STOCHASTIC DILEMMAS: FOUNDATIONS AND APPLICATIONS

BY SERGIU GOSCHIN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Michael L. Littman and Haym Hirsh

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2014

ABSTRACT OF THE DISSERTATION

STOCHASTIC DILEMMAS: FOUNDATIONS AND APPLICATIONS

by SERGIU GOSCHIN

Dissertation Director: Michael L. Littman and Haym Hirsh

One of the significant challenges when solving optimization problems is addressing possible inaccurate or inconsistent function evaluations. Surprisingly and interestingly, this problem is far from trivial even in one of the most basic possible settings: evaluating which of two options is better when the values of the two options are random variables (a *stochastic dilemma*). Problems in this space have often been studied in the statistics, operations research and computer-science communities under the name of "multi-armed bandits". While most of the previous work has focused on dealing with noise in an online setting, in this dissertation, I will focus on offline optimization where the goal is to return a reasonable solution with high probability using a finite number of samples. I will discuss a set of problem settings of increasing complexity that allow one to derive formal algorithmic bounds. I will point to and discuss interesting connections between stochastic optimization and noisy data

annotation, a problem where the goal is to identify the label of an object from a series of noisy evaluations.

As a first contribution, I will introduce and formally analyze a set of novel algorithms that improve the state of the art and provide new insights for solving the stochastic optimization and noisy data-annotation problems. I will then formally prove a novel result: That a widely used derivative-free optimization algorithm (the cross-entropy method) is optimizing for quantiles instead of expectation in stochastic optimization settings. I will back up the theoretical claims on the optimization side with experimental results in a set of non-trivial planning and reinforcement-learning domains. Finally, I will discuss the application of the above algorithms for solving noisy data-annotation problems in a setting involving real crowdsourcing experiments.

Preface

Parts of chapters 2 and 3, are joint work with Chris Mesterharm and Haym Hirsh. Parts of chapters 4 and 5 have appeared in Goschin et al. [2012] and are joint work with Ari Weinstein, Michael Littman and Erick Chastain. The key theoretical result from chapter 6 has appeared in Goschin et al. [2011] and is joint work with Michael Littman and David Ackley. The connection to the Cross-Entropy method and the empirical results from chapter 6 have appeared in Goschin et al. [2013] and are joint work with Ari Weinstein and Michael Littman.

Acknowledgements

This dissertation would not have been possible without the support of a lot of people through the years and it is thus mostly a consequence of their tremendous help.

I would first like to thank my advisers, Michael Littman and Haym Hirsh, for their careful guidance, patience and for continuously challenging me to become a better researcher. I could not have asked for better mentors.

I am deeply grateful to Michael for teaching me how to do research and how to communicate results effectively. His ability to always find the right balance between guiding and allowing me to follow my own path never ceases to amaze me. I owe a lot to Michael for showing me the beauty of formally solving algorithmic problems, a process which had a significant and positive impact on both my professional and personal life.

I am equally grateful to Haym for teaching me the importance of creativity in research and the value of connecting seemingly disparate research areas. His pragmatic perspective combined with a patience and a belief in my abilities that I did not deserve and for which I am indebted, allowed me to grow in ways I did not foresee when I started graduate school. I owe Haym a lot for showing me how to become an applied researcher.

I would also like to thank my committee members, Swastik Kopparty and Shie Mannor for their very valuable suggestions that significantly improved this dissertation.

Among my co-authors, I would like to especially thank Tom Walsh, Chris Mesterharm and Ari Weinstein. At various stages during graduate school, Tom and Chris acted as excellent mentors and I am very grateful for the time we spent doing research together. I would also like to thank them for reviewing parts of this dissertation. Ari had an important impact on my research and I benefited in lots of ways from our frequent brainstorming sessions and from the time we spent writing papers together.

I would also like to thank Alex Borgida for his advice and direction in my first year in graduate school and Michael Saks for showing me the beauty of combinatorics.

I want to thank the members of the RL3 laboratory, Michael Wunder, John Asmuth, Monica Babeş-Vroman, Erick Chastain, Chris Mansley, Carlos Diuk for the many things I learned from them and for hanging out and debating various topics in the lab, all of which made graduate life significantly more exciting.

