federated learning. In this section, we describe in details the experimental setup and the results. Here is its outline:

- Section D.3.1 describes the datasets and tasks.
- Section D.3.2 presents the algorithm and the hyperparameters used.
- Section D.3.3 details the evaluation methodology.
- Section D.3.4 gives the experimental comparison of Δ-FL to baselines.

Since each client has a finite number of datapoints in the examples below, we let its probability distribution π_i to be the empirical distribution over the available examples, and the weight α_i to be proportional to the number of datapoints available on the client.

D.3.1 Datasets and Tasks

We use the two following datasets, described in detail below. The data was preprocessed using LEAF (Caldas et al., 2018).

EMNIST for handwritten-letter recognition.

Dataset. EMNIST (Cohen et al., 2017) is a character recognition dataset. This dataset contains images of handwritten digits or letters, labeled with their identification (a-z,A-Z, 0-9). The images are grey-scaled pictures of $28 \times 28 = 784$ pixels.

Train and Test Devices. Each image is also annotated with the "writer" of the image, i.e., the human subject who hand-wrote the digit/letter during the data collection process. Each client corresponds to one writer. From this set of clients, we discard all clients containing less than 100 images. The remaining clients were partitioned into two groups — 1730 training and 1730 testing clients. For each experiment we subsampled 865 training and 865 testing clients for computational tractability, where the sampled clients vary based on the random seed of each experiment.

Model. We consider the following models for this task.

- **Linear Model**: We use a linear softmax regression model. In this case each F_i is convex. We train parameters $w \in \mathbb{R}^{62\times 784}$. Given an input image $x \in \mathbb{R}^{784}$, the score of each class $c \in [62]$ is the dot product $\langle w_c, x\rangle$. The probability p_c assigned to each class is then computed as a softmax: $p_c = \exp\langle w_c, x\rangle / \sum_{c'} \exp\langle w_{c'}, x\rangle$. The prediction for a given image is then the class with the highest probability.

- **ConvNet**: We also consider a convolutional neural network with two convolutional layers with max-pooling and one fully connected layer (F.C) of which outputs a vector in $\mathbb{R}^{62}$. The outputs of the ConvNet are scores with respect to each class. They are also used with a softmax operation to compute probabilities.

The loss used to train both models is the multinomial logistic loss $L(p, y) = -\log p_y$ where p denotes the vector of probabilities computed by the model and p_y denotes its y^{th} component. In the convex case we add a quadratic regularization term of the form $(\lambda/2)\|w\|_2^2$.

Sent140 for Sentiment Analysis.

Dataset. Sent140 (Go et al., 2009) is a text dataset of 1,600,498 tweets produced by 660,120 Twitter accounts. Each tweet is represented by a character string with emojis redacted. Each tweet is labeled with a binary sentiment reaction (i.e., positive or negative), which is inferred based on the emojis in the original tweet.

Train and Test Devices. Each client represents a twitter account and contains only tweets published by this account. From this set of clients we discarded all clients containing less that 50 tweets, and split the 877 remaining clients rest of clients into a train set and a test set of sizes 438 and 439 respectively. This split was held fixed for all experiments. Each word in the tweet is encoded by its 50-dimensional GloVe embedding (Pennington et al., 2014).

Model. We consider the following models.

- **Linear Model**: We consider a l_2-regularized linear logistic regression model where the parameter vector w is of dimension 50. In this case, each F_i is convex. We summarize each tweet by the average of the GloVe embeddings of the words of the tweet.
- **RNN**: The nonconvex model is a Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) built on the GloVe embeddings of the words of the tweet. The hidden dimension of the LSTM is same as the embedding dimension, i.e., 50. We refer to it as "RNN".

The loss function is the binary logistic loss.

D.3.2 Algorithms and Hyperparameters

Algorithm and Baselines.
The proposed Δ-FL is run for three values of $\theta \in \{0.8, 0.5, 0.1\}$. We compare it to the following baselines:

- FedAvg (McMahan et al., 2017): It is the de facto standard for the vanilla federated learning objective.
- FedAvg, θ: We also consider FedAvg with a random client subselection step: local updates are run on a fraction of the initial number of clients randomly selected per round. For each dataset, we try three values of, corresponding to the average number of clients selected by Δ-FL for the three values of θ used. In the main paper, we report as FedAvg-Sub the performance of FedAvg, θ with $\theta \in \{0.8, 0.5, 0.1\}$ which gives the best performance on Δ-FL (i.e., lowest 90^{th} percentile of test misclassification error). Here we report numbers for all values of θ considered.

