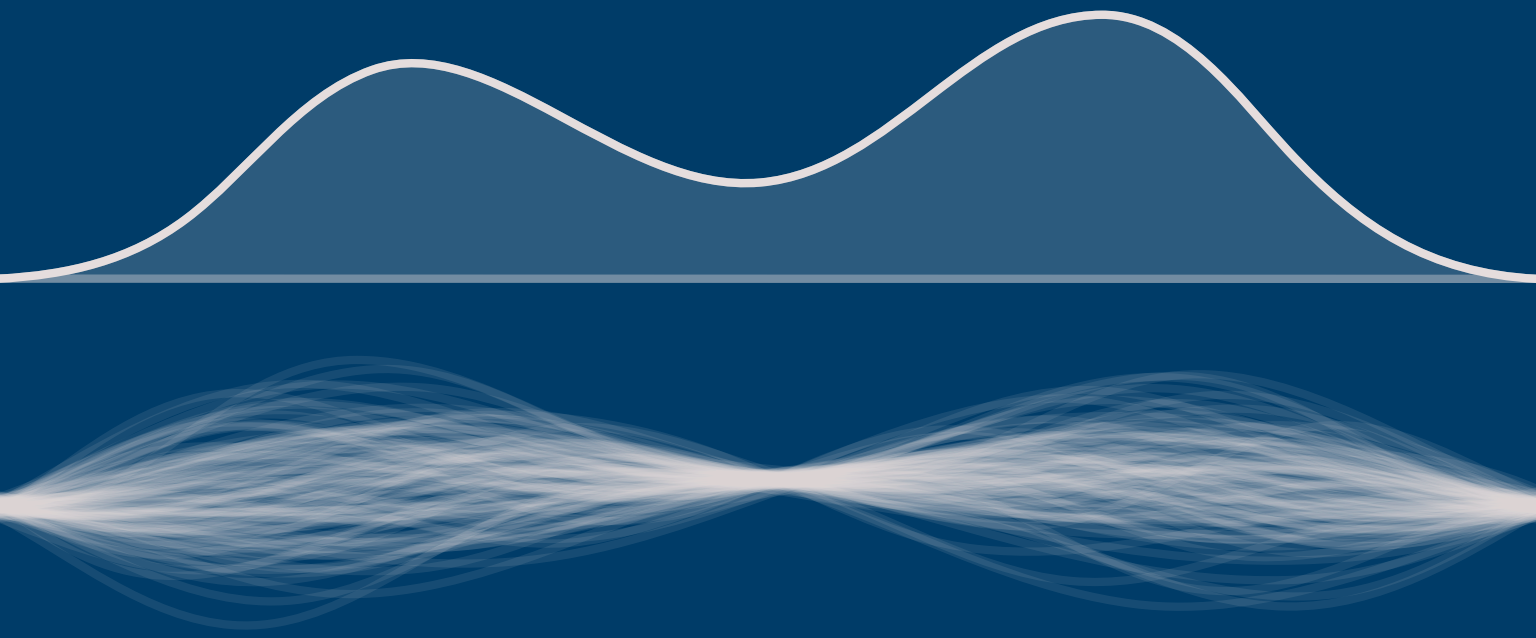
Bayesian Decision-making Algorithms



Alexander Terenin

Alexander Terenin

Bayesian Decision-making Algorithms

Author's draft: August 31st, 2026

Work in progress: currently about 20% complete

Latest version viewable online at: https://bayesianalgorithms.com/

Table of Contents

Preface

This monograph is a work in progress—and an academic experiment on building in public: I am making its content available online as I write it, in both PDF and HTML formats. I'd be delighted to have you follow along, and welcome your feedback. Information on how to contact me can be found on the book's website. There is also a mailing list you can subscribe to in order to learn when major updates are released, which will never be used for any other purpose. If you prefer not to receive email, please see the main website for other options.

Naturally, as this book is a work in progress, it might contain errors—both technical and typographical. If you find one, please contact me immediately by filing a GitHub issue. Effort has and will be made to ensure publicly-visible content has reached a certain level of maturity and can be trusted. The current table of contents contains the plan for the book's content, but at present, many chapters have yet to be written. They are marked with obvious placeholder text that indicates they are *under construction*, and will appear in future drafts.

This book comprises the academic side of what I am currently working on, but it is not my sole focus—or, in some sense, even my primary focus. As such, progress is likely to be uneven and will appear at random times. If you'd like to know about what else is on my mind, please visit my personal website to learn more. And, if you're particularly excited about a planned chapter's contents, please feel free to let me know, so that I have a better sense of what to prioritize.

Once this book reaches a complete draft, I plan to rewrite this preface. The new one will likely become more personal, with reflections on my career—after all, I did make the unconventional decision to leave academia in favor of other ambitions while writing this book, and nonetheless plan to actually get it done.

I originally got interested in Bayesian decision-making because I thought studying it deeply could give clues for how to develop artificial general intelligence—which, in the form of large language models, now exists. Today, I see Bayesian decision-making as a promising way to study how language models actually work, and in particular how they handle exploration. I hope this book inspires the next generation to further connect these research areas. Finally, I thank my wife Kamilė for her love and support, which has made this work possible.

Alexander Terenin
Vilnius, Lithuania, August 2026

Notation

General

$\mathbb{N}$	Natural numbers, not including zero
$\mathbb{Z}$	Integers
$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\mathbb{R}^d$	Euclidean space of dimension d
$\mathbb{R}^X$	Space of functions from X to $\mathbb{R}$
$C(X;\mathbb{R})$	Space of continuous real-valued functions
$[N]$	Finite set of all integers from 1 to N
$\oplus$	Disjoint union of sets, concatenation of sequences
$\mathbb{1}_{(\cdot)}$	Indicator function
$\mathcal{O}(\cdot)$	Asymptotic upper bound
$\Omega(\cdot)$	Asymptotic lower bound
$\Theta(\cdot)$	Asymptotic upper and lower bound

Probability

$(\Omega, \mathcal{F}, \mathbb{P})$	Probability space	61
$\sim$	Distribution of a random variable	61
$\mathbb{E}(\cdot)$	Expectation of a random variable	61
$\mathrm{Var}(\cdot)$	Variance of a random variable	61
$\mathrm{Cov}(\cdot,\cdot)$	Covariance between two random variables	61
$\mathbb{E}(\cdot \mid \cdot)$	Conditional expectation.	61
$\mathcal{M}_1(X)$	Space of probability measures over X.	61
$\mathcal{M}_1(Y \mid X)$	Space of probability kernels over Y given X	61
δ_x	Dirac measure centered at x	61
$\mathrm{N}(\mu, \sigma^2)$	Normal distribution	61
$\mathrm{U}(X)$	Uniform distribution over X	61
$\mathrm{Ber}(p)$	Bernoulli distribution.	61
$\mathrm{Bin}(n, p)$	Binomial distribution.	61
$D_{\mathrm{KL}}(\cdot \mid\mid \cdot)$	Kullback–Leibler divergence	61
$I(\cdot;\cdot)$	Mutual information	61

Markov Decision Processes

Episodic Decision Problems

Bayesian Models and Algorithms

Chapter 1

Introduction

Making decisions in the face of uncertainty is a ubiquitous part of life and intelligent behavior. To illustrate this, let us start with perhaps the most immediate example possible: your experience in deciding to open and read this book. Perhaps you've got a question in mind: for instance, *how does the Thompson sampling algorithm work?* Or, you might've learned about a class of techniques, such as upper confidence bounds, and started to wonder, *what else is possible?* Either way, time is finite, so you will likely have opened this book in pursuit of a goal.

In seeking the answers to these questions, there are different actions you could have chosen to take. The first of these will likely have been to glance at the table of contents—followed, perhaps, by reading this introduction. Or, you might've jumped straight to one of the chapters of interest—in this case, you likely made it to this introduction much later. Was your use of time in pursuing the question of interest effective? From a fundamental perspective, how can we tell?

Uncertainty is a central part of studying questions such as these. Before reading this book, you will not have known its precise contents in full. Thus, understanding what parts to read would have involved gathering information: *is a particular chapter relevant? Should I read it in careful detail?* One can think of the action of skimming a chapter as something like spending a small amount of time to reduce uncertainty about its contents.

Note that uncertainty is a property of your mind's *model* of the book's contents, and not of those contents alone. A data scientist who has implemented bandit algorithms in production systems at a tech company will likely bring a very different profile of uncertainty compared to a student learning about them for the first time. We will reason about such differences through the lens of *Bayesian learning*, applying conditional probability to assess and propagate uncertainty about the quantities of interest.

In essentially all non-trivial cases, a consequence of uncertainty is the emergence of *explore-exploit tradeoffs*—that is, of tradeoffs between gathering new in-

formation, and using the information you've already gathered to achieve your goals. Continuing our example, if you've skimmed one chapter and it looks relevant, should you spend the effort to read it carefully, or first skim the next one, in case it might be even-more-relevant? How does one achieve the right balance between these options? What levels of performance are possible?

This book's goal is to develop a systematic understanding of such questions and how to think about them. We will begin by formalizing the concept of *decision-making under uncertainty*. A crucial part of this will be to consider: *what would you have learned instead if you had taken other actions?* And, closely-related: *how well did you do, compared to what you would have done if you knew everything from the beginning and there was no uncertainty?* We will see that problems can have different difficulties, and study how to tell whether an algorithm works or not.

With a mathematical understanding of the problem structure in place, we will survey and develop an understanding of every major class of decision-making algorithms. The emphasis will be on the *definitions*, not proofs or analysis techniques: for each algorithm class, we will follow the simplest path by which one can conclude that it makes technical sense. In some cases, this will be by first-principles derivation, in others, by positing an algorithm and then analyzing it. We will aim to achieve a comprehensive overview of the algorithmic landscape.

A critical part of this will be to work at the *right level of generality*: we will not be afraid of considering abstract formulations, but will also never dive into abstractions for their own sake, and will back our definitions up with appropriate sets of concrete examples. In doing so, we will not only see that decision-making admits a clean and elegant mathematical picture, but that this picture can provide a precise form to a much broader set of practical phenomena than are otherwise understood to have a central origin.

To this end, we will focus on fundamentals and not on applications, of which there are many. Decision-making, in our general sense, has been discovered and re-discovered in many disciplines, ranging from robotics to psychology, to medicine, to economics, to machine learning and artificial intelligence—our primary domain of interest. The need for a comprehensive and unified way to think about explore-exploit tradeoffs in large language models, and in artificial intelligence more broadly, is a major motivating force behind this book.

In machine learning and artificial intelligence, all of the ideas in this book are known—but not in a unified way that makes their scope of applicability clear. Large language models are conditional distributions over the space of tokens—yet, it is not widely appreciated that Bayesian decision-making provides a clean and natural formalism for studying in-context sample-efficient learning. We hope this book inspires new ways of understanding artificial intelligence systems, and thereby helps ensure they broadly benefit humanity.

With this in mind, let us begin—by casting into mathematics the example of you, the reader, choosing to open this book and read its pages.

Chapter 2

Decision-making Under Uncertainty

We now initiate our technical development. The aim will be to cast the concept of decision-making under uncertainty, intuitively described in Chapter 1, into mathematics. This will allow us to precisely understand what is a *decision-making problem*, what is an *algorithm*, how to assign notions of difficulty to problems, and how to assess performance of a given algorithm. This will set up the language necessary to understand the rest of the book, which studies the most important known classes of decision-making algorithms.

Following the philosophy outlined in Chapter 1, our chief focus will be on the definitions and the concepts they capture. We will therefore first present the general definition, then work through a number of examples in order to understand its scope and character. This will not require a lot of mathematical background beyond an appropriate degree of command over probability theory: for convenience, we provide a short review in Appendix A. In particular, let $\mathcal{M}_1(\cdot)$ be the space of probability distributions over a given set. Let us begin.

2.1. Definitions and Basic Examples

We start with the most important definition in this book.

Episodic Decision Problem

Definition 2.1. *An* EPISODIC DECISION PROBLEM *is defined by the following:*

1. *An* ACTION SPACE A.
2. *A* REWARD FUNCTION CLASS $\mathcal{R} \subseteq \{r : A \to \mathbb{R}\}$.
3. *One of the following forms of* GROUND-TRUTH REWARDS:
   (a) *A* TRUE REWARD FUNCTION $r \in \mathcal{R}$.

(b) A REWARD DISTRIBUTION $q \in \mathcal{M}_1(\mathcal{R})$.

(c) A SEQUENCE OF TRUE REWARD FUNCTIONS $r_t \in \mathcal{R}$ *for* $t = 1, .., T$.

Depending on the chosen option, we say that the respective problems are of the STOCHASTIC, BAYESIAN, *and* OBLIVIOUS ADVERSARIAL *variants.*

4. *An* OBSERVATION SPACE Σ.
5. *A* FEEDBACK FUNCTION $\sigma : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$.

The idea behind this definition—whose non-episodic form, as well as naming, dates at least to Blackwell (1951, 1953)—is to formalize a decision-making process that proceeds over *episodes*, indexed by discrete time points $t \in \mathbb{N}$. Assuming for sake of introduction that we are considering the stochastic variant, this process takes place as follows:

1. At time t, a thinking robot,[1] called the *learner*, chooses an action $a_t \in A$.
2. The learner receives a reward $r(a_t)$, but does not observe this reward.
3. Instead, the learner observes the random feedback $\sigma_t \sim \sigma(r, a_t)$.

At the next time point $t + 1$, the learner may then select a new, potentially random, action a_{t+1}, based on $\mathcal{R}$, as well as $a_1, .., a_t$ and $\sigma_1, .., \sigma_t$, for which we use the shorthand $a_{1:t}$, $\sigma_{1:t}$. Thus, the learner knows the *reward class*, the *actions it chose*, and the *history of observations*, but not the true reward function itself. In fact, it is precisely this lack of knowledge that formalizes the sense in which the learner is making decisions under *uncertainty*.

This defines the protocol according to which the learner interacts. It is not the most general form possible—in particular, it excludes adaptive adversaries that react to the learner's actions, which we will consider in Chapter 7—but we believe it achieves a good balance between being general yet not-too-difficult to define. Let us now consider a set of examples, which will serve to make this otherwise-abstract concept significantly more concrete. We will start with the simplest one—the trivial case in which there is no uncertainty.

Trivial Episodic Decision Problem

Example 2.2. *An episodic decision problem is called* TRIVIAL *if* $\mathcal{R} = \{r\}$ *is a singleton.*

Here, the learner knows the reward function, as the function class is so small that one can infer the true reward—or analog thereof, in the non-stochastic variants—from the function class. Therefore, there is no uncertainty, and all

[1] In his seminal work on the foundations of Bayesian probability, Jaynes (2003) famously introduced the metaphor of a thinking robot, and used it to illustrate how conditional probability can be seen as an extension of Boolean logic which incorporates uncertainty—leading to the framework we call *Bayesian learning*. We conceptualize our thinking robot similarly, but will instead study how it should *resolve* its uncertainty in order to act—in particular, we will want to know how often the robot chooses the right actions, even if its internal model of the world differs from reality. As we proceed, more and more of the moving parts that make up this perspective will be made precise.

the learner needs to do is solve the optimization problem

$$a_t = \arg\max_{a \in A} r(a) \tag{2.1}$$

which they are allowed to do because they possess sufficient information to infer $r \in \mathcal{R}$ from knowledge of $\mathcal{R}$ alone. By doing so, the learner can achieve the best possible reward irrespective of the structure of Σ or σ, without the need to explore different actions or learn anything from the data they observe. A related form of triviality also occurs if $|A| = 1$: without distinct actions, there is nothing to learn and no need for exploration. To see these phenomena, we consider the next example.

Example 2.3. *An episodic decision problem is called a* STOCHASTIC MULTI-ARMED BANDIT *with $K \in \mathbb{N}$ ARMS and standard Gaussian noise if:* Multi-armed Bandit

1. *The action space is $A = [K]$.*
2. *The reward function class is $\mathcal{R} = \{r : A \to [-1, 1]\}$.*
3. *We are considering the stochastic variant.*
4. *The observation space is $\Sigma = \mathbb{R}$.*
5. *The feedback function is $\sigma(r, a) = r(a) + \varepsilon(a)$, where $\varepsilon \sim \mathrm{N}(0, 1)$.*

Here, we have implicitly associated the random function $\sigma : \Omega \times \mathcal{R} \times A \to \Sigma$, where Ω is a probability space, with its distribution, which we also denote by $\sigma : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$, thus ensuring the definition makes sense. See Appendix A for more on the conventions needed to ensure this notation is unambiguous.