Last, but far from least, I would like to thank my amazing wife Simona and my great parents, Paula and Ion-Marian. I would not have been able to go through graduate school and complete this dissertation without Simona's love, patience and unconditional support. It is hard to put into words my gratitude for my parents' love and dedication to my education and well-being.

Dedication

Pentru Simona şi părinţii mei, Ion-Marian şi Paula.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

At the highest level, solving an optimization problem requires finding the argument that minimizes or maximizes a scoring (or reward) function. The goal of designing an optimization algorithm is thus to find the optimal input when given query access to the range of the function, while using as few function evaluations as possible.

Among the difficulties encountered when solving real optimization problems, we mention that: **(1)** the evaluation for a chosen input can be inconsistent and **(2)** arbitrarily delayed in the future, **(3)** the actual optimal value might not be known *a priori*, **(4)** the input space can be highly dimensional or **(5)** the output of the function to be optimized can be multimodal. Moreover, an optimization algorithm might be constrained to have: **(6)** only partial control over what input it can choose to evaluate next, **(7)** only partial visibility of the state where it transitions after one evaluation, **(8)** a limited budget for function evaluations and **(9)** limited computational resources.

Most real problems have most, if not all, of these characteristics, and designing efficient algorithms for solving them is far from trivial. Most of the challenges described above are to a large extent still topics of active research in computer science, operations research or statistics.

Zoom In: Inconsistent Evaluations.

The main focus of this dissertation will be the topic of inconsistent function

evaluations. An evaluation is **inconsistent** or **variable** or **noisy** if two evaluations of the function for the same input can yield two different outcomes.

Noisy evaluations are the rule rather than the exception in real optimization problems. Administering a drug to a population of patients can yield very different outcomes, ranging from helpful to harmful. Similarly, investing in a particular company on the stock market can yield significant gains, catastrophic losses or some intermediate results. As another example, positioning various advertisements on a search page usually leads to different click rates depending on their quality and how well they match the search queries. These optimization problems (finding the best drug for a disease, the best companies on the stock market, or the best ad for a search query) are difficult mainly because the algorithms need to take decisions in an uncertain environment.

Several models have been proposed for processes that generate noisy evaluations. In the worst case, we can imagine an adversary that adapts to our history of inputs [Auer et al., 2003]. As an example, if the goal is to win a game against a competitive player, the optimization strategy should take into account the fact that an adaptive adversary is likely to change its strategy and respond to a fixed input with different actions at different times. Another type of process that generates varying function values is a distribution that drifts in time. As an example, if the optimization problem for a major news website is to keep its main page updated with the most interesting topics, then it has to continuously track the current major events, which sometimes change every few minutes.

Zoom In: Stochastic Optimization.

In this dissertation, I focus on a simpler noise model: for a fixed input, the values are independent and identically distributed (iid) samples from some

distribution. Natural examples of such problems include situations in which the function evaluations correspond to noisy measurement of some physical quantity. Consider, for example, designing a router to minimize delays when sending packages to a particular destination reachable through several different paths. Every time a package is sent on a path, the delay can vary and modeling it as a sample from some fixed distribution is a reasonable model (at least for short time intervals).

But, if the evaluations are samples from a distribution, an important question is: What are we optimizing for? The answer is clear in a deterministic setting (that is, when the evaluations for a fixed input are always the same), but less obvious in the noisy case. Are we targeting the "best" noise distribution, the "best" expected value, the best "quantile" value or something else? These objectives are generally not aligned and, more importantly, ordering distributions does not (usually) lead to a total order.

For most of the dissertation I focus on optimizing for expectation (the standard choice in stochastic optimization) and, in Chapter 6, I will discuss optimizing for quantiles (an objective known as *Value at Risk* in mathematical finance). To make the problem formal, for some sets X, Y and a set of distributions $\mathcal{D}_{x \in X}[Y]$, let F be an evaluation or scoring function $F : X \to \mathcal{D}_{x \in X}[Y]$ accessible via samples $f(x) \sim \mathcal{D}_x[Y]$. The goal is to find:

- $x^* = \text{argmax}_{x \in X} \mathbb{E}_{\mathcal{D}_x}[F(x)]$ or / and
- $x^\theta = \text{argmax}_{x \in X} q^x(\theta)$, for a fixed $\theta \in (0, 1)$ (with $q^x(\theta)$ being the quantile functions)

while minimizing the number of samples.