- FedProx (Li et al., 2020b): It augments FedAvg with a proximal term but still minimizes the vanilla federated learning objective.

- q-FFL (Li et al., 2020d): It raises the per-client losses to the power $(1+q)$, where $q \geq 0$ is a parameter, in order to focus on clients with higher loss. We run q-FFL for values of q in $\{10^j, j \in \{-3, \cdots, 1\}\}$.

- AFL (Mohri et al., 2019): It aims to minimize the worst per-client loss. We implement it as an asymptotic version of q-FFL, using a large value of q, as this was found to yield better convergence with comparable performance (Li et al., 2020d). In the experiments we take $q = 10.0$.

The experiments are conducted on the datasets described in Section D.3.1.

Hyperparameters.

Rounds. We measure the progress of each algorithm by the number of calls to secure aggregation routine for weight vectors, i.e., the number of communication rounds.

For the experiments, we choose the number of communication rounds depending on the convergence of the optimization for FedAvg. For the EMNIST dataset, we run the algorithm for 3000 communication rounds with the linear model and 1000 for the ConvNet. For the Sent140 dataset, we run the 1000 communication rounds for the linear model and 600 for the RNN.

Devices per Round. We choose the same number of clients per round for each method, with the exception of $FedAvg, \theta$. All clients are assumed to be available and selections are made uniformly at random. In particular, we select 100 clients per round for all experiments with the exception of Sent140 RNN for which we used 50 clients per round.

Local Updates and Minibatch Size. Each selected client locally runs 1 epoch of mini-batch stochastic gradient descent locally. We used the default mini-batch of 10 for all experiments (McMahan et al., 2017), except for 16 for EMNIST ConvNet. This is because the latter experiments were run using on a GPU, as we describe in the section on the hardware.

Learning rate scheme. We now describe the learning rate γ_t used during *LocalUpdate*. For the linear model we used a constant fixed learning rate $\gamma_t \equiv \gamma_0$, while for the neural network models, we using a step decay scheme of the learning rate $\gamma_t = \gamma_0 c^{-\lfloor t/t_0 \rfloor}$ for some where γ_0 and $0 < c \leq 1$ are tuned. We tuned the learning rates only for the baseline FedAvg and used the same learning rate for the other baselines and Δ-FL at all values of θ.

For the neural network models, we fixed t_0 so that the learning rate was decayed once or twice during the fixed time horizon T. In particular, we used $t_0 = 400$ for EMNIST ConvNet (where $T = 1000$) and $t_0 = 200$ for Sent140 RNN (where $T = 600$). We tuned c from the set $\{2^{-3}, 2^{-2}, 2^{-1}, 1\}$, while the choice of the range of γ_0 depended on the dataset-model pair. The tuning criterion we used was the mean of the loss distribution over the training clients (with client i weighted by α_i) at the end of the time horizon. That is, we chose the γ_0, c which gave the best terminal training loss.

Table D.1: Metrics for the test misclassification error for EMNIST (Linear Model).

Method	Mean	Standard Deviation	10^{th} Percentile	Median	90^{th} Percentile
FedAvg	34.38 ± 0.38	18.39 ± 0.33	21.54 ± 0.35	32.61 ± 0.39	49.65 ± 0.67
FedAvg $\theta = 0.8$	34.20 ± 0.45	18.25 ± 0.22	$\mathbf{21.37} \pm 0.26$	32.10 ± 0.34	49.92 ± 1.16
FedAvg $\theta = 0.5$	34.51 ± 0.47	18.21 ± 0.30	21.40 ± 0.36	32.36 ± 0.59	50.28 ± 0.77
FedAvg $\theta = 0.1$	34.60 ± 0.46	18.58 ± 0.31	21.71 ± 0.37	32.54 ± 0.37	50.33 ± 1.28
FedProx	$\mathbf{33.82} \pm 0.30$	18.25 ± 0.23	21.37 ± 0.35	$\mathbf{31.75} \pm 0.20$	49.15 ± 0.74
q-FFL (Best $q = 1.0$)	34.71 ± 0.27	19.34 ± 0.30	22.33 ± 0.41	32.80 ± 0.23	49.90 ± 0.58
Tilted-ERM (Best $t = 1.0$)	34.15 ± 0.25	10.78 ± 0.30	22.43 ± 0.29	32.36 ± 0.23	48.59 ± 0.62
AFL	39.32 ± 0.27	25.42 ± 0.27	28.64 ± 0.43	38.16 ± 0.34	51.62 ± 0.28
Δ-FL $\theta = 0.8$	34.48 ± 0.26	19.16 ± 0.32	22.24 ± 0.32	32.85 ± 0.31	49.10 ± 0.24
Δ-FL $\theta = 0.5$	35.01 ± 0.20	20.46 ± 0.34	23.64 ± 0.22	33.83 ± 0.34	$\mathbf{48.44} \pm 0.38$
Δ-FL $\theta = 0.1$	38.32 ± 0.48	23.86 ± 0.59	27.27 ± 0.64	37.52 ± 0.67	50.34 ± 0.95

Tuning of the regularization parameter. The regularization parameter λ for linear models was tuned with cross validation from the set $\{10^{-k} : k \in \{3, \cdots, 8\}\}$. This was performed as described below.