The name *multi-armed bandit* comes from a casino analogy: the idea is that the learner is presented with a collection of slot machines—known in old times as *bandits* for their tendency to rob casino patrons of their wealth. Each machine has a different reward distribution. At a given round, the learner may choose to play one of them—that is, to *pull an arm*—and observe the reward it produces. The learner's aim, intuitively, is to maximize the total rewards they receive.

To do so, the learner must try different arms, and learn which ones perform best by trial and error. Neither the greedy strategy of picking the arm with the best empirical mean according to the history, nor the purely-exploratory strategy of playing arms uniformly at random, are good strategies—at least, according to performance criteria we will introduce shortly in the sequel. Instead, the learner must balance explore with exploit: they must keep trying different arms, but should do so less and less often as information accumulates.

This balance—between explore and exploit—is present in almost all episodic decision problems, but the precise character through which it shows up can differ significantly. Bandits are perhaps the most-important and best-studied problem class: their explore-exploit tradeoffs, and algorithms for balancing them, are well-understood. Indeed, it is often a good idea to first understand

how a given setting works by considering its bandit analog, before examining it more generally: we will adopt this approach throughout this book.

There is absolutely nothing which stops us from defining bandits in substantially greater generality.

Bandit

Example 2.4. *An episodic decision problem is called a* BANDIT *if:*

1. *The observation space is* $\Sigma = \mathbb{R}$.
2. *The feedback function is* $\sigma(r, a) = r(a) + \varepsilon(a)$, *where* ε, *called the* NOISE, *is a random variable satisfying* $\mathbb{E}\,\varepsilon(a) = 0$.

Here, the action space A and reward function class $\mathcal{R}$ are allowed to be general. Moreover, it makes sense to consider stochastic, Bayesian, and adversarial variants—in the first and second case, often with Gaussian or sub-Gaussian noise—and in the third case, usually with $\varepsilon = 0$ deterministically, which the definition allows. Only the structure of the feedback function—namely, *play an action, learn about that specific action*—is assumed as part of the definition.

We can contrast bandits with an episodic decision problem with a simpler feedback structure, given as follows.

Stochastic Online Learning

Example 2.5. *An episodic decision problem is called a* STOCHASTIC ONLINE LEARNING *problem with* $N \in \mathbb{N}$ ACTIONS *and Gaussian noise if:*

1. *The action space is* $A = [N]$.
2. *The reward function class is* $\mathcal{R} = \{r : A \to [-1, 1]\}$.
3. *We are considering the stochastic variant.*
4. *The observation space is* $\Sigma = \mathbb{R}^N$.
5. *The feedback function is* $\sigma(r, a) = r + \varepsilon$, *where* $\varepsilon_n \sim \mathrm{N}(0, 1)$ *for* $n \in [N]$.

Here, unlike in a bandit, the learner observes a noisy version of the reward of *all* arms, not just the ones they played. This means the learner no longer has to explore—at least, as long as the problem variant in question is not adversarial. In Chapter 7, we will see that the adversarial variant of this problem—where, as before, one would typically consider $\varepsilon = 0$ deterministically—is surprisingly rich, and provides a good idealized setting within which one can study randomized exploration from a fundamental point of view.

Full Feedback

Example 2.6. *An episodic decision problem is called a general-action* ONLINE LEARNING *problem, or is said to have* FULL FEEDBACK, *if:*

1. *The observation space is* $\Sigma = \mathbb{R}^A$.
2. *The feedback function is* $\sigma(r, a) = r + \varepsilon$, *where* $\varepsilon : A \to \mathbb{R}$, *called the* NOISE, *is a random function satisfying* $\mathbb{E}\,\varepsilon(a) = 0$ *for all* a.

There are many possible episodic decision problems which sit in-between full feedback and bandit feedback—that is, ones where the learner observes the reward of the action they take, potentially up to noise, along with additional side information that the learner can benefit from. These are as follows.

Decision-making with Structured Observations

Example 2.7. *An episodic decision problem is said to have* STRONGER-THAN-BANDIT FEEDBACK*, if:*

1. *The observation space is $\Sigma = O \times \mathbb{R}$ for some* OBSERVATION SPACE *O.*
2. *The feedback function is $\sigma(r, a) = (o(r, a), r(a) + \varepsilon(a))$ for a function $o : \mathcal{R} \times A \to O$, where we have $\mathbb{E}\,\varepsilon(a) = 0$.*

If, additionally, A is a finite set, it is called DECISION-MAKING WITH STRUCTURED OBSERVATIONS*.*

This class is notable because significant work has gone into understanding its theoretical structure, and we know a fair bit about the fundamental quantities—called *decision-estimation coefficients*—that determine problem difficulty, in spite of its rather general character. We refer the interested reader to Foster and Rakhlin (2023), which provides a comprehensive treatment of these ideas.

Decision-making with structured observations is not the only such class: one can also say a surprising amount about problems where A, Σ, and $\mathcal{R}$ are finite.

Partial Monitoring Game

Example 2.8. *An episodic decision problem is said to be a* PARTIAL MONITORING GAME *if:*

1. *The action space A is a finite set.*
2. *The reward function class $\mathcal{R}$ is a finite set.*
3. *We are considering the adversarial variant.*
4. *The observation space Σ is a finite set.*

The stochastic and Bayesian variants are also known as *stochastic partial monitoring games* and *Bayesian partial monitoring games*, respectively. This is somewhat of a misnomer, particularly for the Bayesian variant, which corresponds to a stochastic control problem rather than a bona-fide game with more than one player. This is one motivation behind our choice of the name *episodic decision problem*,[2] following Blackwell (1951, 1953), to describe the general case.

A remarkable fact about this problem class is that, in a sense we will soon precisely define, there are only four possible problem difficulties: *trivially easy*, *bandit-like*, *harder-than-bandit*, and *impossible*. This is in spite of the fact that—except for finiteness of A, Σ, and $\mathcal{R}$—the class is completely general! Comparatively less is known about algorithms for this problem class.

[2]This name should also never be confused with the notion of a *decision problem* in the sense of Turing machines or decidability, which is completely unrelated and will never arise in this book.

We now turn to examples of a different kind—namely, problems that capture the structure of numerical algorithms used for various purposes in the mathematical and computer sciences.

First-order Convex Optimization

Example 2.9. *An episodic decision problem is said to be a* CONVEX OPTIMIZATION PROBLEM WITH A FIRST-ORDER ORACLE *if:*

1. *The action space $A \subseteq \mathbb{R}^d$ is a convex set.*
2. *The reward function class $\mathcal{R} = -\operatorname{Cvx}(A)$, namely the space of all (negations of) convex functions over A.*
3. *We are considering the stochastic variant.*
4. *The observation space is $\Sigma = \mathbb{R}^{d+1}$.*
5. *The feedback function is $\sigma(r, a) = (r(a), \nabla r(a))$.*

In this example, the learner's aim is to minimize a convex function. They are allowed to evaluate it at every time point, and observe its pointwise value and gradient (in the subdifferential sense). Note that the term *stochastic variant* here is somewhat of a misnomer, as observations are noiseless. Note also that $\mathcal{R}$ includes *all* convex functions: one could modify this to include only α-strongly-convex and β-smooth functions, if desired. Thus, we see that episodic decision problems include many forms of convex optimization as a special case.

This example should immediately evoke the level of generality that episodic decision problems capture: it is clear that there is nothing stopping one from similarly defining other kinds of optimization problems such as black-box global optimization, Lipschitz optimization, or more general kinds of numerical problems such as computation of Nash equilibria and related notions.

Among such problems, in this book, we will focus on the class of *black-box optimization*—where we note that black-box optimization problems can also be seen as bandits where $A = [0, 1]^d$ and $\mathcal{R}$ consists of continuous functions with an additional prescribed degree of smoothness. There is one more problem class which is very much worthy of mention in our overview.

Episodic Reinforcement Learning

Example 2.10. *Let $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$ be an infinite-horizon discounted Markov decision process with initial state $s_0 \in \mathcal{S}$, and let $\mathcal{R}'$ be a class of reward functions for the Markov decision process, with $r \in \mathcal{R}'$. An episodic decision problem is said to be an* EPISODIC REINFORCEMENT LEARNING PROBLEM *if:*

1. *The action space is taken to be the space of policies, namely $\Pi = \{\pi : \mathcal{S} \to \mathcal{A}\}$, where this symbol is used in place of A to clarify notation.*
2. *The reward function class is the space of all value functions which occur under $\mathcal{R}'$, namely $\mathcal{R} = \{\pi \mapsto V^{(\pi)}(s_0; r') : r' \in \mathcal{R}'\}$.*
3. *We are considering the stochastic variant, and the true reward function is taken to be r, or more precisely, the value function $V^{(\cdot)}(s_0; r)$.*

4. *The observation space is the space of all state-action-reward sequences, namely* $\Sigma = \{(s_{(\cdot)}, a_{(\cdot)}, r_{(\cdot)}) : \mathbb{N} \to \mathcal{S} \times \mathcal{A} \times \mathbb{R}\}$.

5. *The feedback function is* $\sigma(r', \pi) = (s_\tau, \pi(s_\tau), r'(s_\tau, \pi(s_\tau)))_{\tau=0}^{\infty}$, *where the rollout is random with distribution determined by* p, *and we implicitly identify each value function with a reward function producing it.*

There are many possible variants on this definition. In this one, the idea is that, at each episode, we try out a new policy and observe a rollout under that policy. Our algorithmic aim is to find the optimal policy. This definition therefore formalizes policy learning by trial and error, given known dynamics. One can easily consider formulations where the dynamics are not known and must be learned, by including them in $\mathcal{R}$. Stochastic contextual bandits also constitute a variant: these have a time horizon of one, and a random initial state.

This shows that episodic decision-making is general-enough to include episodic Markov decision processes as a special case. As consequence, if we had a comprehensive understanding of how to construct strong algorithms in complete generality, this understanding would tell us how to construct sample-efficient reinforcement learning algorithms with the optimal degree of exploration. Thus, episodic decision-making provides a candidate paradigm for understanding exploration in the context of artificial intelligence.

By now, we hope the showcased examples have provided a concrete grounding to the abstract concept introduced as Definition 2.1. We also hope they have made a self-evident case for how many interesting phenomena throughout various technical disciplines can be viewed in a unified way using the introduced mathematical language. We encourage you—that is, the reader—to pick out a special case of particular interest to think about, as you continue.

This book, however, is about *algorithms* rather than problems, and so we will conclude our examples here. In particular, it is about algorithms that quantify uncertainty—which arises through knowledge of only the function class $\mathcal{R}$ rather than the true reward function r—through probabilistic models, with learning performed using Bayes' Rule. To understand this, we proceed to understand what constitutes an algorithm, what kind of algorithms are Bayesian, and how to assess an algorithm's performance.

2.2. Algorithms for Decision-making

We now define the notion of an *algorithm*, previously used informally during our survey of episodic decision problems. Let $\text{Seq}(\cdot)$ be the space of finite-length sequences, defined over an underlying set.

Definition 2.11. *An* ALGORITHM *for solving an episodic decision problem is a function* $p : \text{Seq}(A \times \Sigma) \to \mathcal{M}_1(A)$. *We refer to the space of algorithms as* $\mathcal{P}$. Decision-making Algorithm

Algorithms are also sometimes called *policies* or *strategies*, and we may use these terms interchangeably, unless this would become ambiguous in the given context. The simplest possible algorithm one should generally consider consists of playing actions uniformly at random, without any learning.

Random Search

Definition 2.12. *Suppose A is either a finite set, or a sufficiently well-behaved compact subset of $\mathbb{R}^d$. The* RANDOM SEARCH *algorithm plays actions uniformly at random at all times t, that is*

$$a_t \sim \mathrm{U}(A). \tag{2.2}$$

It is clear that random search will not be a strong algorithm for many problems, as it behaves in an anti-greedy, purely-exploratory fashion, and doesn't learn anything from the data. We will shortly make this claim precise. At the same time, random search is often empirically stronger than one expects: in black-box optimization—especially if the objective's domain is high-dimensional—experience has shown that it can take a surprising amount of work to construct algorithms which are stronger in practice.

In formulating an algorithm, recall that the learner is allowed to use the reward function class $\mathcal{R}$ and feedback function σ, but not the true reward r. We now take the first step to introducing the class of algorithms this book is named after. Let $\mathcal{M}_1(\cdot \mid \cdot)$ be the space of all conditional distributions—formally, the space of probability kernels—over a given pair of sets.

Bayesian Model

Definition 2.13. *A* BAYESIAN MODEL *is defined by the following:*

1. *Let $p_r \in \mathcal{M}_1(\mathcal{R})$ be the* PRIOR.
2. *Let $p_{s|r,a} \in \mathcal{M}_1(\Sigma \mid \mathcal{R} \times A)$ be the* LIKELIHOOD.

For any t and any dataset $a_{1:t}$, $\sigma_{1:t}$, we denote the respective POSTERIOR DISTRIBUTION *under the model by $p_{r|a_{1:t},\sigma_{1:t}} \in \mathcal{M}_1(\mathcal{R} \mid \mathrm{Seq}(A \times \Sigma))$.*

A Bayesian model describes a general way by which the learner quantifies uncertainty about the unknown quantity of interest—that is, in an episodic decision problem, the unknown reward function. In adversarial settings, we allow ourselves to also consider Bayesian models defined over sequences of reward functions—where each time step has its own prior, and its own likelihood—but omit this here to ease presentation.

Throughout this book, we assume that A, $\mathcal{R}$, and Σ are regular enough to ensure that the necessary posterior distributions exist for all datasets, and defer mathematical questions of this kind to further study on a model-specific basis. In cases where the respective spaces are finite sets—or even (subsets of) $\mathbb{R}^d$—questions of existence tend to not be difficult. But, even more importantly, the mathematics behind them is essentially never of a decision-theoretic character. Thus, in doing so, we hand-wave in exactly the cases where this is safe to do.

By itself, a Bayesian model only describes *how to learn*. It does *not* describe *what to do* with what has been learned—how to use the obtained representation of the learner's uncertainty to select future actions. This additional ingredient must be specified in order to obtain an actual algorithm.

Bayesian Algorithm

Definition 2.14. *A* BAYESIAN DECISION-MAKING ALGORITHM *is a pair consisting of a Bayesian model and a potentially-randomized* DECISION RULE

$$\delta : \mathcal{M}_1(\mathcal{R}) \to \mathcal{M}_1(A). \tag{2.3}$$

Thus, the extra ingredient which is part of a Bayesian algorithm is a rule by which one transforms distributions over unknown reward functions into distributions over actual actions. This rule tells the learner how to translate what they know into what they do. We say a decision rule is *deterministic* if its image consists of Dirac measures. Additionally, we note that one can also consider time-varying Bayesian algorithms, but omit this here to ease notation.

It is extremely important to note that Bayesian algorithms can be used on non-Bayesian problems—so much so that we will state it with emphasis:

($\star$) *A Bayesian algorithm constitutes a valid algorithm for* ANY *episodic decision problem variant, including stochastic and adversarial ones.*

Thus, throughout this book, we will find ourselves thinking about Bayesian algorithms for non-Bayesian problems—this will constitute a form of *model-mismatch*, meaning that the model differs from reality. We will return to this concept in the sequel, once we have developed an understanding of how to evaluate models under different forms of reality. The vast majority of Bayesian algorithms in current use come from the following class.

Acquisition Function

Definition 2.15. *A Bayesian decision-making algorithm is said to be based on an* ACQUISITION FUNCTION $\alpha_t : A \to \mathbb{R}$*, which we potentially allow to be random, if, for every posterior distribution, its decision rule takes the form*

$$\delta(p_{r|a_{1:t},\sigma_{1:t}}) = \arg\max_{a \in A} \alpha_t(a). \tag{2.4}$$

Algorithms based on acquisition functions, therefore, are those for which the Bayesian model's posterior distribution $p_{r|a_{1:t},\sigma_{1:t}}$ is used to form a function α_t which ranks the actions according to which is the most-promising. The algorithm proceeds by choosing the highest-ranking action. We allow for both deterministic and randomized *tie-breaking rules*, but suppress them from notation unless their role is critical to the situation at hand. Every Bayesian algorithm class considered in this book will turn out to have the presented form.

2.3. Evaluating Performance via Regret

Having formalized the notion of a decision-making algorithm, and introduced the class of algorithms we will study, we now turn to the question of what it means for an algorithm to *perform well*, starting from the stochastic variant where there is a single ground-truth reward function r.