To address the core issues of optimizing in noisy settings, I simplify all the

other design choices of an optimization problem except the one concerning variable function evaluations. In particular, I will mostly focus on one of the models that has been heavily studied in the past few decades in statistics and computer science: *multi-armed bandits* [Robbins, 1952]. In this model, the range of the function to be optimized is finite and "small" ($X = \{1, 2, \ldots, n\}$ with the inputs labeled *arms*). In this context, it is acceptable for the number of evaluations (which we will also call sample complexity) of an algorithm to be linear in the size of the input space X. This assumption simplifies the challenges of designing optimization algorithms, with the only difficulty that remains to be addressed being variable function evaluations. While the simplification is significant, the algorithmic insights are very helpful in more complex settings, as I will also demonstrate in this dissertation.

Most of the research in multi-armed bandits in the case of stochastic rewards focuses on **online optimization** [Auer et al., 2002], where the goal is to minimize the expected difference between the reward of a strategy and the reward of the best policy in retrospect (with this type of performance measure usually known as *minimizing the expected cumulative regret*).

We will instead focus on **offline optimization** where the goal is to return, with high probability and with finite sample complexity, the item with the highest expected value. This type of model has been formalized by Even-Dar et al. [2002] under the name of PAC-Bandits. It is motivated by applications like stochastic planning, where the standard optimization goal is to return in finite time a reasonable action recommendation to be executed in an intrinsically uncertain environment.

I will simplify the multi-armed bandit model in Chapter 2 and assume that

we only have to deal with one arm that has one of two possible expected values. The goal is to find which of these two values the arm has and I will call this problem a *Stochastic Dilemma* (which is also known in the literature as the problem of learning a biased coin). I will show in Chapter 4 that a Stochastic Dilemma is equivalent to solving a multi-armed bandit problem with $n = 2$ arms. While the problem might seem (deceptively) simple, we don't yet know of an optimal algorithm for it. Moreover, a Stochastic Dilemma is a rich source of algorithmic and analysis ideas that can be used in more complex stochastic optimization problems.

1.1 Contributions

In this section, we preview the key contributions from this dissertation. In each chapter, I informally introduce a problem, motivate it and formally describe the model for it. I then focus on the theoretical aspects and discuss upper and lower bounds and, in the applied chapters, I present the experimental results.

Chapter 2. The key contribution of Chapter 2 is a new type of adaptive algorithm (Beat-By-K)—based on the analysis of the classical ruin problem from random walks—for solving stochastic dilemmas. I show that Beat-By-K is provably better than non-adaptive strategies and that Beat-By-K is a constant multiplicative factor better than the tightest lower bound for non-adaptive algorithms. I show that Beat-By-K improves a simple adaptive version of such strategies. I also describe several related algorithmic tools introduced in statistics and computer science to solve a more difficult version of the problem.

Chapter 3. In the third chapter, I extend the algorithms from Chapter 2 to solve a more general stochastic dilemma and empirically investigate whether

they are viable annotation strategies in a crowdsourced data annotation setting involving experiments with Amazon Mechanical Turk (AMT) and Galaxy Zoo [Willett et al., 2013]. I demonstrate empirically that Beat-By-K dominates several other baseline methods for data annotation.

Chapter 4. The key contribution of the fourth chapter is a proof of how a variation of Beat-By-K is asymptotically optimal in an extension of the Probably Approximately Correct (PAC) Bandit framework [Even-Dar et al., 2002] involving an infinite number of arms. The result is interesting because it provides an alternative to algorithms inspired by Median Elimination (the only known asymptotically optimal algorithm in the PAC Bandit setting). We also discuss the relations between several multi-armed bandit models and various reduction techniques.

Chapter 5. The contribution of Chapter 5 is to generalize the infinite bandit model from Chapter 4 and extend the algorithms introduced in the preceding chapters with the goal of applying them to solve realistic stochastic planning problems. I describe experimental results in the computer games Infinite Mario and Pitfall.