For each dataset, we held out half the training clients as validation clients. Then, for different values of the regularization parameter, we trained a model with the (smaller subset of) training clients and evaluate its performance on the validation clients. We selected the value of the regularization parameter as the one which gave the smallest 90^{th} percentile of the misclassification error on the validation clients.

Baselines Parameters. We tune the proximal parameter of FedProx with cross validation. The procedure we followed is identical to the procedure we described above for the regularization parameter λ. The set of parameters tested is $\{10^{-j}, j \in \{0, \ldots, 3\}\}$. We did not attempt to tune the parameter q of q-FFL and report the performance of all values of q which we tried.

Hyperparameters of Δ-FL. We optimize Δ-FL via Algorithm 6.3 with a fixed number of local steps, corresponding to one epoch. For simplicity, we calculate the quantile exactly, assuming client losses are available to the server.

D.3.3 Evaluation Strategy and Other Details

Evaluation metrics. We record the loss of each training client and the misclassification error of each testing client, as measured on its local data.

The evaluation metrics noted in Section D.3.4 are the following : the weighted mean of the loss distribution over the training clients, the (unweighted) mean misclassification error over the testing clients, the weighted τ-percentile of the loss over the training client and the (unweighted) τ-percentile of the misclassification error over the testing clients for values of τ among $\{20, 50, 60, 80, 90, 95\}$. We also present the 90^{th} and 95^{th} superquantile of the test misclassification error (i.e., average misclassification error of the worst 10% and 5% of the clients respectively), as well as the average test misclassification error of the best 10% clients. The weight α_i used for training client i was set proportional to the number of datapoints on the client.

Table D.2: Metrics for the test misclassification error for EMNIST (ConvNet Model).

Method	Mean	Standard Deviation	10^{th} Percentile	Median	90^{th} Percentile
FedAvg	16.63 ± 0.50	$\mathbf{4.94} \pm 0.14$	$\mathbf{6.43} \pm 0.24$	15.34 ± 0.37	28.46 ± 1.07
FedAvg $\theta = 0.8$	15.95 ± 0.42	5.25 ± 0.19	6.86 ± 0.38	14.84 ± 0.24	26.82 ± 1.28
FedAvg $\theta = 0.5$	16.22 ± 0.23	5.06 ± 0.17	6.47 ± 0.28	15.05 ± 0.25	27.56 ± 0.81
FedAvg $\theta = 0.1$	15.97 ± 0.43	5.40 ± 0.42	7.10 ± 0.64	14.76 ± 0.20	26.35 ± 2.08
FedProx	16.01 ± 0.54	5.16 ± 0.32	6.68 ± 0.44	14.88 ± 0.29	27.01 ± 1.86
q-FFL (Best $q = 0.001$)	16.58 ± 0.30	5.05 ± 0.21	6.53 ± 0.20	15.40 ± 0.43	28.02 ± 0.80
Tilted-ERM (Best $t = 1.0$)	15.69 ± 0.38	7.31 ± 0.68	7.26 ± 0.51	$\mathbf{14.66} \pm 0.16$	25.46 ± 1.49
AFL	33.00 ± 0.37	20.38 ± 0.23	22.92 ± 0.23	31.58 ± 0.27	45.07 ± 1.00
Δ-FL $\theta = 0.8$	16.08 ± 0.40	5.60 ± 0.14	7.31 ± 0.29	14.85 ± 0.48	26.23 ± 1.15
Δ-FL $\theta = 0.5$	$\mathbf{15.48} \pm 0.30$	6.13 ± 0.15	8.08 ± 0.16	14.73 ± 0.22	$\mathbf{23.69 \pm 0.94}$
Δ-FL $\theta = 0.1$	16.37 ± 1.03	6.61 ± 0.42	8.28 ± 0.65	15.49 ± 1.03	25.45 ± 2.77

Table D.3: Metrics for the test misclassification error for Sent140 (Linear Model).