Regret

Definition 2.16. *The* REGRET *of an algorithm p is defined as*

$$R_T(p, r) = \mathop{\mathbb{E}}_{a_t \sim p} \sup_{a \in A} \sum_{t=1}^{T} r(a) - r(a_t). \tag{2.5}$$

Regret, therefore, is the difference between how well the algorithm actually did, and how well it *could have done* if it knew what the ground-truth reward actually was. It is always non-negative, and minimizing regret is equivalent to maximizing rewards. Our rewards are assumed non-stochastic: in situations where stochastic rewards are of central interest, this definition is sometimes called *pseudo-regret*. We choose this notation in particular to emphasize that it depends critically on three key quantities:

1. The learner's strategy p.
2. The true reward function r.
3. The time horizon T.

This book's central questions will focus on how to construct strong algorithms. In most cases, we will take this to mean algorithms that perform well no matter what the true reward function actually is, as long as it is contained in the reward function class $\mathcal{R}$. Ideally, we will want this to hold for any time horizon—but it will be interesting to study how behavior should change according to different horizons. This will allow us, for example, to make precise the intuitive idea that, if one has more time to learn, they should explore more.

High-probability vs. Expected Regret

The above notion of regret is formulated in expectation. It is also possible to study *random regret*—that is, the random variable inside the expectation above. In doing so, it is typical to consider *high-probability* behavior: this can be used, for instance, to show that an algorithm works well with probability $1-\varepsilon$, not just on average. The remaining ε-probability might correspond to extremely rare cases—for instance, hopeless situations where a large set of Gaussian draws are all simultaneously positive, and the data is completely misleading.

The notion presented above is sometimes called *cumulative regret*, since it counts all rewards at all time points. This is not the only reasonable choice: one can just as well restrict attention to the reward obtained at the final time point.

Simple Regret

Definition 2.17. *The* SIMPLE REGRET *of an algorithm p is defined as*

$$r_T(p, r) = \mathop{\mathbb{E}}_{a_T \sim p} \sup_{a \in A} r(a) - r(a_T). \tag{2.6}$$

Compared to cumulative regret, this notion effectively disregards the suboptimality of the points the algorithm chooses to explore. We will see that its overall behavior is similar—but, for a given algorithm class, it can be much more convenient to work with either simple regret, or cumulative regret, depending on the situation.

Both of the preceding notions are finite-horizon notions, in the sense that they are defined for a given $T \in \mathbb{N}$. It can be just as natural to instead work with an infinite-horizon discounted formulation, as follows.

Definition 2.18. *Let $\gamma < 1$. The* DISCOUNTED CUMULATIVE REGRET *of an algorithm p is defined as* Discounted Cumulative Regret

$$R_\gamma(p, r) = \mathop{\mathbb{E}}_{a_t \sim p} \sup_{a \in A} \sum_{t=1}^{\infty} \gamma^{t-1}(r(a) - r(a_t)). \tag{2.7}$$

One advantage of a discounted formulation is that it is stationary with respect to time. On the other hand, a disadvantage is that discount factors can be slightly less intuitive to think about than time horizons. One way to reconcile this is to view discounting as finite-horizon with a fixed termination probability, and consider regret in expectation. Note that, for bounded rewards—concretely, those with $|r| \leq 1$—discounted cumulative rewards are uniformly bounded by

$$\sum_{t=1}^{\infty} \gamma^{t-1} r(a_t) \leq \frac{1}{1-\gamma}. \tag{2.8}$$

This is related to the finite-horizon bound

$$\sum_{t=1}^{T} r(a_t) \leq T. \tag{2.9}$$

A clean way, therefore, to think about discounting is that it gives rise to an *effective time horizon*

$$T_\gamma = \frac{1}{1-\gamma} \tag{2.10}$$

arising from the geometric series formula. For almost all of this book, we will work with undiscounted formulations, but it is not difficult to develop discounted analogs of the ideas presented.

A second way to introduce stationarity with respect to time is to *allow the algorithm to choose T*. To facilitate non-trivial behavior, we introduce a strictly positive cost function $c : A \to \mathbb{R}$. We can then define the following notion.

Definition 2.19. *The* COST-ADJUSTED SIMPLE REGRET *of an algorithm p is defined as* Cost-adjusted Simple Regret

$$r_c(p, r) = \mathop{\mathbb{E}}_{a_t \sim p} \sup_{a \in A} r(a) - r(a_T) + \sum_{t=1}^{T} c(a_t). \tag{2.11}$$

For this to be well-defined, together with actions, the algorithm must also output a binary variable which determines whether to continue to the next episode, or to stop: we refer to this as the *stopping policy*, and to the actual action a_t as the *steering policy*, in contexts where the distinction applies. We require the stopping policy to be defined with respect to the same history of observations as the steering policy, and implicitly extend the notion of a Bayesian algorithm to also cover this case. This formulation is called the *cost-per-sample* problem.

These notions can be extended to allow for a cost function class $\mathcal{C}$ which plays a similar role to $\mathcal{R}$—here, we work with a fixed cost function to ease notation. We use simple rather than cumulative regret to ensure stopping immediately is not optimal, but other formulations are also possible. In general, one can mix-and-match the definitions of this chapter as needed, in essentially all cases where it makes sense to do so. We now turn to regret for non-stochastic variants.

Bayesian Regret

Definition 2.20. *The* BAYESIAN REGRET *of an algorithm p is defined as*

$$R_T(p, q) = \mathop{\mathbb{E}}_{r\sim q} R_T(p, r) = \mathop{\mathbb{E}}_{\substack{a_t\sim p\\ r\sim q}} \sup_{a\in A} \sum_{t=1}^{T} r(a) - r(a_t). \tag{2.12}$$

Model-match vs. Model-mismatch

Thus, Bayesian regret is simply regret averaged over the reward functions sampled from the reward distribution q. For a Bayesian algorithm p, if the prior p_r matches the true reward distribution q, and the likelihood $p_{s|r,a}$ matches the distribution induced by the feedback function σ, we say that we are in the *model-matched* setting. Otherwise, we say there is *model mismatch*: the Bayesian model's assumptions differ from the exact nature of ground-truth reality. Both settings lead to a rich and interesting theory, each with their own specifics.

We have used the term *Bayesian* to refer to several distinct concepts: (i) episodic decision problems of the *Bayesian variant*, (ii) *Bayesian algorithms* for general episodic decision problems, and (iii) the above notion of *Bayesian regret*. When the first and third concepts are combined, something very special happens: the formulation gives rise to a Markov decision process.

Underlying MDP

Definition 2.21. *For an episodic decision problem of the Bayesian variant, define its* UNDERLYING MARKOV DECISION PROCESS *by:*

1. *States: $S_{\text{MDP}} = \text{Seq}(A \times \Sigma)$, with initial state $s_0 = \varnothing$.*
2. *Actions: $A_{\text{MDP}} = A$.*
3. *Transition kernel: given an action a_t, we have $s_{t+1} = s_t \oplus (a_t, \sigma_t)$, where $\sigma_t \sim \sigma(r, a_t)$ with $r \sim q(\cdot \mid s_t)$ drawn from the respective posterior, and $\oplus$ denotes the operation of appending an element to a finite sequence.*
4. *Rewards: at each round, $r_{\text{MDP}}(s, a) = r(a)$, where $r \sim q(\cdot \mid s)$.*

Here, states correspond to the data gathered so far by the algorithm. In turn, algorithms for the episodic decision problem are equivalent to policies for its

underlying Markov decision process. Moreover, the value function is equivalent to Bayesian regret up to a negation and constant shift, meaning

$$V_{\mathrm{MDP}}^{(p)}(s_0) = C - R_T(p, q) \tag{2.13}$$

where $C = \mathbb{E}_{r\sim q} \sup_{a\in A} \sum_{t=1}^{T} r(a)$ depends only on q and not on p, and is viewed as a prior-dependent constant. This statement extends from the initial state s_0 to general states s, as long as the Bayesian regret term is understood to be conditioned on the corresponding history: in doing so, C continues to be a constant, in the sense that it depends on s but not on p.

Bayesian-optimal Algorithm

This correspondence—between episodic decision problems and Markov decision processes—is unique to the Bayesian setting. It has significant consequences, including the existence of an optimal value function V_{MDP}^{*}, and an optimal policy whose respective algorithm we call the *Bayesian-optimal algorithm*. Other kinds of episodic decision problems correspond to *families of Markov decision processes* with different reward functions. This technical difference should not be understated, and its themes will recur throughout this book's chapters.

One rather powerful consequence of having a Markov decision process formalism is that it allows us to check whether basic properties that one would intuitively expect to hold are actually true—if they were not, it would be evidence our definitions were poorly chosen. For instance, consider two episodic decision problems which are identical except for the feedback functions σ_1 and σ_2, where the latter is *strictly more noisy*—meaning, for example, that given the same history and actions the observed feedback satisfies

$$\sigma_t^{(2)} = \sigma_t^{(1)} + \varepsilon \tag{2.14}$$

in distribution, where $\varepsilon \sim \mathrm{N}(0, 1)$. Then, we would expect learning to be more difficult, and the Bayesian-optimal algorithm to achieve less reward in expectation under σ_2. One would expect the same if σ_1 represents full feedback, and σ_2 represents bandit feedback, both with the same noise distribution for each action. Not only are these comparisons true—but, when reformulated in a general rather than specific manner, the implication actually goes in *both* directions.

To see this, we will need a little bit of setup, because it does not suffice to directly compare σ_1 with σ_2. Instead, we need to associate each with the class of *all other analogous feedback functions* σ_1' and σ_2' which, in some sense, carry the same information—including ones that occur under different episodic decision problems. Defining what it means to be analogous is tricky, because a given feedback function $\sigma : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ includes $\mathcal{R}$ in its definition.

To handle this, the rough idea is to start with σ, and replace each reward $r \in \mathcal{R}$ by some $r' \in \mathcal{R}'$: this completely changes the rewards, but preserves feedback structure, since the rest of σ is kept the same. These rewards r' can potentially include new actions $a' \notin A$, as long as they are assumed to be non-informative.

Definition 2.22. *Let $\sigma : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ be a feedback function. A feedback function $\sigma' : \mathcal{R}' \times A' \to \mathcal{M}_1(\Sigma')$ is said to be* ANALOGOUS TO σ *if:*

1. *We have $A \subseteq A'$.*
2. *We have $\Sigma' = \Sigma \oplus \{\boxtimes\}$, where $\oplus$ denotes the disjoint union of sets, and the symbol $\boxtimes$ represents no additional information.*
3. *There is a $\rho : \mathcal{R}' \to \mathcal{R}$ such that $\sigma(\rho(r'), a) = \sigma'(r', a)$ for all $a \in A$.*
4. *We have $\sigma'(r', a') = \delta_{\boxtimes}$ for all $r' \in \mathcal{R}'$ and $a' \in A' \setminus A$.*

We say that a pair of feedback functions σ'_1, σ'_2 is MUTUALLY-ANALOGOUS *to σ_1, σ_2 if analogousness holds individually under a shared function ρ.*

We will also need a suitable notion of what it means to add noise. For a probability kernel $g \in \mathcal{M}_1(\Sigma \mid \Sigma \times A)$, define its action on measures $\mu \in \mathcal{M}_1(\Sigma)$ by integration in the standard manner, namely $g(\mu, a) = \int_\Sigma g(\cdot \mid s, a) \, \mathrm{d}\mu(s) \in \mathcal{M}_1(\Sigma)$ for all $a \in A$. This generalizes the previously-seen example of adding Gaussian noise, by allowing the noise distribution to be arbitrary and action-dependent.

We will also consider a third way to compare feedback functions. For an episodic decision problem and an algorithm p, define λ_p to be the function that maps rewards r to the distribution of actions played during a rollout—with actions generated in the environment where r is ground-truth. Using this, define the set of all *feasible action-sequence distributions* $\Lambda = \{\lambda_p : \mathcal{R} \to \mathcal{M}_1(\mathrm{Seq}(A)) : p \in \mathcal{P}\}$. Observe that this set differs for different σ: the more feedback is given, the more ways an algorithm can change its behavior using that feedback.

With these concepts at hand, we are finally ready to state the key result.

Blackwell's Informativeness Theorem

Theorem 2.23. *Let $\sigma_1 : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ and $\sigma_2 : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ be two feedback functions, where A and Σ are assumed finite. Then the following are equivalent:*

1. *For any two episodic decision problems of the Bayesian variant whose feedback functions are mutually-analogous to σ_1 and σ_2, and where all other components are defined identically—including the priors and time horizons—the respective value functions satisfy $V^*_{\mathrm{MDP},2}(s_0) \leq V^*_{\mathrm{MDP},1}(s_0)$.*
2. *There is a* GARBLING *$g \in \mathcal{M}_1(\Sigma \mid \Sigma \times A)$ such that $\sigma_2(r, a) = g(\sigma_1(r, a), a)$ for all $r \in \mathcal{R}$ and $a \in A$.*
3. *Under the same quantifiers as in the first point, the set of feasible action-sequence distributions is larger under σ_1 than σ_2, in the sense $\Lambda_2 \subseteq \Lambda_1$.*

As consequence, one can define the BLACKWELL ORDER *$\preceq$ over feedback functions, which forms a partial order, up to an equivalence.*

This result tells us that there are three ways of thinking about informativeness of feedback functions: (1) with more information, Bayesian-optimal algorithms achieve more reward in expectation, (2) more-informative feedback functions contain less noise, and (3) under a more-informative feedback function, algorithms can choose a wider range of possible distributions over actions. We view

the fact that all three are actually equivalent as a strong piece of evidence that our definitions are the right ones and lead to a rich mathematical theory.

We defer the proof to Section 2.7, noting briefly that: (a) quantifying over families of episodic decision problems is necessary for equivalence as opposed to implication in one direction, (b) finiteness is not essential, and is assumed only to focus attention on decision-theoretic rather than regularity aspects, and (c) there are extensions of this result to other settings, beyond our model-matched Bayesian setting, for example through the concept of *Le Cam deficiency*—see Le Cam (1986, Ch. 2, Sec. 3, Theorem 2) for a non-sequential variant.

We now briefly examine adversarial problems. Here, there is a major subtlety that one must wrangle with: we do not have a single ground-truth reward, not even a randomly-sampled one. In all cases before, we compared ourselves with the optimal point, but for a sequence there is no single optimal point. What should one compare themselves to? The idea will be to introduce a *single* action a with respect to which to compare the algorithm's performance.

Regret Relative to a Comparator Action

Definition 2.24. *For an adversarial episodic decision problem, the* REGRET *of an algorithm p relative to a* COMPARATOR ACTION *$a \in A$ is defined as*

$$R_a(p, r_1, .., r_T) = \mathop{\mathbb{E}}_{a_t \sim p} \sum_{t=1}^{T} r_t(a) - r_t(a_t). \tag{2.15}$$

Best Action in Hindsight

In particular, the regret against the *best action in hindsight* is defined as

$$R_T(p, r_1, .., r_T) = \mathop{\mathbb{E}}_{a_t \sim p} \sup_{a \in A} \sum_{t=1}^{T} r_t(a) - r_t(a_t) \tag{2.16}$$

where we emphasize that we are comparing $\sum_{t=1}^{T} r_t(a_t)$ with the *supremum of the sum*, and not the sum of suprema over individual functions.

Why this notion in particular? At first, it may look rather idiosyncratic. On the face, it is not obvious whether it ought to exhibit any kind of interesting behavior. It will turn out that the seemingly-more-obvious notion, involving the aforementioned sum of suprema, is an impossibly difficult benchmark and does not give rise to non-trivial algorithms. On the other hand, the above notion, as we will see in Chapter 7, does: it will turn out to provide a natural setting for studying exploration by random actions.

More generally, one can think beyond comparator points, and also consider comparator sequences with various constraints, as well as even-more-general notions. For some of these, note that regret can be negative. As these notions are more advanced, we defer them for the moment: the critical thing to understand, for now, is simply that comparing an algorithm's performance with strategies that are not necessarily optimal is possible, and is often a very useful way to benchmark performance or facilitate understanding.

To conclude our overview of regret, we will now state the book's first set of actual results, which will not be difficult: we will study the regret of random search, as well as the obviously-bad algorithm that picks some arbitrary point over and over again. This might seem like a rather dry exercise, but it is worth doing, because certain details carry implications for benchmarking and are therefore worth drawing attention to.