Chapter 6. In Chapter 6, I prove novel theoretical results about the Cross-Entropy Method (CE), a global optimization algorithm known to have excellent empirical properties in solving a variety of difficulty real-world optimization problems. In particular, I prove that CE optimizes for quantiles in a stochastic optimization context and demonstrate that the theoretical result is consistent with empirical results in several benchmark problems. I also provide a simple modification to CE that maximizes for the expected value.

Chapter 7. I conclude the thesis in Chapter 7 with a summary of the contributions and a discussion of future work.

1.1.1 A Common Theme

Beyond the exact definitions of the models, the phenomenon of iid noise in function evaluations is a central concern in all the problems presented in the dissertation. It is thus not surprising that algorithmic techniques like Hoeffding Races [Maron and Moore, 1997] appear over and over again in various forms in stochastic optimization.

By focusing on some of the simplest possible models in the first chapters, the hope was to uncover other key algorithmic ideas that would positively impact the practice of stochastic optimization. From this perspective the Beat-By-K algorithm (Chapter 2), the Greedy Rejection algorithm (Chapter 4) and the (previously unknown) properties of the Cross-Entropy Method (Chapter 6) are the key contributions in this dissertation.

From a modeling perspective, there is a common template to the problems we discuss. At a high level, we are given sampling access to a distribution $\mathcal{P}$ over "items" and a threshold value θ as a parameter. The goal is to decide for each item sampled from $\mathcal{P}$ whether it has an expected value **above** or **below the threshold**. It is this type of **binary decision** in a **noisy** setting that motivates the title of the dissertation: **Stochastic Dilemmas**.

Concretely:

- In Chapter 2, $\mathcal{P}$ is a uniform distribution over two values, $\theta = 0.5$. and the goal is to classify an object drawn from $\mathcal{P}$ as having an expected value larger or smaller than θ.

- In Chapter 3, $\mathcal{P}$ is an arbitrary distribution over $[0, 1]$, $\theta = 0.5$ and the goal is the same as in the second chapter.

- In Chapter 4, algorithms are given sampling access to $\mathcal{P}$ (which is a categorical distribution over two values), $\theta = 0.5$ and the goal is to find an object with expected value larger than θ.

- In Chapter 5, we generalize $\mathcal{P}$ to be again an arbitrary distribution over $[0, 1]$, θ is an arbitrary value in $[0, 1]$ and the goal remains unchanged as compared to the fourth chapter.

- In Chapter 6, for a fixed iteration of the Cross-Entropy method, $\mathcal{P}$ is a distribution maintained by the algorithm over the input space as a way to balance exploration and exploitation, and θ is defined implicitly as the value that separates the "elite" sample from the rest of the current set of sampled inputs.

Distribution $\mathcal{P}$ plays the role of a natural oracle providing items to an agent. We interpret $\mathcal{P}$ as playing an analogous role to that of the target distribution over data items in the Probably Approximately Correct learning framework [Kearns and Vazirani, 1994], with the key difference that we focus on optimization as opposed to learning problems in this dissertation. We discuss the motivation and the role of $\mathcal{P}$ in detail in Sections 3.2.2 and 5.2.1.

1.1.2 Thesis Statement

Noise in function evaluations makes even the simplest optimization problems difficult. Developing correct and efficient techniques for solving stochastic dilemmas has a significant impact on improving the solving of more complex stochastic optimization problems. I will introduce novel algorithms for solving such stochastic dilemmas and prove how the analysis and algorithmic ideas can be re-used to solve more general problems.

Chapter 2
Stochastic Dilemmas

2.1 Introduction

We address the following problem: given an unfair coin, determine with high probability whether the coin is biased towards heads or tails using as few flips of the coin as possible (and we call this problem a *stochastic dilemma*). The answer can be incorrect with a certain probability as there is always a chance that the flips are not representative of the true bias. The problem is a key component for the proofs in learning theory for Probably Approximately Correct learning [Kearns and Vazirani, 1994; Anthony and Bartlett, 2009].