Method	Mean	Standard Deviation	10^{th} Percentile	Median	90^{th} Percentile
FedAvg	34.74 ± 0.31	12.16 ± 0.15	21.89 ± 0.24	34.81 ± 0.38	46.83 ± 0.54
FedAvg $\theta = 0.8$	34.47 ± 0.03	12.08 ± 0.16	21.69 ± 0.26	34.62 ± 0.17	46.59 ± 0.38
FedAvg $\theta = 0.5$	34.46 ± 0.07	12.11 ± 0.24	$\mathbf{21.55} \pm 0.51$	$\mathbf{34.48} \pm 0.20$	47.00 ± 0.40
FedAvg $\theta = 0.1$	34.79 ± 0.32	11.97 ± 0.37	22.08 ± 0.75	34.93 ± 0.35	46.69 ± 0.84
FedProx	34.74 ± 0.31	12.16 ± 0.15	21.89 ± 0.24	34.82 ± 0.39	46.83 ± 0.54
q-FFL (Best $q = 1.0$)	34.48 ± 0.06	11.96 ± 0.14	21.61 ± 0.24	34.57 ± 0.16	$\mathbf{46.38} \pm 0.40$
Tilted-ERM (Best $t = 1.0$)	34.71 ± 0.31	12.00 ± 0.14	21.83 ± 0.34	34.91 ± 0.39	46.70 ± 0.50
AFL	35.97 ± 0.08	11.83 ± 0.09	23.58 ± 0.28	36.09 ± 0.17	47.51 ± 0.32
Δ-FL $\theta = 0.8$	$\mathbf{34.41} \pm 0.22$	12.17 ± 0.11	21.77 ± 0.34	34.64 ± 0.25	46.44 ± 0.38
Δ-FL $\theta = 0.5$	35.28 ± 0.25	$\mathbf{11.68} \pm 0.40$	23.03 ± 0.38	35.55 ± 0.53	46.64 ± 0.41
Δ-FL $\theta = 0.1$	37.78 ± 0.89	12.86 ± 0.52	23.93 ± 0.99	37.80 ± 1.30	51.38 ± 1.07

Evaluation times. We evaluate the model during training process for once every l communication rounds. The value of l used was $l = 50$ for EMNIST linear model, $l = 10$ for EMNIST ConvNet, $l = 20$ for Sent140 linear model and $l = 25$ for Sent140 RNN.

Hardware. We run each experiment as a simulation as a single process. The linear models were trained on m5.8xlarge AWS instances, each with an Intel Xeon Platinum 8000 series processor with 128 GB of memory running at most 3.1 GHz. The neural network experiments were trained on workstation with an Intel i9 processor with 128 GB of memory at 1.2 GHz, and two Nvidia Titan Xp GPUs. The Sent140 RNN experiments were run on a CPU while the other neural network experiments were run using GPUs.

Software Packages. Our implementation is based on NumPy using the Python language. In the neural network experiments, we use PyTorch to implement the *LocalUpdate* procedure, i.e., the model itself and the automatic differentiation routines provided by PyTorch to make SGD updates.

Table D.4: Metrics for the test misclassification error for Sent140 (RNN Model).

Method	Mean	Standard Deviation	10^{th} Percentile	Median	90^{th} Percentile
FedAvg	30.16 ± 0.44	4.36 ± 1.26	10.06 ± 2.06	29.51 ± 0.33	49.66 ± 3.95 1
FedAvg $\theta = 0.8$	$\mathbf{29.85} \pm 0.46$	5.39 ± 1.32	11.90 ± 2.27	29.57 ± 0.31	46.93 ± 3.84 1
FedAvg $\theta = 0.5$	31.06 ± 1.01	$\mathbf{4.33} \pm 2.73$	$\mathbf{9.69} \pm 4.89$	30.14 ± 0.71	53.10 ± 7.22 1
FedAvg $\theta = 0.1$	31.96 ± 1.47	4.82 ± 2.09	11.65 ± 4.83	31.55 ± 1.13	52.87 ± 8.41 1
FedProx	30.20 ± 0.48	4.35 ± 1.23	10.37 ± 2.08	$\mathbf{29.51} \pm 0.32$	49.85 ± 4.07
q-FFL (Best $q = 0.01$)	29.99 ± 0.56	4.90 ± 1.66	10.98 ± 2.88	29.56 ± 0.39	48.65 ± 4.68
Tilted-ERM (Best $t = 1.0$)	30.13 ± 0.49	14.17 ± 2.10	13.18 ± 3.33	29.96 ± 0.84	46.54 ± 3.27
AFL	37.74 ± 0.65	9.90 ± 1.46	18.19 ± 1.99	36.95 ± 1.03	57.78 ± 1.19
Δ-FL $\theta = 0.8$	30.30 ± 0.33	6.75 ± 2.68	13.05 ± 3.87	29.92 ± 0.38	$\mathbf{46.46} \pm 4.39$
Δ-FL $\theta = 0.5$	33.58 ± 2.44	8.74 ± 3.98	16.77 ± 6.62	33.28 ± 2.27	50.47 ± 8.24
Δ-FL $\theta = 0.1$	51.97 ± 11.81	9.11 ± 5.47	16.67 ± 9.15	52.44 ± 13.21	86.44 ± 10.95