Regret of Random Search

Proposition 2.25. *For any reward function r which is bounded and not (almost everywhere) constant, random search incurs linear regret, namely*

$$R_T(p_{\mathrm{RS}}, r) = \Theta(T). \tag{2.17}$$

Proof. This follows directly from definitions and linearity of expectation by

$$R_T(p_{\mathrm{RS}}, r) = \mathop{\mathbb{E}}_{a_t \sim p_{\mathrm{RS}}} \sup_{a \in A} \sum_{t=1}^{T} r(a) - r(a_t) \tag{2.18}$$

$$= \sum_{t=1}^{T} \sup_{a \in A} r(a) - \mathop{\mathbb{E}}_{a_t \sim p_{\mathrm{RS}}} r(a_t) \tag{2.19}$$

$$= T \underbrace{\left(\sup_{a \in A} r(a) - \mathop{\mathbb{E}}_{a_t \sim p_{\mathrm{RS}}} r(a_t) \right)}_{\text{strictly positive}} \tag{2.20}$$

where the final term is finite by boundedness of rewards, non-negative by definition of a supremum, and non-zero because the rewards are non-constant. ■

We have noted that random search is almost never a strong algorithm: it ignores the data, does not attempt to balance tradeoffs between explore and exploit, and instead only explores. Nonetheless, its performance should be compared to an even-worse algorithm: one that picks some arbitrary point over and over again. Let us show how to carry this comparison out—under finite actions and bounded rewards, for concreteness.

Proposition 2.26. *Let A be finite with $|A| \geq 2$, and let $\mathcal{R} = \{r : A \to [-1, 1]\}$ consist of bounded functions. For an action a, let p_{δ_a} be the algorithm that deterministically plays a. Then there is a reward function $r_a \in \mathcal{R}$ for which*

$$R_T(p_{\delta_a}, r_a) = \sup_{\substack{p \in \mathcal{P} \\ r \in \mathcal{R}}} R_T(p, r) = \Theta(T). \tag{2.21}$$

Proof. Choose

$$r_a(a') = \begin{cases} -1 & a = a' \\ 1 & a \neq a' \end{cases} \tag{2.22}$$

so that $R_T(p_{\delta_a}, r_a) = 2T$. To check that this is the worst regret possible, consider an arbitrary algorithm p and reward function r, and write

$$R_T(p, r) = \mathop{\mathbb{E}}_{a_t \sim p} \sup_{a \in A} \sum_{t=1}^{T} r(a) - r(a_t) \leq \mathop{\mathbb{E}}_{a_t \sim p} \sum_{t=1}^{T} |r(a_r^*)| + |r(a_t)| \leq 2T. \quad (2.23)$$

■

This seemingly-trivial pair of calculations already shows that random search can be much better than picking some arbitrary action. While both methods achieve $\Theta(T)$ regret, for random search, the constant factor hidden in the Θ-notation is exactly the gap between the best action and the average action with respect to the uniform distribution. This is usually smaller than the worst possible regret, and for this reason random search can be a good initial baseline to benchmark for problems where it is otherwise not clear what to do.

Maximizing Expected Value

Intuitively, one might expect that learning is necessary in order to achieve good performance on an episodic decision problem. One can therefore ask: by itself, does it suffice? Arguably the simplest possible decision rule which actually uses a Bayesian model in its construction is that of *maximizing (posterior) expected value*, which is defined up to an arbitrary tie-breaking rule as

$$a_t = \mathop{\arg\max}_{a \in A} \mathbb{E}(r(a) \mid a_{1:t-1}, \sigma_{1:t-1}). \quad (2.24)$$

Let us see how this performs on a simple non-trivial example.

Proposition 2.27. *Consider a stochastic multi-armed bandit, with $|A| \geq 2$, bounded rewards $\mathcal{R} = \{r : A \to [0, 1]\}$, and standard Gaussian noise. Let the Bayesian model consist of a standard Gaussian prior on the rewards, along with a conjugate unit-variance Gaussian likelihood. Then, if the decision rule maximizes expected value, there is an $r \in \mathcal{R}$ for which*

$$R_T(p_{\text{EV}}, r) = \Omega(T). \quad (2.25)$$

The proof, given in Section 2.7, works by constructing a constant-probability unlucky event which causes the algorithm to never try the optimal action more than once. It is not difficult to construct similar failures for other problems, or for other models—indeed, an analogous result even holds for Bayesian regret in the model-matched setting! This suggests that the issue is fundamental: in sufficiently non-trivial situations, maximizing expected value simply does not produce sufficient exploration, and can be just as bad as learning nothing.

If maximizing posterior expected value doesn't work, what alternatives do? This question turns out to be mathematically rich, and it will take the whole of this book for us to develop an incomplete but reasonably-comprehensive sense for what is possible. We will see that several approaches, including ones with exploration mechanisms of very different-looking character, can work. To get there, we first need to understand a bit more about episodic decision problems, by developing a sense for what levels of performance are possible.

2.4. Problem Difficulty and Lower Bounds

We have now explored a wide set of notions of regret, which provide a metric by which to evaluate the performance of a particular decision-making algorithm on a particular instance of an episodic decision problem. In order to be able to determine whether an algorithm is performing well, we need to be able to assess the difficulty of a given problem. We therefore study how to do that.

For a given notion of regret, the difficulty of an episodic decision problem depends chiefly on three factors:

1. The function class $\mathcal{R}$.
2. The feedback function σ.
3. The time horizon T.

In particular, the structure of the action space A is encoded in the definition of $\mathcal{R}$, and the structure of the observation space Σ is encoded in the definition of σ. Thus, this list covers all components of Definition 2.1, along with the key hyperparameter arising from the regret's definition itself. As a rule, we generally do not consider dependence on the true reward function r, nor on the actual sample $r \sim q$ in the event that it is random. Instead, the way we handle this part of the definition differs according to the problem variant:

1. In the stochastic variant, we consider the worst case $r \in \mathcal{R}$.
2. In the Bayesian variant, we consider the average case $r \sim q$.

Other variants are handled analogously: for instance, in adversarial variants, we might consider worst-case fixed reward sequences, or worst-case reward sequences generated adaptively by an adversary based on the same information available to the learner. This perspective does not preclude one from studying dependence on a particular reward function instance $r \in \mathcal{R}$, but instead asks one to encode the relevant properties of the instance in the definition of the function class $\mathcal{R}$. We will soon see an example that illustrates this.

We are now ready to cast the intuitive description above into a formal definition. The final ingredient we need to consider is some parameter that we can change in order to make the problem easier or harder, so that different problems become comparable to one another quantitatively. This could be the time horizon T, or for instance the size of the action space $|A|$. We will call the set of these variables V.

Minimax Regret

Definition 2.28. *Consider a family of episodic decision problems of the stochastic variant, parameterized by a set of variables $v \in V \subseteq \mathbb{R}^d$. Define the* MINIMAX REGRET

$$M(v) = \inf_{p \in \mathcal{P}} \sup_{r \in \mathcal{R}} R_T(p, r). \tag{2.26}$$

We define the DIFFICULTY *of the parameterized family to be the rate by which $M(v)$ varies with v.*

This means the difficulty is defined according to the best performance achievable by any algorithm, assuming it plays against the worst-case function within the given function class. We assume throughout that the episodic decision problem is regular enough for this value to be well-defined. The difficulty of a Bayesian variant is defined similarly, but where we do not consider worst-case performance, and replace it with average-case performance with respect to the reward function's distribution.

Bayesian-optimal Regret

Definition 2.29. *Consider a family of episodic decision problems of the Bayesian variant, parameterized by a set of variables $v \in V \subseteq \mathbb{R}^d$. Define the* BAYESIAN-OPTIMAL REGRET

$$B(v) = \inf_{p \in \mathcal{P}} R_T(p, q). \tag{2.27}$$

We define the DIFFICULTY *of the parameterized family to be the rate by which $B(v)$ varies with v.*

This definition coincides with the optimal value function of the problem's underlying Markov decision process up to constants, as seen previously. Thus, we can now reinterpret Theorem 2.23 as characterizing the manner in which less-informative feedback functions lead to more-difficult problems.

An important part of both of these definitions is that we focus our attention on the *rate*, which describes how difficulty varies with the family's parameters. In particular, we can consider how the rate varies with the time horizon T, or with parameters such as the size of the action space $|A|$, assuming we are considering a finite-action problem. This notion of difficulty therefore mirrors perspectives commonly found in complexity theory and theoretical computer science.

To aid understanding, we now briefly state the rate of the two most basic examples considered previously—the latter, under two different reward function classes. We will omit various restrictions on the variables to ease notation, and will defer proofs of the necessary statements to later.

Example 2.30. *The difficulty of online learning with $N = |A|$ total actions and bounded rewards $\mathcal{R} = \{r : A \to [-1, 1]\}$, under either a stochastic variant with standard Gaussian noise, or an adversarial variant without noise, is $\sqrt{T \log N}$.*

Example 2.31. *The difficulty of a multi-armed bandit with $K = |A|$ arms and bounded rewards $\mathcal{R} = \{r : A \to [-1, 1]\}$, under a stochastic variant with standard Gaussian noise, is $\sqrt{KT}$.*

Example 2.32. *The difficulty of a multi-armed bandit with $K = |A|$ arms and bounded rewards with a gap of at least Δ, namely*

$$\mathcal{R} = \{r : A \to [-1, 1] : r(a^*) - r(a) \geq \Delta, \forall a \neq a^*\} \tag{2.28}$$

where $a^* = \arg\max_{a \in A} r(a)$, under a stochastic variant with standard Gaussian noise, is $(K-1)\frac{\log T}{\Delta}$.

This illustrates how both the reward function class and the feedback function can affect an episodic decision problem's difficulty. We choose these three examples because the arguments involved are sufficiently simple that it is reasonable to present them as part of an introductory chapter—we will do so once a little bit more of the necessary machinery is established.

At this stage, it is reasonable to ask: what is gained by studying rates, rather than one episodic decision problem by itself? Why not just compute the saddle point arising in the minimax regret, and look at the resulting algorithm? If this were feasible, it would be fantastic—but, unfortunately, there are only a few problems where computing this quantity, or its analog for other variants, is possible. Working with rates gives us the freedom to study a much larger class of possible algorithms, many of which will turn out to be interesting.

Regret Lower Bounds

Focusing on the rate means that, to compute the difficulty of an episodic decision problem, it suffices to compute a sufficiently-sharp *regret lower bound*

$$C \leq R_T(p, r) \tag{2.29}$$

in the sense that for any algorithm p there is a reward r for which the inequality holds. For the given algorithm-specific reward, we automatically have

$$C \leq R_T(p, r) \leq \sup_{r \in \mathcal{R}} R_T(p, r), \quad \forall p \in \mathcal{P} \tag{2.30}$$

and since the statement is universally quantified over p, it is equivalent to

$$C \leq \inf_{p \in \mathcal{P}} \sup_{r \in \mathcal{R}} R_T(p, r). \tag{2.31}$$

If the lower bound is sharp up to constants, the specific form of C therefore reveals the correct rate. Thus, by itself, a lower bound shows a problem to be *at least* of a certain difficulty. Showing it to be *exactly* a given difficulty requires one to have a matching upper bound or some other way to attest sharpness.

So, how does one actually obtain lower bounds? We need to prove that, for *any* algorithm p, there is some kind of difficult reward function. As we will see in this book, there are many algorithms one can consider—ones whose principles look sufficiently different from one another. From this viewpoint, finding a universal construction for difficult reward functions may seem daunting. We will bypass this challenge rather than solving it directly, through a fundamental idea which recurs in many different guises throughout computer science.

Hard Reward Distributions

The key idea is to not look for a family of difficult reward functions directly, but to instead look for a *hard distribution* $q \in \mathcal{M}_1(\mathcal{R})$ over reward functions. Such a distribution should certify a regret lower bound in expectation. The key observation is that the existence of such a distribution implies the existence of a difficult reward function which certifies the required regret lower bound.

Lemma 2.33. *Suppose there is a $q \in \mathcal{M}_1(\mathcal{R})$ such that, for any $p \in \mathcal{P}$, we have*

$$C \leq R_T(p, q) = \mathop{\mathbb{E}}_{r\sim q} R_T(p, r). \tag{2.32}$$

Then for every $p \in \mathcal{P}$ there is an $r_p \in \mathcal{R}$ for which

$$C \leq R_T(p, r_p). \tag{2.33}$$

Proof. Suppose the contrary, namely that $R_T(p, r) < C$ for all $r \in \mathcal{R}$. Then by averaging over q, we would conclude $\mathbb{E}_{r\sim q} R_T(p, r) < C$—contradiction. ■

We can therefore pass from working with deterministic reward functions to working with randomized ones, and study what properties the corresponding distributions need to have. Naturally, these properties must be studied in full detail case-by-case. But there is a central principle to how they operate. Let us illustrate a special case of it through the following observation.

Equalizing Reward Distributions

Lemma 2.34. *Suppose that q is* EQUALIZING*, in the sense that the function*

$$a \mapsto \mathop{\mathbb{E}}_{r\sim q}(r(a) \mid a_{1:t}, \sigma_{1:t}) \tag{2.34}$$

is constant for any history of observations $a_{1:t}, \sigma_{1:t}$. Then $C = R_T(\cdot, q)$ is constant in its first argument, and thus we have $C \leq R_T(p, r_p)$.

Proof. We first note that, generically, the conditional expectation above does not depend on the learner's algorithm, so the statement itself makes sense. We have $R_T(p, q) = \mathbb{E}_{\substack{a_t\sim p\\ r\sim q}} \sup_{a\in A} \sum_{t=1}^T r(a) - r(a_t)$. The first term is by definition constant in p. For the second term, letting a' be an arbitrary action, we have

$$\mathbb{E}\sum_{t=1}^T r(a_t) = \mathbb{E}\sum_{t=1}^T \mathbb{E}(r(a_t) \mid a_{1:t-1}, \sigma_{1:t-1}) \tag{2.35}$$

$$\overset{\text{(i)}}{=} \mathbb{E}\sum_{t=1}^T \mathbb{E}(r(a') \mid a_{1:t-1}, \sigma_{1:t-1}) \overset{\text{(ii)}}{=} \mathbb{E}\sum_{t=1}^T r(a') \tag{2.36}$$

where (i) uses equalization together with the fact that a_t is conditionally independent of r, and (ii) applies the Tower Rule, where the result does not depend on the choice of a'. Since r and a' are constant in p, the claim follows. ■

For reward distributions q with this property, no matter what an algorithm does, the conditional expected rewards of each action remain the same. There is nothing an algorithm can learn, it may as well play at random. Non-constant reward distributions with this property are rarely available: for most episodic decision problems, it is possible for the learner to learn at least *something* about the randomly-drawn rewards, even if they are zero in expectation. One important exception is the adversarial full feedback setting.

In presenting these results, we defer proofs to Section 2.7. The key reason for this is they represent a jump in difficulty—not a large one, but enough that we would rather focus attention on showcasing what is known, before seeing how.

Proposition 2.35. *Consider adversarial online learning under full feedback, with bounded rewards $\mathcal{R} = \{r : A \to [-1, 1]\}$ and no noise. Define the reward distribution q to be independent Rademacher across time and actions, namely*

$$r_t(a) \sim \mathrm{Rad}(\tfrac{1}{2}). \tag{2.37}$$

Under this distribution, for any algorithm, if we suppose that $N = |A| \geq 16$, T is even, and $T \geq \log N$, then

$$\frac{1}{15}\sqrt{T \log N} \leq R_T(\cdot, q). \tag{2.38}$$

Constructions of this kind are fundamental in adversarial online learning, including generalizations where the action space A has a richer structure. If the adversary is playing random noise at every iteration, the learner has no hope of finding the best point, so their performance will be controlled by the typical gap between the best point and an average point. The bound follows by quantifying this gap using standard tools developed for analyzing expected suprema of random processes, here the sum of rewards over time.

Nearly-equalizing Reward Distributions

In non-adversarial settings with a sufficiently-rich feedback function σ, we cannot expect to find an equalizing reward distribution. The problem is that $r \sim q$ is sampled once at the beginning, which means it will be possible for the algorithm to learn at least something from the data. But, we can instead find a *nearly-equalizing* distribution, and try to quantify how much it was possible to learn. This leads to the following general principle for deriving lower bounds:

($\star$) *Sharp lower bounds usually arise from reward distributions q under which it is difficult for the learner to tell what the optimal action is.*

This idea can be made precise in many distinct ways, and the guise through which it shows up in a given context can vary significantly depending on its details. As a result, we will develop this idea through presenting it in a number of relatively simple cases—though, in doing so, will try to cast as much emphasis on the broader picture as we can. We start with the setting that mirrors the preceding one—namely, stochastic online learning.