The standard algorithm, commonly referred to as Majority Vote, flips the coin a fixed number of times and decides the direction of the bias based on the majority label of the samples. Even though this algorithm is simple, it is often used in practice. For example, it is the baseline method used for categorical data annotation [Sheng et al., 2008].

Most of the previous theoretical work focused on obtaining lower bounds on the number of flips necessary for solving the problem [Simon, 1993; Canetti et al., 1995; Ben-David and Lindenbaum; Baxter, 2000; Anthony and Bartlett, 2009]. While the bounds are tight asymptotically, a lot of effort has been put into improving the constant factors.

Almost all known results assume that the number of samples is fixed *a priori* as a function of the model parameters (that is the bias of the coin and the desired failure probability), and lower bounds are proven using information-theoretic arguments. The only exception we are aware of is a result by Mannor et al. [2004] who extended the lower bound for adaptive strategies (that is algorithms that make decisions based on the history of samples). One (implicit) open question is whether such adaptive strategies, while more general, offer an advantage with respect to non-adaptive algorithms like Majority Vote. This question is resolved in the affirmative in this chapter.

Contributions. The main contribution from this chapter is to propose a new type of adaptive algorithm (that we call Beat-By-K) based on the classical ruin problem from random walks. We show that Beat-By-K is provably better than non-adaptive strategies and that Beat-By-K is a constant multiplicative factor better than the tightest lower bound for non-adaptive algorithms. We also show that the performance of Beat-By-K exceeds that of a simple adaptive version of such strategies.

We also examine a set of algorithms for solving a stochastic dilemma in a setting where the bias of the coin is unknown. This version of the problem is intrinsically harder and we will need different types of strategies that are able to adapt to the unknown parameter. We will unify similar algorithmic ideas introduced in various contexts: model selection in machine learning [Maron and Moore, 1997], confidence sequences in statistics [Robbins, 1970] and multi-armed bandits [Even-Dar et al., 2002].

Both types of algorithms are used as building blocks for designing strategies for more complex problems in the rest of the dissertation. From this perspective, even if the model we work with in this chapter is simple, it captures

some of the fundamental characteristics of realistic noisy optimization, learning and data annotation problems.

2.2 Model

In this section, we will formally define the simplest version of a stochastic dilemma. Let us also assume that for a fixed $p \in [0, 0.5)$ we are given a set of distributions

$$D = \{\mathcal{D}_0 = \text{Bernoulli}(p), \mathcal{D}_1 = \text{Bernoulli}(1-p)\} \tag{2.1}$$

Let us also assume that we are given access to samples from an unknown distribution that was chosen according to a uniform distribution over D ($\mathcal{D} \in D$). Finally, let us assume we are given an accepted failure probability parameter $\delta \in (0, 0.5)$.

The stochastic dilemma problem is to estimate with failure probability at most δ the expected value of the distribution $\mathcal{D}$, $\mathbb{E}_{x\sim\mathcal{D}}[x]$, while minimizing the (expected) number of samples $m(p, \delta)$ from $\mathcal{D}$. In addition, since the expected value of $\mathcal{D}$ can only have two values, we constrain solutions to this problem to be one of these two values.

We note that since the empirical average of a Bernoulli distribution is a sufficient statistic for the parameter of the distribution, the problem above is equivalent to deciding based on random samples whether $\mathcal{D} = \text{Bernoulli}(p)$ or $\mathcal{D} = \text{Bernoulli}(1-p)$, hence the label of *stochastic dilemmas*. For ease of notation, we will label this problem $\text{SD}(p, \delta)$.

2.3 Known Parameters

In this chapter, we study two versions of the problem. In the first part (this section), the value of the parameter p is assumed known. As we will see, this assumption leads to an easier problem than when p is unknown and impacts the type of performance one can get in practice in a variety of cases. Assuming p is known is of course unrealistic in most practical settings. The assumption is reasonable because in practice it is often possible to estimate p from previous samples, and then use an approximation of it afterwards. Thus, designing algorithms that take advantage of such knowledge is of both practical and theoretical interest.

2.3.1 Majority Vote

We will use two variants of a baseline method that we call **Majority Vote** for solving a stochastic dilemma.