Randomness. Since several sampling routines appear in the procedures such as the selection of clients or the local stochastic gradient, we carry our experiments with five different seeds and plot the average metric value over these seeds. Each simulation is run on a single process. Where appropriate, we report one standard deviation from the mean.

D.3.4 Experimental Results

We now present the experimental results of the paper.

- We present different metrics on the distribution of test misclassification error over the clients, comparing Δ-FL to baselines.
- We study the convergence of Algorithm 6.3 for Δ-FL over the course of the optimization, and compare it with FedAvg.
- We plot the histograms of the distribution of losses over train clients as well as the test misclassification errors over test clients at the end of the training process.
- We present in the form of scatter plots the training loss and test misclassification error across clients achieved at the end of training, versus the number of local data points on the client.
- We present the number of clients having a loss greater than the quantile at each communication round for Δ-FL. This gives the effective number of clients selected in each round, cf. Proposition 6.3 and Remark 6.4.

Comparison to Baselines. We now present a detailed comparison of various statistics of the test misclassification error distribution for different methods in Table D.1- For each column the smallest mean over five random runs is highlighted in bold. Further, if no other method is within one standard deviation of this method, the entire entry (i.e., mean $\pm$ std) is highlighted in bold.

Histograms of Loss and Test Misc. Error over Devices. Here, we plot the histograms of the loss distribution over training clients and the misclassification error distribution over testing clients. We report the losses and errors obtained at the end of the training process. Each metric is averaged per client over