Proposition 2.36. *Consider stochastic online learning under full feedback, with bounded rewards $\mathcal{R} = \{r : A \to [0, 1]\}$ and standard Gaussian noise. Define the reward distribution q according to*

$$r(a) = \Delta \mathbb{1}_{a=\alpha} \qquad\qquad \alpha \sim \mathrm{U}(A). \tag{2.39}$$

Under this distribution, for any algorithm, if we suppose that $N = |A| \geq 16$ and $T \geq \frac{\log N}{4}$, and take $\Delta = \sqrt{\frac{\log N}{4T}}$, then

$$\frac{1}{4}\sqrt{T \log N} \leq R_T(\cdot, q). \tag{2.40}$$

The hard distribution is constructed using *random spike rewards* that are all-zero, except for one special action whose reward is Δ, where the optimal action is

chosen uniformly at random. The rewards at each point are the same in expectation at the beginning, and the observations are nearly the same: the reward of the best action differs only a small amount, namely Δ, and this is hard to separate from variability. In this sense, this reward distribution is nearly-equalizing.

The proof operates by introducing and considering *information-theoretic* quantities such as Kullback–Leibler divergences and mutual informations, and rests on an explicit calculation for how different the observations would have looked if two different arms were optimal—this is how Δ makes an appearance. There are many possible variations on the argument, some of which may avoid restrictions on N and T at cost of being more tricky. We used rewards in $[0, 1]$, rather than $[-1, 1]$ as before, to simplify algebra.

It turns out that the *exact same* reward distribution q is also the hard instance for stochastic bandits, with a proof that is similar in spirit, but slightly different in terms of its technical details.

Proposition 2.37. *Consider a stochastic multi-armed bandit, with bounded rewards $\mathcal{R} = \{r : A \to [0, 1]\}$ and standard Gaussian noise. Define the reward distribution q according to*

$$r(a) = \Delta \mathbb{1}_{a=\alpha} \qquad \alpha \sim \mathrm{U}(A). \tag{2.41}$$

Under this distribution, for any algorithm, if we suppose that $K = |A| \geq 2$ and $T \geq \frac{K}{4}$, and take $\Delta = \frac{1}{2}\sqrt{\frac{K}{T}}$, then

$$\frac{1}{8}\sqrt{KT} \leq R_T(\cdot, q). \tag{2.42}$$

For bandits, the argument presented in this chapter goes through Pinsker's inequality, whereas for online learning, it went through Fano's inequality. For more details on these and other technical aspects, see the proofs in Section 2.7.

We now turn to the final example we will work out in detail: stochastic bandits, but where the reward function class only includes functions with some gap $\Delta > 0$. The most important aspect we will see is that this changes the rate.

Proposition 2.38. *Consider a stochastic multi-armed bandit, where the rewards are bounded with a gap, namely*

$$\mathcal{R} = \{r : A \to [0, 1] : r(a^*) - r(a) \geq \Delta, \forall a \neq a^*\} \tag{2.43}$$

where $a^* = \arg\max_{a \in A} r(a)$, we assume $0 < \Delta \leq \frac{1}{2}$, and the noise is standard Gaussian. Define the reward distribution q according to

$$r(a) = \Delta \mathbb{1}_{a=\alpha_1} + 2\beta\Delta \mathbb{1}_{a=\alpha_2} \qquad \begin{array}{c} \alpha_1 \sim \mathrm{U}(A) \\ \alpha_2 \mid \alpha_1 \sim \mathrm{U}(A \setminus \{\alpha_1\}) \end{array} \qquad \beta \sim \mathrm{Ber}(\tfrac{1}{2}). \tag{2.44}$$

Under this distribution, for any algorithm, if we suppose that $K = |A| \geq 2$ *and* $\frac{\sqrt{K-1}}{T^{1/4}} \leq \Delta$*, then*

$$\frac{(K-1)\log T}{16\Delta} \leq R_T(\cdot, q). \tag{2.45}$$

This reveals that, if our lower bounds are tight, then multi-armed bandits with rewards that have a gap of at least Δ are an easier problem class than those which do not have a gap. Thus, the function class $\mathcal{R}$ plays a critical role in controlling problem difficulty. We will later see, by way of algorithms developed in Chapter 5, that they are indeed tight. The proof here is essentially the same as that of the preceding result, except that the gap is now specified rather than tuned, and minor modifications are made for sharpness.

This concludes our presentation of lower bounds—specifically those for which we will prove the respective claims. There are as many lower bound arguments in the literature as there are variations of episodic decision problems and related mathematical constructions, many with a rich and intricate structure. There is, however, one more result which we believe deserves presentation, because of how surprising it is.

Classification of Partial Monitoring Games

Result 2.39. *Consider an (adversarial) partial monitoring game, with a finite class of bounded but otherwise unstructured rewards* $\mathcal{R} \subset \{r : A \to [-1, 1]\}$ *with* $|\mathcal{R}| < \infty$*, and assume* σ *is deterministic. Then its difficulty is either:*

$$M(T) = \begin{cases} 0 & \text{if it is trivial} \\ \Theta(\sqrt{T}) & \text{if } \sigma \text{ is locally observable} \\ \Theta(T^{2/3}) & \text{if } \sigma \text{ is globally but not locally observable} \\ \Omega(T) & \text{if it is impossible.} \end{cases} \tag{2.46}$$

We will not formally define what is meant by global and local observability, other than to say that these notions formalize the essence of the question: *must the learner play strictly-suboptimal actions in order to learn what the optimal action is?* If not, the game is locally observable. If yes, the game is globally but not locally observable. The other two possibilities are corner cases which handle situations where the optimal action can be inferred without needing to look at data, or ones where it cannot be learned, even in principle.

This result should be seen as a major achievement of the theory of partial monitoring games, and is remarkable because of how simple it is. Its main limitation is that it only offers a classification with respect to time, not the number of actions, or properties of the reward function class $\mathcal{R}$. Nonetheless, we view it as a key mathematical hint that studying episodic decision-making at our level of generality is not hopeless—a hint that suggests a relatively universal understanding ought to be possible.

2.5. Empirical Benchmarking

The preceding sections introduced the notion of an *episodic decision problem*, and defined *algorithms* for such problems, including the class of *Bayesian algorithms* that will be our main focus. We introduced notions of *regret* that can be used to distinguish between algorithms that perform well and ones that perform poorly, along with notions of *difficulty* through *regret lower bounds* that allow us to know what is possible. All of these notions were mathematical in character.

In Chapter 1, we noted that our focus throughout this book will be on definitions, and not on proofs and analysis techniques. This choice is not made by accident. There are at least two completely different ways that one can conclude a decision-making algorithm is strong:

1. Prove a theoretical guarantee on its regret, ideally in the form of a sharp upper bound in the setting of interest.
2. Implement and test the algorithm on a comprehensive suite of benchmarks, and check that it performs well in practice.

Our perspective will be to *take both approaches seriously*, as each one has different strength and weaknesses, and thereby helps us see different parts of the overall picture. These include:

- *Scope.* Mathematical guarantees are rigid: they hold exactly as stated. The degree to which a guarantee in one situation generalizes to another situation, in the absence of a proof, is a matter of speculation and personal judgment. The same is true of performance on a set of benchmarks.
- *Realism.* Performance guarantees are only possible for settings which are simple enough that mathematical tools can say something non-trivial about them. In contrast, benchmarks can be performed on arbitrarily-complex and realistic problems, so long as running them is practical.
- *Comprehensiveness.* A regret lower bound can reveal what degrees of performance are outright impossible. In contrast, if no algorithm performs well on a benchmark, it is not obvious whether this is because the problem is impossible, or because the known algorithms are not good enough.
- *Accessibility.* A theoretical result requires a certain degree of mathematical literacy in order to interpret. Benchmarks, in contrast, require one to be able to distinguish good experimental practices from poor ones. Both require a non-trivial degree of skill, but the skills themselves differ.
- *Reproducibility.* A correct theorem is by definition perfectly reproducible. In contrast, benchmarks have hyperparameters and implementation details, and are typically randomized. It is possible for these factors to influence performance as much as the benchmark or algorithm itself does.

These differences are by no means complete, and we refer the reader to Hardt (2026) for a systematic introduction to machine learning benchmarking, as well as an overview of the properties a well-designed benchmark suite should have.

2.5.1. Best Practices and Implementation Details for Bayesian Optimization

We now discuss several decision-making-specific best practices. These help ensure that an empirical benchmark suite provides useful information about an algorithm's performance. Where possible, we will relate each best practice back to the theory seen so far, in order to contextualize it and justify the need to use it from first principles.

Black-box Optimization

Example 2.40. *An episodic decision problem is called a* BLACK-BOX OPTIMIZATION PROBLEM *if:*

1. *The action space is* $A = [0, 1]^d$.
2. *The reward function class* $\mathcal{R} \subseteq C(A; \mathbb{R})$ *is a subset of the space of continuous functions.*
3. *We consider both stochastic and Bayesian variants.*
4. *The observation space is* $\Sigma = \mathbb{R}$.
5. *The feedback function is* $\sigma(r, a) = r(a)$*, without any noise.*

Bayesian optimization refers to the use of Bayesian algorithms to solve black-box optimization problems and generalizations thereof. Garnett (2023) provide a comprehensive treatment of such algorithms, along with the models they are most commonly used with. We start with our most basic recommendation.

Random search baseline. One should always compare how an algorithm performs to an implementation of random search. One would expect any reasonable algorithm to be stronger than random search, and verifying that this occurs can be seen as the most basic test a good algorithm should pass. The justification for this principle is that, from a theoretical perspective, random search provides an upper bound on a problem's difficulty.

An algorithm can fail to outperform random search for at least four distinct reasons. First, there might be a bug in the implementation. Second, the problem might be impossible, in which case random search might be near-optimal. Third, the algorithm's hyperparameters might be poorly-tuned. Fourth, the algorithm might be a weak algorithm for the given setting.

To distinguish between these, one can examine the actions that an algorithm picks. In a reasonable problem, a good algorithm should typically pick points which balance expected performance with uncertainty. In addition, selected actions' posterior uncertainty should be neither zero nor maximal. One can compute and display appropriate quantities, such as posterior means and standard deviations, to evaluate whether this happens. If it does not, it can be an indication of a poorly-chosen model or poor tuning.

As a second, closely-related baseline, it can be worthwhile to consider *max-variance search*: this algorithm chooses the point of highest uncertainty at each

iteration. Similar to random search, if this algorithm is strong, it gives an indication that the episodic decision problem under study is not particularly rich.

In spite of how basic these comparisons are, and how easy they are to carry out, our experience is that they are not always performed—even in published research. It is also our experience that outperforming random search in relatively sophisticated practical settings, for instance those with a sufficiently high-dimensional action space, is often much harder than one would expect.

Problem randomization. To the extent it makes sense, in light of computational costs and other concerns, a benchmark should be randomized: this ensures an algorithm cannot perform well purely because it prefers certain actions, independent of any learning. Concretely, in global optimization, algorithms that tend to pick points near zero, no matter what happens, will perform better on objective functions whose optimum is at zero. This can be counteracted by randomizing the location of the optimum.

The principle here is that a benchmark should define a *non-trivial* episodic decision problem: its function class should not be a singleton. If one considers average-case performance, randomization corresponds to the Bayesian episodic decision problem variant. It can also be used to approximate performance of a stochastic variant, where the function class $\mathcal{R}$ is defined by the distribution's support: to do this, one should not average, but instead compute worst-case performance over all random samples.

Model evaluation. A Bayesian decision-making algorithm involves multiple moving parts: the Bayesian model, its numerical implementation, and the decision rule built on top of the model. We recommend evaluating these by assessing the model's performance first: if a model is not working well, there is no reason to expect a decision-making algorithm built on top of it to work well.

In particular, a Bayesian model's performance can be assessed by considering how accurate its predictions are on a held-out test set. Moreover, a model's marginal likelihood, if it can be computed, directly indicates how typical or atypical a model views the data it has seen, which in turn provides a sense of how well the model expects itself to perform. For more about Bayesian model evaluation in Gaussian process models, which are the most-common models used in Bayesian optimization, see Rasmussen and Williams (2006).

Normalization. A given algorithm for an episodic decision problem can either be implemented directly according to its equations, or with normalization, defined as follows. At each time point, we compute the *normalized observations*

$$\sigma_{1:t}^{(\mathrm{std})} = \frac{\sigma_{1:t} - \mathrm{mean}(\sigma_{1:t})}{\mathrm{std}(\sigma_{1:t})} \tag{2.47}$$

and provide these to the model instead of the original observations, where we have assumed that computing means and standard deviations actually makes sense in the given setting. Note that the rescaling this gives is time-dependent.

By changing the behavior of a Bayesian model, normalization will change how an algorithm explores, and will generally do so in a non-obvious manner. We are not aware of any general theoretical study of its behavior. Nonetheless, there is strong empirical evidence that using it often significantly improves performance and reliability in practice, and as a result it is usually enabled by default in practical black-box optimization packages.

It is difficult to speculate whether these observations are due to better algorithmic behavior, or due to better-behaved models or other aspects of problem formulation. When benchmarking an algorithm, we therefore recommend trying it both with and without normalization, and reporting both results.

Numerical stability. Almost all Bayesian models require computational methods in order to actually obtain their posterior distribution in practice. These can involve everything from numerical linear algebra, for instance to solve linear systems arising in Gaussian process models, to optimization and sampling algorithms. For each numerical algorithm used, one should consider carefully whether its behavior will change significantly if implemented in 64-bit floating point arithmetic instead of 32-bit arithmetic.

For an introduction to these considerations, we refer the reader to Higham (1996). At the same time, there are enough algorithms one might use under-the-hood for Bayesian computation, that no single treatment can hope to cover all that one might encounter—especially new or recently-popularized algorithms. Thus, we omit further details. In practice, one should expect to need to think about the situation at hand, at least a little bit, in every case individually.

There is, however, one very useful fact about floating-point arithmetic that one can use to improve numerical stability of many implementations: there are about as many floating-point numbers in the interval $[0, 1)$ as in $[1, \infty)$. Thus, where choices are possible, it is often a good idea to standardize the numerical representation of actions and other variables to ensure they live near the origin.

Acquisition function optimization. In practice, for algorithms based on acquisition functions—as all algorithms considered in this book are—one needs to solve their respective optimization problems numerically. These optimization problems are essentially-never convex. The standard approach for solving them is to apply multi-start gradient-based optimization: one picks a set of random initial points, runs an optimization algorithm on each point in parallel, and picks the best achieved value.

In practice, both stochastic optimization methods such as Adam, and more classical quasi-Newton methods such as L-BFGS, are commonly used—the latter only for purely-deterministic acquisition functions which do not involve random sampling in their computations, where its use actually makes sense. For such algorithms, we recommend trying both approaches, comparing performance, and choosing whatever method is strongest in practice.

Evaluation with and without model mismatch. We recommend that Bayesian algorithms are evaluated both with and without model-mismatch, to quantify

the effect that it has on performance. The principle here is that *good algorithms should be general*: their performance should not be fundamentally tied to the reward function class or distribution.

To evaluate a model without model-mismatch, one can randomly sample reward functions from it. In some model classes, this may require appropriate approximations, which are safe to make as long as the error they induce is comparable to other sources of errors in the total setup.

To evaluate a model under model-mismatch, one can benchmark it against reward functions from a standard benchmark suite. We discuss this in more detail next. For now, though, we emphasize that performance under model-mismatch can change in model-specific ways, and can also be affected by numerical implementation aspects, such as training algorithm convergence.

Use of both synthetic and empirical benchmarks. When evaluating an algorithm on a benchmark suite of reward functions, we recommend using a combination of both synthetic benchmark functions, and those constructed to resemble real-world problems. The implied reward function classes that such examples come from can be significantly different from one another. As a result, understanding how performance differs in both cases can provide information about an algorithm's overall reliability.

Use of both community-standard and original benchmarks. Finally, when evaluating an algorithm's performance, we recommend considering both standard benchmarks widely used in the literature, such as the well-known *Ackley* and *Rosenbrock* functions, and bespoke benchmarks which are different from those considered by others. A good algorithm should perform similarly in both cases, and should avoid overfitting to the specific instances the community has collectively decided to test most algorithms on.

2.5.2. Benchmarking Bayesian Optimization

To conclude this section, we present a simple comparison of each of the decision-making algorithms studied in this book's chapters, using a from-scratch implementation written in a manner that aims to be long-term reproducible. In doing so, our goal is to provide a simple snapshot of how well the methods studied in today's era actually work. We describe our benchmark suite and full experimental details in Appendix B.

Under construction.

2.6. Exercises

The exercises in this book will be of a somewhat non-standard character: they are not intended to be attempted in isolation by the reader, but are instead designed for cooperative use with an AI assistant. Thus, they are somewhat less precisely specified than typical mathematical exercises. The intention is that,

in working together with the AI system to complete them, you apply an appropriate degree of effort to engage deeply with the material, thinking carefully about what is going on.

Some of our exercises will focus on constructing definitions and exploring their properties. When working through these, you should instruct the AI to never reveal an exercise's answer to you—instead, when tempted to do so, it should ask you a carefully-chosen guiding question in response. Thinking about this question should help you realize what you need to do to make progress towards the answer. You should allow the AI to perform calculations for you, but should also rely solely on your own thinking to tell it what calculations to perform.

Other exercises will require you to implement a minimal software package to put the definitions into practice. The purpose here is to learn how the mathematical ideas map onto software, and gain command over all of the details involved in such an implementation. You should take a central role in designing your package's structure, abstractions, and interface, and think carefully about how to make them as clean as possible. You should hand over low-level numerical details to AI, and then verify that they are implemented correctly.

2.1 (Simple vs. Cumulative Regret)**.** In this exercise, you will study the relationship between simple and cumulative regret. Construct a pair of algorithms, which are allowed to depend on the time horizon T, where, respectively:

1. Cumulative regret is $o(T)$, but simple regret does not converge to zero.
2. Simple regret converges to zero, but cumulative regret is $\Omega(T)$.

In spite of this, the two notions are closely related. For $T \geq 2$, show that:

1. For any algorithm, we have

$$R_T(p, r) \leq (1 + \log T) \max_{t=1,..,T} t \cdot r_t(p, r). \tag{2.48}$$

2. Given an algorithm p, there exists an algorithm p' for which

$$r_T(p', r) \leq \frac{1}{T-1} R_T(p, r). \tag{2.49}$$

In doing so, you might find it helpful to work with the *time-horizon-weighted simple regret*, defined as $t \cdot r_t(p, r)$.

2.2 (Budget-constrained Regret)**.** In this exercise, you will define another variant of an episodic decision problem, together with an associated form of regret. To do so, write down formal definitions for each of the following steps:

1. Extend Definition 2.1 to additionally include a *cost function class* $\mathcal{C}$. Decide which components of the core definition need to be modified, and how, in order to ensure that the resulting definition makes technical sense.

2. Extend Definition 2.11 to allow the algorithm to decide when to stop the decision-making process, formalizing the ideas discussed informally in the context of cost-adjusted simple regret.

3. Let $B > 0$ be a number corresponding to the *total cost budget*. Introduce a notion of a *feasible algorithm class*, consisting of algorithms that do not exceed the budget, and formalize this notion in at least one way.

4. Consider alternative definitions for the above step. Can you think of a second, non-equivalent way to formalize such a notion of a feasible algorithm class? *Hint:* the algorithm's choice to stop can be performed adaptively based on the randomly-generated data. Should we require constraints to hold in all random realizations? Or, is it enough for them to hold most of the time? What about on average?

5. Deduce whether the notion of *simple regret* continues to make sense under your chosen form of a feasible algorithm class. Is there always at least one algorithm which satisfies the constraint?

6. Consider the Bayesian form of simple regret, together with your definitions introduced above. Does the optimization problem for the Bayesian optimal policy form a constrained optimization problem? If so, formally write down the optimization objective and constraint set.

7. Suppose the resulting optimization problem is regular-enough that a Lagrange Multiplier Theorem holds. What can you conclude about its Lagrangified form? Does it resemble a definition you have seen previously?

For the last step, you are welcome to rely on informal reasoning: making this step fully precise and correct, in cases where it is possible, is more difficult than it looks. The problem is that a very careful handling of tie-breaking rules may be needed to ensure that an appropriate Lagrange Multiplier Theorem actually holds with equality.

2.3 (Research as an Episodic Decision Problem)**.** In this exercise, your goal will be to show that the process of studying the mathematical properties of an episodic decision problem can itself be viewed as an episodic decision problem. Assume that we have an episodic decision problem, together with a collection of N algorithms for solving that problem. Then:

1. Formally define an action space which corresponds to *benchmarking each possible algorithm* in some manner chosen by you, where each algorithm can be benchmarked in at least two different ways.

2. Formally define a reward function class which maps each algorithm to its overall *ground-truth rank*, where the best algorithm is ranked first, the worst algorithm is ranked last, and the rank is assumed to depend only on the algorithm. What properties must the reward function class satisfy?

3. Formally define an observation space and feedback function which describes the output produced when a given algorithm is tested. Describe

the relationship between the types of benchmarks and the algorithm's ground-truth rank.

4. Describe the tradeoffs between explore and exploit which occur in your setup. How would you expect these to change, intuitively speaking, if a cost function class was added in, so that benchmarks were split into cheap-but-noisy and expensive-but-accurate options?

2.4. Build a library using your favorite machine learning programming framework for black-box global optimization benchmarking. To do so:

1. Implement at least the *Ackley*, *Levy*, and *Rosenbrock* functions, whose expressions are given in Appendix B.
2. Invent and implement at least one original benchmark objective function. How easy or difficult do you expect this objective function to be?
3. Implement and test the random search algorithm on these benchmarks. How well does it do, in terms of regret?

2.5. Find a library built on top of your favorite machine learning programming framework which supports Bayesian learning via Gaussian process models, or any other model class of your choice. To prepare for exercises in future chapters, which will involve implementing Bayesian optimization under various acquisition functions, you will implement the abstractions needed to run a decision-making loop. To do so:

1. Build a class representing the learner, which includes the Gaussian process or other Bayesian model chosen by you, together with an acquisition function, and a record of all data that has been seen so far.
2. Implement the random search acquisition function, which ignores the model and data, but will form an initial baseline.
3. Implement a class for the decision-making loop, and run the model together with the random search acquisition function. Verify the regret matches that of the previous benchmarking exercise.

You are encouraged to complete this exercise only partially, and return to it as you read further chapters and learn about actual algorithms and acquisition functions. These will make it more clear how to organize your code cleanly.

2.6. Modify the proof of Theorem 2.23 to relax the finiteness assumptions on A and Σ, replacing them with the most general notions you have command over. *Hint:* at minimum, we recommend taking A, Σ, and $\mathcal{R}$ to be Polish topological spaces, with A and Σ compact, $\mathcal{R}$ consisting of uniformly bounded functions, all relevant functions assumed measurable, and, if needed, garblings allowed to depend on the prior q. If the preceding notions are unfamiliar or difficult, try with A finite, $\Sigma = [0, 1]$, and where σ admits a continuous Lebesgue density.

2.7. Prove Proposition 2.38. *Hint:* consider first learning the proof of Proposition 2.37, which is similar. The main difference you should expect is that the argument's core will involve the Bretagnolle–Huber inequality instead of Pinsker's inequality.

2.7. Deferred Proofs

Theorem 2.23. *Let $\sigma_1 : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ and $\sigma_2 : \mathcal{R} \times A \to \mathcal{M}_1(\Sigma)$ be two feedback functions, where A and Σ are assumed finite. Then the following are equivalent:*

1. *For any two episodic decision problems of the Bayesian variant whose feedback functions are mutually-analogous to σ_1 and σ_2, and where all other components are defined identically—including the priors and time horizons—the respective value functions satisfy $V^*_{\mathrm{MDP},2}(s_0) \leq V^*_{\mathrm{MDP},1}(s_0)$.*
2. *There is a GARBLING $g \in \mathcal{M}_1(\Sigma \mid \Sigma \times A)$ such that $\sigma_2(r, a) = g(\sigma_1(r, a), a)$ for all $r \in \mathcal{R}$ and $a \in A$.*
3. *Under the same quantifiers as in the first point, the set of feasible action-sequence distributions is larger under σ_1 than σ_2, in the sense $\Lambda_2 \subseteq \Lambda_1$.*

As consequence, one can define the BLACKWELL ORDER $\preceq$ over feedback functions, which forms a partial order, up to an equivalence.

Proof. Our argument loosely follows De Oliveira (2018, Theorem 5), but is adapted to the sequential setting and formalism used by this book. We will proceed by proving a circular chain of implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$ for the three characterizations. Before diving into the technical details, it is worth briefly noting why each implication ought to hold:

1. $1 \Rightarrow 2$: arguing by contrapositive, if no garbling exists, then σ_1 provides genuinely different information compared to σ_2, so picking a reward that depends on that information leads to the opposite inequality on optimal values compared to what was assumed.
2. $2 \Rightarrow 3$: an algorithm can always choose to imitate a garbling using its own internal randomness, so the set of algorithms pre-garbling is at least as big as the set of algorithms post-garbling.
3. $3 \Rightarrow 1$: the optimal value V^*_{MDP} can be written as the maximum of a certain linear functional over the set of feasible action-sequence distributions, thus increasing the feasible set results in a higher value.

We now make these claims precise.

Part I: $1 \Rightarrow 2$. We argue by contrapositive: assume that for any garbling $g \in \mathcal{M}_1(\Sigma \mid \Sigma \times A)$ there is an $\widetilde{a} \in A$ and $\widetilde{r} \in \mathcal{R}$ such that $\sigma_2(\widetilde{r}, \widetilde{a}) \neq g(\sigma_1(\widetilde{r}, \widetilde{a}), \widetilde{a})$. Since g is by definition a function of a, we can equivalently assume that there is an $\widetilde{a}$ such that for any garbling $g(\cdot, \widetilde{a}) \in \mathcal{M}_1(\Sigma \mid \Sigma)$ there is an $\widetilde{r} \in \mathcal{R}$ for which

$\sigma_2(\widetilde{r}, \widetilde{a}) \neq g(\sigma_1(\widetilde{r}, \widetilde{a}))$. From this, we seek to prove $V^*_{\mathrm{MDP},\sigma_2'}(s_0) > V^*_{\mathrm{MDP},\sigma_1'}(s_0)$ for some mutually-analogous σ_1', σ_2', as well as T and q. We choose $T = 2$.

Next, we handle the choice of q. For each r, define the set $\{g \in \mathcal{M}_1(\Sigma \mid \Sigma) : \sigma_2(r, \widetilde{a}) = g(\sigma_1(r, \widetilde{a}))\}$, which is a closed subset of $\mathcal{M}_1(\Sigma \mid \Sigma)$. By hypothesis, the intersection of such sets over all of $\mathcal{R}$ is empty. Since $\mathcal{M}_1(\Sigma \mid \Sigma)$ is a compact subset of $\mathbb{R}^d$ for some d, there must be a finite subset $\mathcal{R}_q \subseteq \mathcal{R}$ for which the intersection is also empty. Choose q to be uniform on this subset.

We now define the mutually-analogous feedback functions. Choose A' to be the disjoint union $A' = A \oplus \Sigma \oplus \{\boxdot\}$. For each function $r \in \mathcal{R}_q$, define the function

$$r'(a) = \begin{cases} 0 & a = \widetilde{a} \\ c & a \in A \setminus \{\widetilde{a}\} \\ s(r, a) & a \in \Sigma \\ c - i(r) & a = \boxdot \end{cases} \tag{2.50}$$

where $i : \mathcal{R}_q \to [0, 1]$ is an arbitrary injective function, $c < 0$ is a constant, and $s : \mathcal{R}_q \times \Sigma \to \mathbb{R}$ is a function, the latter two to be determined later. Define $\mathcal{R}'$ to be the set of all such functions: by injectivity of i, it is in bijection with $\mathcal{R}_q \subseteq \mathcal{R}$, so take ρ the inverse of this bijection. Together, these ingredients, along with the requirement of being analogous to σ_1 and σ_2, respectively, lead uniquely to a definition of σ_1' and σ_2'. Consider the set of all garbled observations

$$\mathcal{G} = \{g(\sigma_1(r, \widetilde{a})) \in \mathcal{M}_1(\Sigma \mid \mathcal{R}_q) : g \in \mathcal{M}_1(\Sigma \mid \Sigma)\}. \tag{2.51}$$

This set is convex and compact, and by definition of $\mathcal{R}_q$ we have $\sigma_2(\cdot, \widetilde{a}) \notin \mathcal{G}$. Hence, there is a function $s : \mathcal{R}_q \times \Sigma \to \mathbb{R}$ for which

$$\mathop{\mathbb{E}}_{\substack{r \sim q \\ \varsigma \sim \gamma(\cdot \mid r)}} s(r, \varsigma) < \mathop{\mathbb{E}}_{\substack{r \sim q \\ \varsigma \sim \sigma_2(r, \widetilde{a})}} s(r, \varsigma), \quad \forall \gamma \in \mathcal{G}. \tag{2.52}$$

Moreover, given one such function, one can always obtain another one by adding any constant-in-ς function. Using this, we replace $s(r, \varsigma)$ with $s(r, \varsigma) - s(r, \varsigma_s^*)$, where $\varsigma_s^* = \arg\max_{\varsigma \in \Sigma} \mathbb{E}_{r \sim q} s(r, \varsigma)$, with ties broken arbitrarily. This gives a choice of s which satisfies the same inequality, along with the properties

$$s(r, \varsigma_s^*) = 0, \quad \forall r \in \mathcal{R}_q \qquad \mathop{\mathbb{E}}_{r \sim q} s(r, \varsigma) \leq 0, \quad \forall \varsigma \in \Sigma. \tag{2.53}$$

We now relate the two sides of the above inequality to the optimal policies of interest, starting with the left-hand-side. Given $T = 2$, we take two actions, call them a_1 and a_2. We have

$$V^*_{\mathrm{MDP},\sigma_1'}(s_0) = \sup_{a_1 \in A'} \underbrace{\mathop{\mathbb{E}}_{r' \sim q'} r'(a_1) + \mathop{\mathbb{E}}_{\varsigma_1 \sim \sigma_1'(r', a_1)} \sup_{a_2 \in A'} \mathbb{E}(r'(a_2) \mid a_1, \varsigma_1)}_{\text{denote by } (*)} \tag{2.54}$$

where the latter conditional expectation is taken over the respective posterior

distribution over r'. Consider the inner supremum: given a_1 and ς_1, we have

$$\sup_{a_2 \in A'} \mathbb{E}(r'(a_2) \mid a_1, \varsigma_1) \overset{\text{(i)}}{=} \max \begin{pmatrix} 0, c, c - \mathbb{E}(i(r) \mid a_1, \varsigma_1), \\ \sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \end{pmatrix} \tag{2.55}$$

$$\overset{\text{(ii)}}{=} \sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \tag{2.56}$$

where (i) follows by splitting the supremum into four cases, (ii) follows because $s(r, \varsigma_s^*) = 0$ for all $r \in \mathcal{R}_q$ implies $\mathbb{E}(s(r, \varsigma_s^*) \mid a_1, \varsigma_1) = 0$, which together with $c < 0$ and $i(r) \geq 0$, implies $c - i(r) \leq c < 0 \leq \sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1)$. In doing so, we have calculated the optimal value function at $t = 2$. To handle $t = 1$, we will bound the sum which occurs inside the outer supremum for all possible values of a_1. We need to handle four cases:

1. $a_1 = \widetilde{a}$. Then $r'(a_1) = 0$, and the sum equals

$$(*) = \mathop{\mathbb{E}}_{\substack{r' \sim q' \\ \varsigma_1 \sim \sigma_1'(r', a_1)}} \sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \tag{2.57}$$

$$= \mathop{\mathbb{E}}_{\substack{r' \sim q' \\ \varsigma_1 \sim \sigma_1'(r', a_1)}} s(r, a^*_{a_1, \varsigma_1}) < \mathop{\mathbb{E}}_{\substack{r \sim q \\ \varsigma \sim \sigma_2(r, \widetilde{a})}} s(r, \varsigma) \tag{2.58}$$

   by taking γ to be the conditional distribution of any choice of maximizer $a^*_{a_1, \varsigma_1}$. Since a_1 is fixed, this conditional distribution depends only on ς_1, and hence lies in $\mathcal{G}$.

2. $a_1 \in A \setminus \{\widetilde{a}\}$. Then $r'(a_1) = c$, and the sum equals

$$(*) = c + \mathop{\mathbb{E}}_{\substack{r' \sim q' \\ \varsigma_1 \sim \sigma_1'(r', a_1)}} \sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \tag{2.59}$$

$$\leq 0 = \mathop{\mathbb{E}}_{r \sim q} s(r, \varsigma_s^*) < \mathop{\mathbb{E}}_{\substack{r \sim q \\ \varsigma \sim \sigma_2(r, \widetilde{a})}} s(r, \varsigma) \tag{2.60}$$

   by taking $c \leq -\sup_{r \in \mathcal{R}_q, \varsigma \in \Sigma} s(r, \varsigma)$, which is finite by finiteness of $\mathcal{R}_q$ and Σ, and $\gamma = \delta_{\varsigma_s^*}$, at which point all choices deferred at the beginning have been determined. This conditional distribution lies in $\mathcal{G}$ because it contains all constant kernels.