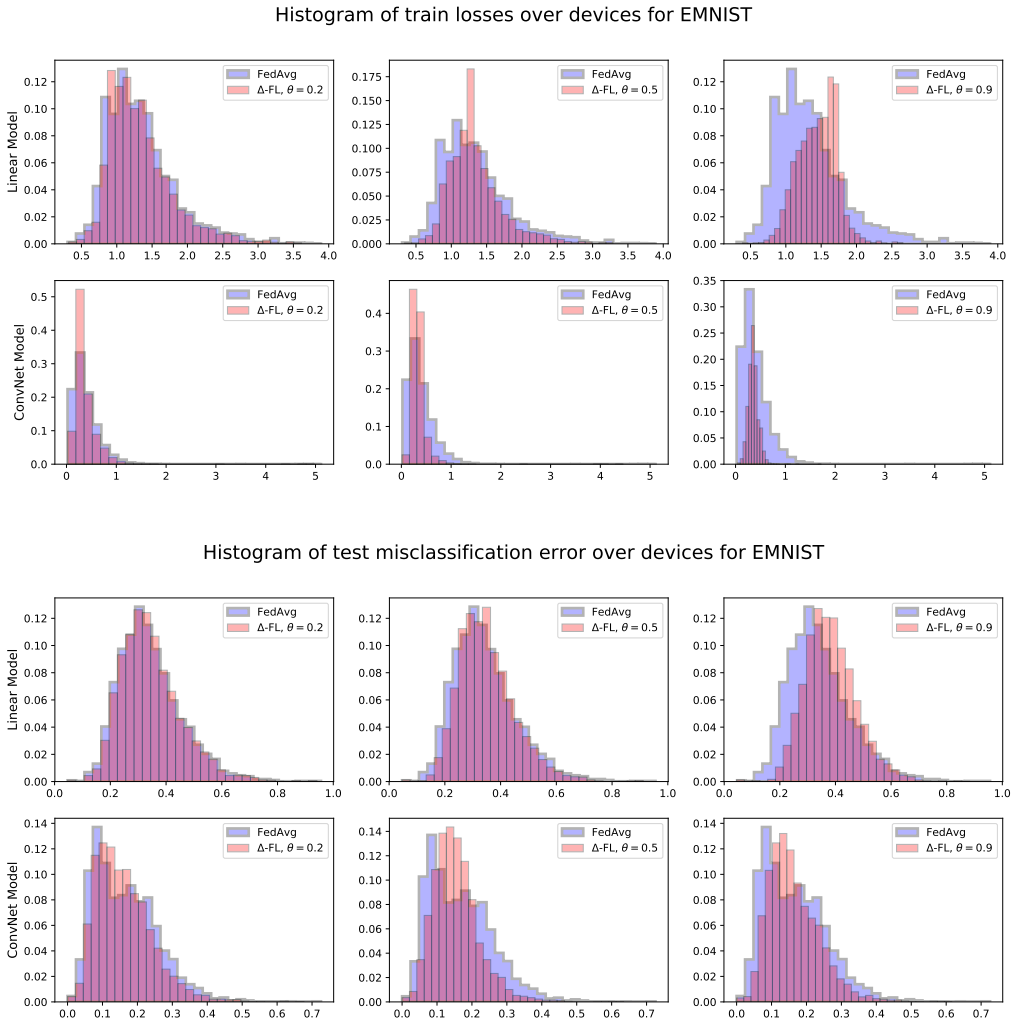


Figure D.1: Histogram of loss distribution over training clients and misclassification error distribution over testing clients for EMNIST. The identification of the model (linear or ConvNet) is given on the y-axis of the histograms.

5 runs of the random seed. Figure D.1 shows the histograms for EMNIST, while Figure D.2 shows the histograms for Sent140 dataset. for Sent140. We note that Δ-FL tends to exhibit thinner upper tails at at multiple values of θ and a lower variance of the distribution in most of the cases. This is also confirmed by the figures in Tables D.1 to D.4. This shows the benefit of using Δ-FL over vanilla FedAvg.

Performance compared to local data size. Next, we plot the loss on training clients versus the amount of local data on the client and the misclassification error on the test clients versus the amount of local data on the client. See Figure D.3 for EMNIST and Figure D.4 for Sent140.

Observe firstly that improvement over the worst cases is achieved regardless of the local data size of the clients. Indeed, the client re-weighting step operates a sorting of the loss of the clients which does not prevent small clients from being selected. In contrary, FedAvg, by averaging with respect to the weights