3. $a_1 \in \Sigma$. Then $\varsigma_1 = \boxtimes$ deterministically, and the posterior over r' is equal to the prior, which means $\sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \leq 0$ and

$$(*) \leq \mathop{\mathbb{E}}_{r \sim q} s(r, a_1) < \mathop{\mathbb{E}}_{\substack{r \sim q \\ \varsigma \sim \sigma_2(r, \widetilde{a})}} s(r, \varsigma) \tag{2.61}$$

   by taking $\gamma = \delta_{a_1}$, which similarly lies in $\mathcal{G}$ because it is constant in ς_1.

4. $a_1 = \boxdot$. As in the third case, we have $\varsigma_1 = \boxtimes$, and the posterior over r' is equal to the prior. Using $c - i(r) < 0$ and $\sup_{a_2 \in \Sigma} \mathbb{E}(s(r, a_2) \mid a_1, \varsigma_1) \leq 0$, we get an upper bound of zero on the sum. From there onwards, the desired upper bound follows identically to the second case.

Together, this proves

$$V^*_{\mathrm{MDP},\sigma'_1}(s_0) < \mathop{\mathbb{E}}_{\substack{r\sim q\\ \varsigma\sim\sigma_2(r,\widetilde{a})}} s(r,\varsigma). \tag{2.62}$$

We now proceed to handle the right-hand-side of this inequality: define the policy π to be the policy that chooses $a_1 = \widetilde{a}$ and $a_2 = \varsigma_1 \in A'$, where $\varsigma_1 \sim \sigma'_2(r', a_1)$ is the observed feedback. Then we have

$$\mathop{\mathbb{E}}_{\substack{r\sim q\\ \varsigma\sim\sigma_2(r,\widetilde{a})}} s(r,\varsigma) \overset{\text{(i)}}{=} V^{(\pi)}_{\mathrm{MDP},\sigma'_2}(s_0) \overset{\text{(ii)}}{\leq} V^*_{\mathrm{MDP},\sigma'_2}(s_0) \tag{2.63}$$

where (i) follows by definitions of r' and $V^{(\pi)}_{\mathrm{MDP},\sigma'_2}$, and (ii) follows by optimality. Combining inequalities, we conclude $V^*_{\mathrm{MDP},\sigma'_2}(s_0) > V^*_{\mathrm{MDP},\sigma'_1}(s_0)$. Part I follows.

Part II: $2 \Rightarrow 3$**.** Assume the existence of a garbling g such that $\sigma_2(r,a) = g(\sigma_1(r,a),a)$. To ease notation, we prove the claim for σ_1 and σ_2: the argument will extend immediately to a mutually-analogous pair. Take $\lambda_{p_2} \in \Lambda_{\sigma_2}$, which we recall is a function $\lambda_{p_2} : \mathcal{R} \to \mathcal{M}_1(\mathrm{Seq}(A))$, and denote its associated algorithm by $p_2 \in \mathcal{P}$, which we also recall is a function $p_2 : \mathrm{Seq}(A\times\Sigma) \to \mathcal{M}_1(A)$. We need to show $\lambda_{p_2} \in \Lambda_{\sigma_1}$. To do this, define the algorithm $\widetilde{p}_1$ according to

$$a_{t+1} \sim p_2(a_{1:t}, \widetilde{\sigma}_{1:t}) \qquad\qquad \widetilde{\sigma}_\tau \sim g(\sigma_\tau, a_\tau) \tag{2.64}$$

where we extend g to act on the random variable $\sigma_\tau \sim \sigma_1(r, a_\tau)$ in the natural manner. This algorithm uses its internal randomness to apply the garbling to the feedback it receives from σ_1, consistently across time, then plugs the result into p_2. The construction almost suffices: however, understood literally, $\widetilde{p}_1$ is not a map from $\mathrm{Seq}(A\times\Sigma)$ into $\mathcal{M}_1(A)$, due to the use of auxiliary randomness.

To alleviate this and define a valid algorithm, we marginalize this randomness out, and instead draw $\widetilde{\sigma}_{1:t}$ at each time point from the respective conditional distribution given $a_{1:t}, \sigma_{1:t}$. This defines a map $p_1 \in \mathcal{P}$, for which $\lambda_{p_1} \in \Lambda_{\sigma_1}$ by construction. At the same time, since $g(\sigma_1(r,a_\tau),a_\tau) = \sigma_2(r,a_\tau)$, by standard properties of conditional distributions the action-sequence distribution of p_1 is identical to that of p_2, hence $\lambda_{p_1} = \lambda_{p_2}$. Part II follows.

Part III: $3 \Rightarrow 1$**.** Assume that $\Lambda_{\sigma'_2} \subseteq \Lambda_{\sigma'_1}$. To show the claim, it suffices to show that an optimal value function V^*_{MDP} can be written as a supremum taken over its respective set Λ. For this, note that by definition of λ we have

$$\mathop{\mathbb{E}}_{a_{1:T}\sim\lambda(r)} \sum_{t=1}^{T} r(a_t) = \sum_{t=1}^{T} \mathbb{E}(r(a_t) \mid r). \tag{2.65}$$

Taking expectations and applying the Tower Rule gives

$$\mathop{\mathbb{E}}_{\substack{a_{1:T}\sim\lambda(r)\\ r\sim q}} \sum_{t=1}^{T} r(a_t) = \mathop{\mathbb{E}}_{r\sim q} \sum_{t=1}^{T} \mathbb{E}(r(a_t) \mid r) = \mathop{\mathbb{E}}_{\substack{a_t\sim p\\ r\sim q}} \sum_{t=1}^{T} r(a_t). \tag{2.66}$$

Taking suprema over $\mathcal{P}$ of both sides, and using the fact that Λ is by definition parameterized by $\mathcal{P}$ to rewrite the expression in terms of an equivalent supremum over Λ, we obtain

$$\sup_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{\substack{a_{1:T} \sim \lambda(r) \\ r \sim q}} \sum_{t=1}^{T} r(a_t) = \sup_{p \in \mathcal{P}} \mathop{\mathbb{E}}_{\substack{a_t \sim p \\ r \sim q}} \sum_{t=1}^{T} r(a_t) = V^*_{\mathrm{MDP}}(s_0). \tag{2.67}$$

Using this representation, the desired implication follows by relaxing the supremum defining $V^*_{\mathrm{MDP},\sigma'_2}$ from $\Lambda_{\sigma'_2}$ to $\Lambda_{\sigma'_1}$, obtaining $V^*_{\mathrm{MDP},\sigma'_1}$. Part III follows. ■

Proposition 2.27. *Consider a stochastic multi-armed bandit, with $|A| \geq 2$, bounded rewards $\mathcal{R} = \{r : A \to [0,1]\}$, and standard Gaussian noise. Let the Bayesian model consist of a standard Gaussian prior on the rewards, along with a conjugate unit-variance Gaussian likelihood. Then, if the decision rule maximizes expected value, there is an $r \in \mathcal{R}$ for which*

$$R_T(p_{\mathrm{EV}}, r) = \Omega(T). \tag{2.25}$$

Proof. To begin, we remind ourselves what the algorithm is doing: the posterior distribution for a given action is

$$r(a) \mid a_{1:t}, \sigma_{1:t} \sim \mathrm{N}\left(\frac{\sum_{\tau=1}^{t} \sigma_\tau \mathbb{1}_{a_\tau = a}}{N_a(t) + 1}, \frac{1}{N_a(t) + 1} \right) \tag{2.68}$$

where $N_a(t)$ is the number of times action a has been played up to time t. We will take our true reward function to be

$$r(a) = \mathbb{1}_{a=1}. \tag{2.69}$$

Thus, the first arm is optimal and gives a reward of one, while the others give zero rewards. Let $\varepsilon_t : A \to \mathbb{R}$ be the noise random variables at time t. To define our unlucky event, we require that two things happen:

1. At the first time τ when the optimal action $a = 1$ is chosen, if this occurs at all, we have $\varepsilon_\tau(1) \leq -3$, which makes the optimal action look very bad.
2. For the suboptimal action $a = 2$, the noise sum is never too-misleading, in the sense that $\sum_{\tau=1}^{t} \varepsilon_\tau(2) \mathbb{1}_{a_\tau = 2} > -N_2(t) - 1$ for all t.

Let U be the unlucky event that both of the above conditions occur. Consider what the algorithm will do in this situation: the posterior expectation for the suboptimal arm, at every time t, using $r(2) = 0$, will be

$$\frac{\sum_{\tau=1}^{t} \sigma_\tau \mathbb{1}_{a_\tau = 2}}{N_2(t) + 1} = \frac{\sum_{\tau=1}^{t} \varepsilon_\tau(2) \mathbb{1}_{a_\tau = 2}}{N_2(t) + 1} > -1. \tag{2.70}$$

Now, consider the optimal arm, namely $a = 1$. If, at time t, it has been played zero times, its posterior expectation is zero. If it has been played once, its posterior expectation is

$$\frac{\sum_{\tau=1}^{t} \sigma_\tau \mathbb{1}_{a_\tau = 1}}{N_1(t) + 1} \leq \frac{1 - 3}{2} = -1. \tag{2.71}$$

This shows that the arm $a = 1$ is never played more than once: from the algorithm's point of view, the action $a = 2$ is strictly better. Conditional on these events, it follows that $\mathbb{E}(N_1(T) \mid U) \leq 1$. Note also that $\mathbb{E}(N_1(T) \mid U^{\complement}) \leq T$.

We now proceed to bound the probability of U. Note first that we can re-index the noise vectors ε_τ to depend on a and $N_a(t)$ instead of a and t: when modified using this re-indexing, the two events defining U depend on disjoint random variables, and are therefore independent. It therefore suffices to bound both probabilities individually. To do this, let the respective events be U_1 and U_2. By definition, we have $\mathbb{P}(U_1) = \Phi(-3)$, where Φ is the Gaussian cumulative distribution function. For U_2 we apply a basic union bound to conclude

$$1 - \mathbb{P}(U_2) \leq \sum_{n=1}^{\infty} \Phi\left(-\frac{n+1}{\sqrt{n}}\right) \leq \frac{1}{2} \sum_{n=1}^{\infty} e^{-\frac{n+2}{2}} = \frac{1}{2e(\sqrt{e}-1)} \leq \frac{3}{10} \tag{2.72}$$

along with a Gaussian tail bound and some basic algebra. Now, consider how this affects regret: we have

$$\begin{aligned} R_T(p_{\mathrm{EV}}, r) &= T - \mathbb{E}\, N_1(T) && (2.73)\\ &= T - \mathbb{E}(N_1(T) \mid U)\, \mathbb{P}(U) - \mathbb{E}(N_1(T) \mid U^{\complement})(1 - \mathbb{P}(U)) && (2.74)\\ &\geq T - \mathbb{P}(U) - T(1 - \mathbb{P}(U)) && (2.75)\\ &= (T-1)\, \mathbb{P}(U) && (2.76) \end{aligned}$$

which, since $\mathbb{P}(U) \geq \frac{7}{10}\Phi(-3)$, is $\Omega(T)$. The claim follows. ∎

Proposition 2.35. *Consider adversarial online learning under full feedback, with bounded rewards $\mathcal{R} = \{r : A \to [-1, 1]\}$ and no noise. Define the reward distribution q to be independent Rademacher across time and actions, namely*

$$r_t(a) \sim \mathrm{Rad}(\tfrac{1}{2}). \tag{2.37}$$

Under this distribution, for any algorithm, if we suppose that $N = |A| \geq 16$, T is even, and $T \geq \log N$, then

$$\frac{1}{15}\sqrt{T \log N} \leq R_T(\cdot, q). \tag{2.38}$$

Proof. There are many variations of the argument presented here known within the literature's folklore: the one we give is essentially a specialization of Orabona (2026, Theorem 5.1 and Theorem 5.3), to our setting—which, themselves, are a sharpened form of Orabona and Pál (2015, Theorem 8). For an alternative argument, see for instance Negrea et al. (2021, Appendix C). Our strategy will be to apply various binomial probability estimates, including a tail bound, and we begin by rewriting the regret into a form amenable to this. Note first that

$$\begin{aligned} R_T(p, q) &= \mathop{\mathbb{E}}_{r_t \sim q} R_T(p, r_1, .., r_T) && (2.77)\\ &= \mathop{\mathbb{E}}_{\substack{a_t \sim p \\ r_t \sim q}} \sup_{a \in A} \sum_{t=1}^{T} r_t(a) - r_t(a_t) = \mathop{\mathbb{E}}_{r_t \sim q} \sup_{a \in A} \sum_{t=1}^{T} r_t(a) && (2.78) \end{aligned}$$

because a_t and r_t are independent, and $\mathbb{E}_{r_t \sim q}\, r_t(a_t) = 0$ since this is an expectation over a Rademacher random variable. This means that, for any p, the expected regret equals the supremum of an N-dimensional random vector whose components are independent Rademacher sums, which are in turn rescaled binomials. We will change notation to make this explicit, by writing

$$R_T(p,q) = \mathop{\mathbb{E}}_{r_t \sim q} \sup_{a \in A} \sum_{t=1}^{T} r_t(a) = \mathbb{E} \sup_{n=1,..,N} \sum_{t=1}^{T} \varepsilon_{n,t} = 2\,\mathbb{E} \sup_{n=1,..,N} b_{n,T} - T \quad (2.79)$$

where $\varepsilon_{n,t} \sim \text{Rad}(\frac{1}{2})$ independently across n and t, which means $\frac{\varepsilon_{n,t}+1}{2} \sim \text{Ber}(\frac{1}{2})$, and in turn $b_{n,T} \sim \text{Bin}(T, \frac{1}{2})$. Continuing from above, write

$$\begin{aligned} R_T(p,q) &= 2\,\mathbb{E} \sup_{n=1,..,N} b_{n,T} - T && (2.80)\\ &\overset{\text{(i)}}{=} 2 \sum_{u=1}^{T} \mathbb{P}\left(\sup_{n=1,..,N} b_{n,T} \geq u \right) - T && (2.81)\\ &= T - 2 \sum_{u=1}^{T} \mathbb{P}\left(\sup_{n=1,..,N} b_{n,T} < u \right) && (2.82)\\ &\overset{\text{(ii)}}{=} T - 2 \sum_{u=1}^{T} \mathbb{P}(b_{1,T} < u)^N && (2.83) \end{aligned}$$

where (i) follows from the tail sum identity for expectations of non-negative random variables, where the upper sum index is T because $\mathbb{P}(b_{n,T} \geq T+1) = 0$, and (ii) follows because $b_{n,T}$ are independent and identically distributed across n. From here, the idea is to split the sum to reflect two distinct regimes: for u small the overall probability is mostly governed by the fact that there are N binomials, whereas for u large it is mostly governed by tail behavior. Write

$$\begin{aligned} R_T(p,q) &= T - 2 \sum_{u=1}^{T/2} \mathbb{P}(b_{1,T} < u)^N - 2 \sum_{u=\frac{T}{2}+1}^{T} \mathbb{P}(b_{1,T} < u)^N && (2.84)\\ &\geq T - \frac{1}{2^{N-2}} \sum_{u=1}^{T/2} \mathbb{P}(b_{1,T} < u) - 2 \sum_{q'=1}^{T/2} \mathbb{P}\left(b_{1,T} < \frac{T}{2} + q' \right)^N && (2.85) \end{aligned}$$

applying $\mathbb{P}(b_{1,T} < u) \leq \frac{1}{2}$ for $u \leq \frac{T}{2}$ to $N-1$ of N terms, and using the assumption that T is even. Let us examine the middle term. We have

$$\begin{aligned} \sum_{u=1}^{T/2} \mathbb{P}(b_{1,T} < u) = \mathbb{E} \sum_{u=1}^{T/2} \mathbb{1}_{b_{1,T} < u} &\overset{\text{(i)}}{=} \mathbb{E} \max\left(0, \frac{T}{2} - b_{1,T}\right) && (2.86)\\ &\overset{\text{(ii)}}{=} \frac{1}{2}\,\mathbb{E}\left| \frac{T}{2} - b_{1,T} \right| \overset{\text{(iii)}}{\leq} \frac{\sqrt{T}}{4} && (2.87) \end{aligned}$$

where (i) follows because the sum of indicators by definition counts the number of integers u above the binomial random variable, and (ii) follows because $\frac{T}{2} -$

$b_{1,T}$ is symmetric, and (iii) follows by Jensen's inequality, along with the fact that $\mathrm{Var}(b_{1,T}) = \frac{T}{4}$. Combining, this gives

$$R_T(p,q) \geq T - \frac{\sqrt{T}}{2^N} - 2\sum_{q'=1}^{T/2} \mathbb{P}\left(b_{1,T} < \frac{T}{2} + q'\right)^N. \tag{2.88}$$