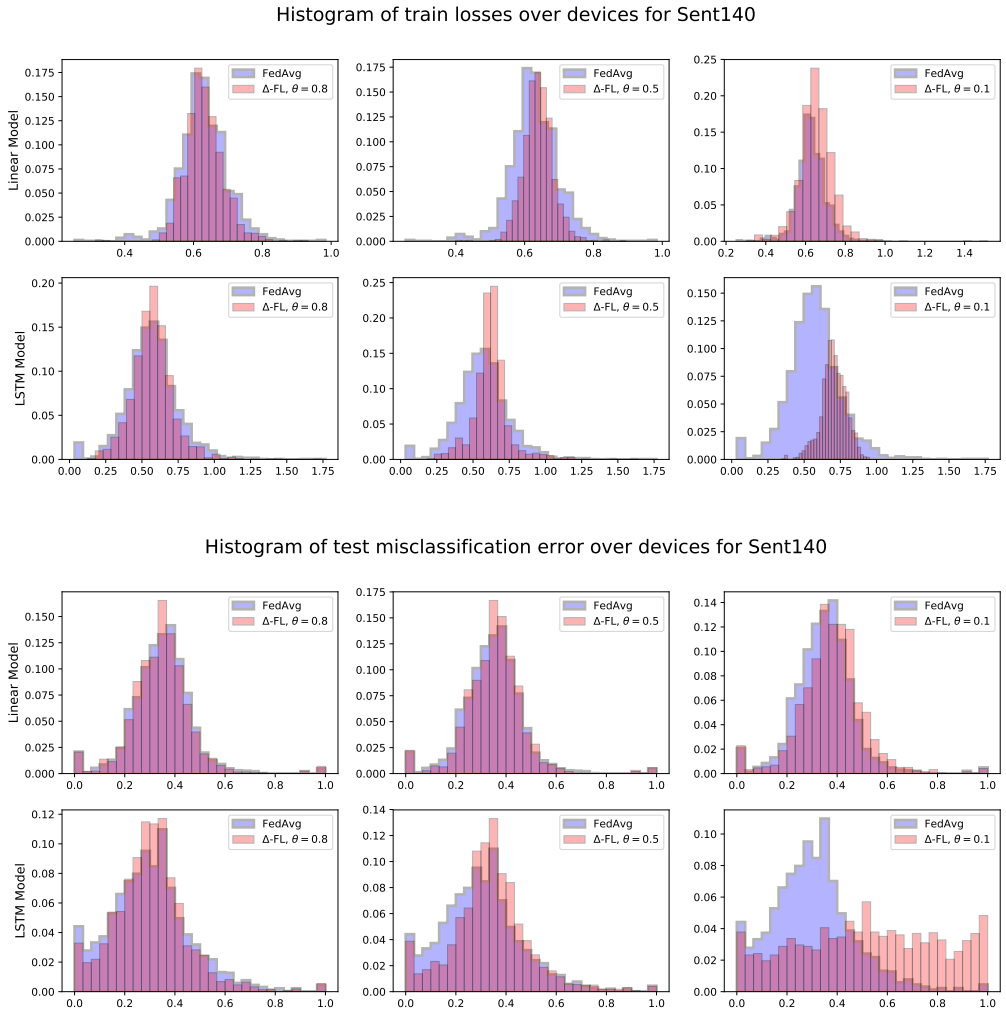


Figure D.2: Histogram of loss distribution over training clients and misclassification error distribution over testing clients for Sent140. The identification of the model (linear or RNN) is given on the y-axis of the histograms.

of the clients is likely to put more the accent on the clients with larger local data size. Secondly, Δ-FL appears to reduce the variance of of the loss on the train clients. Lastly, note that amongst test clients with a small number of data points (e.g., < 200 for EMNIST or < 100 for Sent140), Δ-FL reduces the variance of the misclassification error. Both effects are more pronounced on the neural network models.

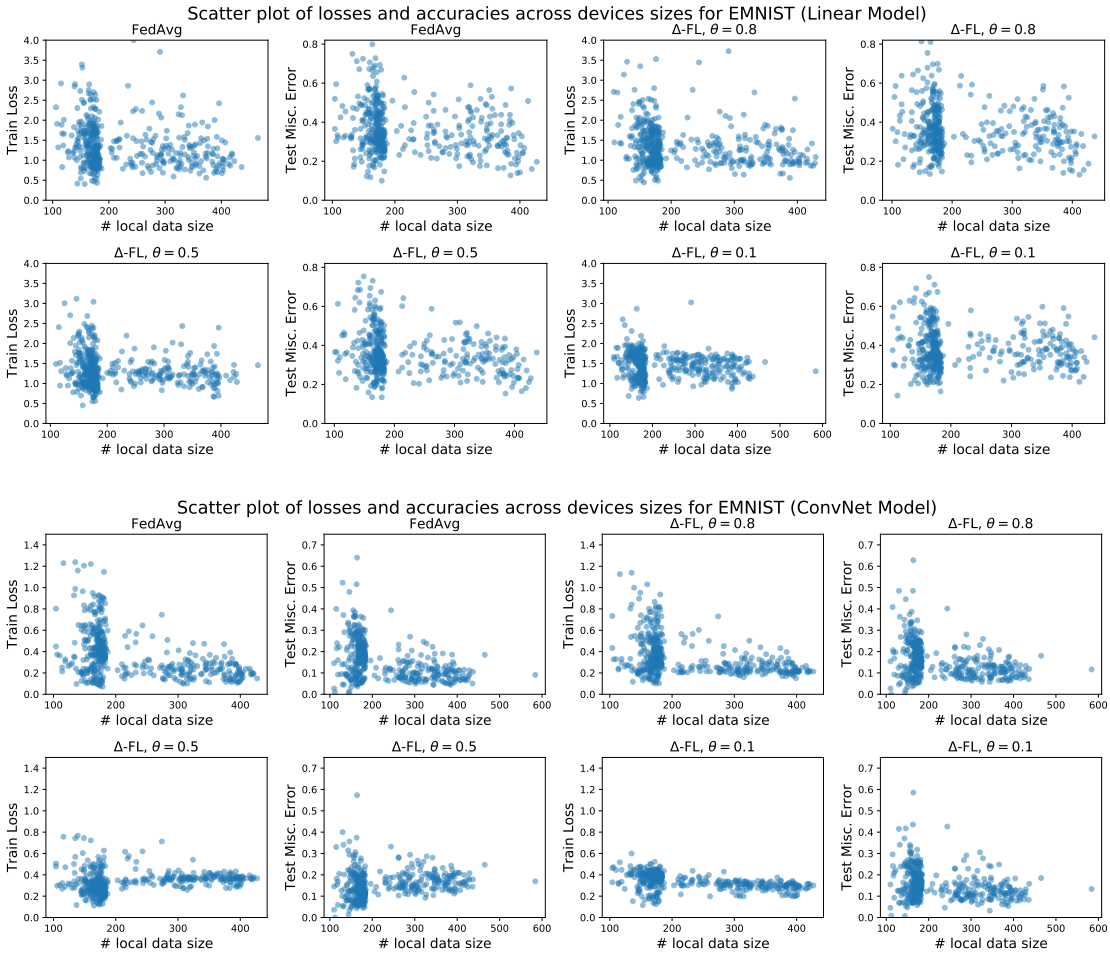


Figure D.3: Scatter plot of (a) loss on training client vs. amount of local data, and (b) misclassification error on testing client vs. amount of local data for EMNIST.

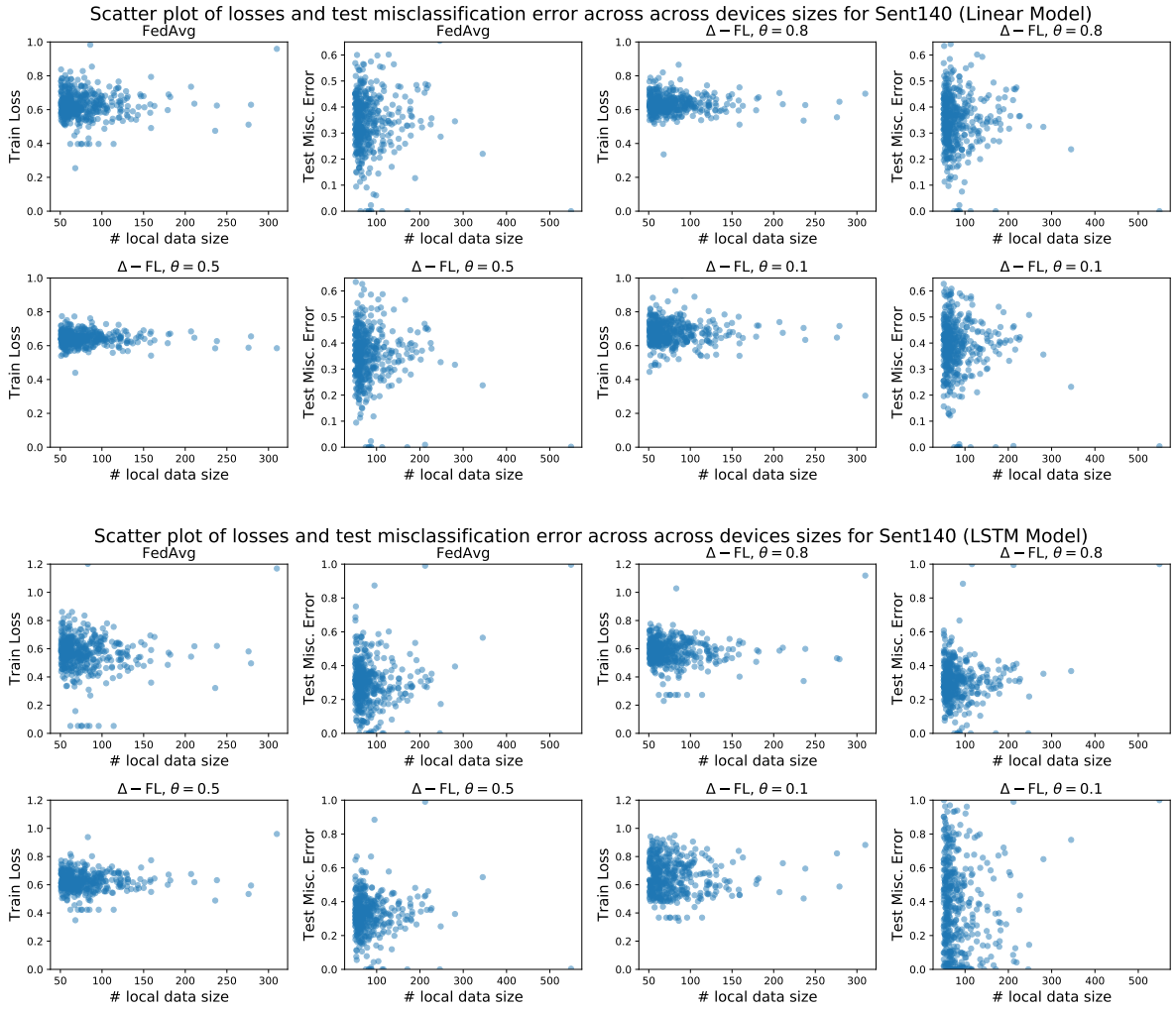


Figure D.4: Scatter plot of (a) loss on training client vs. amount of local data, and (b) misclassification error on testing client vs. amount of local data for Sent140.