We are almost ready to apply the binomial tail bound, but will first simplify the sum a little bit in order to avoid getting swallowed up by needlessly complex algebra later on. For this, write

$$R_T(p,q) \overset{\text{(i)}}{\geq} T - \frac{\sqrt{T}}{2^N} - 2q_0\,\mathbb{P}\left(b_{1,T} < \frac{T}{2} + q_0\right)^N - 2\left(\frac{T}{2} - q_0\right) \tag{2.89}$$

$$= 2q_0\left(1 - \left(1 - \mathbb{P}\left(b_{1,T} \geq \frac{T}{2} + q_0\right)\right)^N\right) - \frac{\sqrt{T}}{2^N} \tag{2.90}$$

$$\overset{\text{(ii)}}{\geq} 2q_0\left(1 - \exp\left(-N\,\mathbb{P}\left(b_{1,T} \geq \frac{T}{2} + q_0\right)\right)\right) - \frac{\sqrt{T}}{2^N} \tag{2.91}$$

where for (i) we have picked a threshold $q_0 \in \mathbb{N}$, and used monotonicity to bound the first q_0 terms in the sum by $\mathbb{P}\big(b_{1,T} < \frac{T}{2} + q_0\big)$, while bounding the remaining $\frac{T}{2} - q_0$ probabilities in the sum by one, and for (ii) we have used the inequality $1 - (1-x)^N \geq 1 - e^{-Nx}$. We now apply the tail bound of Orabona (2026, Lemma A.17), whose hypotheses are satisfied because T is even and $q_0 \leq T/2 - 1$, and which tells us that

$$\mathbb{P}\left(b_{1,T} \geq \frac{T}{2} + q_0\right) \geq \frac{1}{3} \cdot \frac{1}{\frac{2q_0}{\sqrt{T}} + 1} \exp\left(-T D_{\mathrm{KL}}\left(\mathrm{Ber}\left(\frac{1}{2} + \frac{q_0}{T}\right) \,||\, \mathrm{Ber}\left(\frac{1}{2}\right)\right)\right) \tag{2.92}$$

$$\geq \frac{1}{3} \cdot \frac{1}{\frac{2q_0}{\sqrt{T}} + 1} \exp\left(-2\frac{q_0^2}{T} - 3.1\frac{q_0^4}{T^3}\right) \tag{2.93}$$

where the second form follows more-or-less by Taylor-expanding the respective Kullback–Leibler divergence between Bernoulli distributions: see the final line of the proof given in Orabona (2026, Theorem 5.5). Continuing the algebra, we get

$$R_T(p,q) \geq 2q_0\left(1 - \exp\left(-\frac{N}{3} \cdot \frac{1}{\frac{2q_0}{\sqrt{T}} + 1} \exp\left(-2\frac{q_0^2}{T} - 3.1\frac{q_0^4}{T^3}\right)\right)\right) - \frac{\sqrt{T}}{2^N}. \tag{2.94}$$

We now choose

$$q_0 = \left\lfloor \sqrt{\frac{T \log N}{8}} \right\rfloor. \tag{2.95}$$

The conditions $N \geq 16$ and $T \geq \log N$ imply a uniform bound on the exponential term, and after some algebra, we obtain

$$R_T(p,q) \geq \frac{13}{20}\left(\sqrt{\frac{T\log N}{2}} - 2\right) - \frac{1}{10^5}\sqrt{T\log N} \geq \frac{1}{15}\sqrt{T\log N} \tag{2.96}$$

which gives the claim. ■

Proposition 2.36. *Consider stochastic online learning under full feedback, with bounded rewards $\mathcal{R} = \{r : A \to [0,1]\}$ and standard Gaussian noise. Define the reward distribution q according to*

$$r(a) = \Delta \mathbb{1}_{a=\alpha} \qquad \alpha \sim \mathrm{U}(A). \tag{2.39}$$

Under this distribution, for any algorithm, if we suppose that $N = |A| \geq 16$ and $T \geq \frac{\log N}{4}$, and take $\Delta = \sqrt{\frac{\log N}{4T}}$, then

$$\frac{1}{4}\sqrt{T \log N} \leq R_T(\cdot, q). \tag{2.40}$$

Proof. To improve readability, we prove a mild generalization where the noise has variance ς^2. Note first that $\Delta \leq 1$ by assumption, thus the statement itself makes sense. Let $q_{\sigma_t|\alpha}$ denote the conditional distribution of σ_t given α, understood as a probability kernel. Since these are Gaussian, with the same diagonal covariance but different means, by the standard formula for $\alpha_1 \neq \alpha_2$ we have

$$D_{\mathrm{KL}}(q_{\sigma_t|\alpha_1} \mid\mid q_{\sigma_t|\alpha_2}) = \frac{\|\Delta(\mathbb{1}_{\cdot=\alpha_1} - \mathbb{1}_{\cdot=\alpha_2})\|^2}{2\varsigma^2} = \frac{\Delta^2}{\varsigma^2}. \tag{2.97}$$

Since $\sigma_{1:T}$ are independent across time, this means

$$D_{\mathrm{KL}}(q_{\sigma_{1:T}|\alpha_1} \mid\mid q_{\sigma_{1:T}|\alpha_2}) = \frac{T\Delta^2}{\varsigma^2}. \tag{2.98}$$

Next, we pass to the mutual information, by writing

$$I(\alpha; a_t) \overset{\text{(i)}}{\leq} I(\alpha; \sigma_{1:T}) \overset{\text{(ii)}}{\leq} \frac{1}{N^2} \sum_{\alpha_1, \alpha_2 \in A} D_{\mathrm{KL}}(q_{\sigma_{1:T}|\alpha_1} \mid\mid q_{\sigma_{1:T}|\alpha_2}) \overset{\text{(iii)}}{\leq} \frac{T\Delta^2}{\varsigma^2} \tag{2.99}$$

where (i) follows from the data processing inequality in two steps, namely by applying $I(\alpha; a_t) \leq I(\alpha; \sigma_{1:t}) \leq I(\alpha; \sigma_{1:T})$, (ii) uses the fact that α is uniform in order to apply the mixture bound on mutual information—one can prove this by writing out the mutual information explicitly, conditioning the right-hand-side variables on the left-hand-side variables, and applying Jensen's inequality, and (iii) applies the inequality $0 < T\Delta^2$ for the diagonal terms in the sum, and is an equality for off-diagonal terms.

Now, we apply the mutual information form of Fano's inequality for uniform α, to obtain

$$\mathbb{P}(a_t \neq \alpha) \geq 1 - \frac{I(\alpha; a_t) + \log 2}{\log N}. \tag{2.100}$$

The condition $N \geq 16$ together with the tuned value of $\Delta = \varsigma\sqrt{\frac{\log N}{4T}}$, along with the preceding mutual information bound, imply that

$$\mathbb{P}(a_t \neq \alpha) \geq \frac{1}{2}. \tag{2.101}$$

With this probability, each action incurs regret Δ. For the tuned value, we get

$$R_T(p,q) = \sum_{t=1}^{T} \Delta\, \mathbb{P}(a_t \neq \alpha) \geq \frac{\Delta T}{2} = \frac{\varsigma}{4}\sqrt{T \log N}. \tag{2.102}$$

The claim follows by setting $\varsigma^2 = 1$. ■

Proposition 2.37. *Consider a stochastic multi-armed bandit, with bounded rewards $\mathcal{R} = \{r : A \to [0,1]\}$ and standard Gaussian noise. Define the reward distribution q according to*

$$r(a) = \Delta \mathbb{1}_{a=\alpha} \qquad\qquad \alpha \sim \mathrm{U}(A). \tag{2.41}$$

Under this distribution, for any algorithm, if we suppose that $K = |A| \geq 2$ and $T \geq \frac{K}{4}$, and take $\Delta = \frac{1}{2}\sqrt{\frac{K}{T}}$, then

$$\frac{1}{8}\sqrt{KT} \leq R_T(\cdot, q). \tag{2.42}$$

Proof. Similar to above, we prove a mild generalization where the noise has variance ς^2. Given α, define $n_\alpha(t)$ to be the number of times the arm corresponding to α is pulled up to time t. Conditional on α, each round either contributes Δ or nothing to the regret. So, the regret can be written in terms of the expected number of times bad arms are pulled, which is $T - \mathbb{E}\, n_\alpha(T)$. Thus

$$R_T(\cdot, q) = \Delta(T - \mathbb{E}\, n_\alpha(T)) \tag{2.103}$$

by the Tower Rule, where the expectation is taken only over $\alpha \sim \mathrm{U}(A)$, and all other randomness is contained inside the definition of n_α. We will now analyze this term. Note first that $\Delta \leq 1$ by assumption.

Let $q_{\sigma_t|a_t,\alpha}$ denote the conditional distribution of σ_t given both a_t and α, understood as a probability kernel. We will now make a comparison between the environment with a single-best-arm reward of size Δ, and one where all rewards are zero: denote the second respective conditional distribution by $q_{\sigma_t|a_t,0}$. For a given action at a given time, we have

$$D_{\mathrm{KL}}(q_{\sigma_t|a_t,0} \mid\mid q_{\sigma_t|a_t,\alpha}) = \frac{\Delta^2}{2\varsigma^2}\mathbb{1}_{a_t=\alpha}. \tag{2.104}$$

By the chain rule for KL divergences, and using the fact that actions only depend on α through the history, we have

$$\begin{aligned} &D_{\mathrm{KL}}(q_{\sigma_{1:T},a_{1:T}|0} \mid\mid q_{\sigma_{1:T},a_{1:T}|\alpha}) && (2.105)\\ &\quad= \mathbb{E}\sum_{t=1}^{T} D_{\mathrm{KL}}(q_{\sigma_t,a_t|\sigma_{1:t-1},a_{1:t-1},0} \mid\mid q_{\sigma_t,a_t|\sigma_{1:t-1},a_{1:t-1},\alpha}) && (2.106)\\ &\quad= \mathbb{E}\sum_{t=1}^{T} D_{\mathrm{KL}}(q_{\sigma_t|\sigma_{1:t-1},a_{1:t},0} \mid\mid q_{\sigma_t|\sigma_{1:t-1},a_{1:t},\alpha}) && (2.107)\\ &\quad= \frac{\Delta^2}{2\varsigma^2}\,\mathbb{E}\, m_\alpha(T) && (2.108) \end{aligned}$$

where $m_\alpha(t)$ is the amount of times arm α would have been pulled if the true reward was zero everywhere. By Pinsker's inequality, we obtain

$$\mathbb{E}\, n_\alpha(T) - \mathbb{E}\, m_\alpha(T) \le |\mathbb{E}\, n_\alpha(T) - \mathbb{E}\, m_\alpha(T)| \tag{2.109}$$

$$\le T\sqrt{\frac{1}{2} D_{\mathrm{KL}}(q_{\sigma_{1:T},a_{1:T}|0} \mid\mid q_{\sigma_{1:T},a_{1:T}|\alpha})} \tag{2.110}$$

$$= \frac{T\Delta}{2\varsigma}\sqrt{\mathbb{E}\, m_\alpha(T)} \tag{2.111}$$

where the T appears because $0 \le n_\alpha(T) \le T$ and similar for m_α. Summing over α then gives

$$\sum_{\alpha\in A} \mathbb{E}\, n_\alpha(T) \overset{\text{(i)}}{\le} \sum_{\alpha\in A} \mathbb{E}\, m_\alpha(T) + \frac{T\Delta}{2\varsigma} \sum_{\alpha\in A} \sqrt{\mathbb{E}\, m_\alpha(T)} \tag{2.112}$$

$$\overset{\text{(ii)}}{\le} T + \frac{T\Delta}{2\varsigma}\sqrt{K \sum_{\alpha\in A} \mathbb{E}\, m_\alpha(T)} \tag{2.113}$$

$$\overset{\text{(iii)}}{=} T + \frac{T\Delta\sqrt{KT}}{2\varsigma} \tag{2.114}$$

where (i) applies the preceding Pinsker bound termwise, (ii) is a variant of Cauchy–Schwarz, and (iii) uses the identity $\sum_{\alpha\in A} m_\alpha(T) = T$ deterministically. We now apply this. Combining with the preceding bound gives

$$R_T(\cdot, q) \ge \Delta T\left(1 - \frac{1}{K}\right) - \frac{\Delta^2 T^{3/2}}{2\varsigma\sqrt{K}}. \tag{2.115}$$

Using $K \ge 2$ to bound $1 - \frac{1}{K} \ge 1/2$, and plugging in the tuned value $\Delta = \frac{\varsigma}{2}\sqrt{\frac{K}{T}}$ gives the result

$$R_T(\cdot, q) \ge \frac{\varsigma\sqrt{KT}}{4} - \frac{\varsigma\sqrt{KT}}{8} = \frac{\varsigma}{8}\sqrt{KT}. \tag{2.116}$$

■

Proposition 2.38. *Consider a stochastic multi-armed bandit, where the rewards are bounded with a gap, namely*

$$\mathcal{R} = \{r : A \to [0,1] : r(a^*) - r(a) \ge \Delta, \forall a \ne a^*\} \tag{2.43}$$

where $a^* = \arg\max_{a\in A} r(a)$, we assume $0 < \Delta \le \frac{1}{2}$, and the noise is standard Gaussian. Define the reward distribution q according to

$$r(a) = \Delta \mathbb{1}_{a=\alpha_1} + 2\beta\Delta\mathbb{1}_{a=\alpha_2} \qquad \begin{matrix} \alpha_1 \sim \mathrm{U}(A) \\ \alpha_2 \mid \alpha_1 \sim \mathrm{U}(A \setminus \{\alpha_1\}) \end{matrix} \qquad \beta \sim \mathrm{Ber}(\tfrac{1}{2}). \tag{2.44}$$

Under this distribution, for any algorithm, if we suppose that $K = |A| \ge 2$ and $\frac{\sqrt{K-1}}{T^{1/4}} \le \Delta$, then

$$\frac{(K-1)\log T}{16\Delta} \le R_T(\cdot, q). \tag{2.45}$$

Proof. Exercise 2.7. ■

Chapter 3

Expected Improvement

Under construction. Headings indicate planned content.

3.1. Bayesian Dynamic Programming

3.2. Algorithms via One-step Approximations

3.3. General Feedback and Expected Utility Improvement

Chapter 4

Gittins Indices

Under construction. Headings indicate planned content.

4.1. The Pandora's Box Decision Problem

4.2. The Gittins Index in Pandora's Box

4.3. Gittins Indices for General Decision Problems

Chapter 5

Optimism

Under construction. Headings indicate planned content.

Chapter 6

Information-theoretic Algorithms

Under construction. Headings indicate planned content.

6.1. Entropy Search

6.2. Bayesian Algorithm Execution

Chapter 7

Thompson Sampling

Under construction. Headings indicate planned content.

7.1. Concentration-based Analysis

7.2. Adversarial Analysis

7.3. Exploration in Large Language Models

Bibliography

David Blackwell. Comparison of Experiments. *Berkeley Symposium on Mathematical Statistics and Probability*, 1951. Cited on pages 6, 9.

David Blackwell. Equivalent Comparisons of Experiments. *Annals of Mathematical Statistics*, 1953. Cited on pages 6, 9.

Henrique De Oliveira. Blackwell's Informativeness Theorem Using Diagrams. *Games and Economic Behavior*, 2018. Cited on page 37.

Dylan J. Foster and Alexander Rakhlin. Foundations of Reinforcement Learning and Interactive Decision Making. *arXiv:arXiv: 2312.16730*, 2023.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. Cited on page 30.

Moritz Hardt. *The Emerging Science of Machine Learning Benchmarks*. Princeton University Press, 2026. Cited on page 29.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 1996. Cited on page 32.

Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003. Cited on page 6.

Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer, 1986. Cited on page 19.

Jeffrey Negrea, Blair Bilodeau, Nicolò Campolongo, Francesco Orabona, and Daniel M. Roy. Minimax Optimal Quantile and Semi-Adversarial Regret via Root-Logarithmic Regularizers. *Advances in Neural Information Processing Systems*, 2021. Cited on page 42.

Francesco Orabona. *A Modern Introduction to Online Learning*. Cambridge University Press, 2026. Cited on pages 42, 44.

Francesco Orabona and Dávid Pál. Optimal Non-Asymptotic Lower Bound on the Minimax Regret of Learning with Expert Advice. *arXiv:arXiv: 1511.02176*, 2015.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. Cited on page 31.

[heading=bibintoc]

Appendix

Under construction. Headings indicate planned content.

A. Mathematical Background

A.1. Probability Theory

A.2. Information Theory

A.3. Markov Decision Processes

B. Benchmarking Details