Given a budget $m(p, \delta)$ (m odd, to be determined as a function of the parameters), **Majority Vote** obtains m samples $x_i \sim \mathcal{D}, i \in [m]$. It then returns $\mathcal{D} = \mathcal{D}_0$ if the empirical average $\hat{x} = \frac{\sum_{i=1}^{m} x_i}{m} < 0.5$ and $\mathcal{D} = \mathcal{D}_1$ otherwise.

Early Majority Vote is an optimized version of Majority Vote that seeks labels[1] incrementally and stops once either label appears at least $\frac{m+1}{2}$ times, since after that point additional labels will not change the outcome of the majority vote.

[1] We will use the terms "labels" and "samples" interchangeably in the dissertation.

Algorithm 1: Majority Vote (p, δ):

1. Take $m = \lceil \frac{2}{(1-2p)^2} \log \frac{2}{\delta} \rceil$ samples $x_i \sim \mathcal{D}, i \in [m]$ and let $\hat{x} = \frac{\sum_{i=1}^{m} x_i}{n}$.
2. If $\hat{x} < 0.5$ return $\mathcal{D}_0$ otherwise return $\mathcal{D}_1$.

Approaches similar to Majority Vote approaches have been repeatedly studied in the literature. Majority Vote is one of the the baseline techniques used in categorical data annotation [Sheng et al., 2008]. The reader is referred for example to Yang and Carbonell [2009] for a detailed analysis of a general version of majority vote and its upper bounds. Nevertheless, for completeness, we will give a proof for an upper bound on the sample complexity for the version of the algorithm described above.

Before proving the main theorem from this section, we will first state a fact about the relation between Majority Vote and Early Majority Vote:

Fact 2.3.1. *For any $SD(p, \delta)$ problem, and a fixed parameter m, Majority Vote(m) has the same failure probability as Early Majority Vote(m).*

Proof. Consider an arbitrary sequence of samples of length m. And consider that Early Majority Vote takes m samples, but ignores all samples after a majority of 0 or 1 labels is formed (this change only affects the sample complexity of the algorithm, not its decision about the identity of $\mathcal{D}$).

The decisions of Majority Vote and Early Majority Vote with respect to the identity of $\mathcal{D}$, conditioned on that sequence actually happening, are of course identical. Then, the failure probabilities of the two strategies for such a sequence are also the same. Since this fact applies to any sequence, it means that the failure probability over all possible sequences is identical for both strategies. □

Theorem 2.3.2. *If the (Early) Majority Vote algorithm takes* $m = \lceil \frac{2}{(1-2p)^2} \log \frac{2}{\delta} \rceil = O(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$ *samples, it will correctly solve a* $SD(p, \delta)$ *problem.*

Proof. We will show that choosing m as in the theorem statement is a sufficient condition for solving the problem. Let q be the unknown parameter of $\mathcal{D}$. If we apply the Hoeffding's inequality A.1.1 with $\alpha = 0.5 - p$, we get:

$$P(\hat{x} \notin (q - 0.5 + p, q + 0.5 - p)) \leq 2e^{\frac{(1-2p)^2}{2}m} \tag{2.2}$$

If we constrain the error probability to be $2e^{\frac{(1-2p)^2}{2}m} \leq \delta$, and we solve for m, we get the relation from the theorem statement. To clarify the meaning of this inequality, let's assume $q = p$ (the case $q = 1 - p$ is similar). Then the inequality 2.2 states that the probability that $\hat{x} \geq 0.5$ (which would lead to a wrong decision given the definition of the algorithm) is at most δ (as desired).

□

Several observations are in order at this point:

- The bound in Theorem 2.3.2 is sufficient but not necessary. This fact is a consequence of the implicit looseness of applying Hoeffding's inequality. As it will be shown in Section 2.3.3, there is a gap between the lower bound and the upper bound for Majority Vote.

- It is worth noting that if $\delta \leq p$, then $m = 1$ is a sufficient condition for solving the problem, since the probability that $\hat{x} = x_1$ is on the "wrong" side of 0.5 is at most p. So, the bound in the theorem statement is informative only when $\delta < p$.

2.3.2 Beat-By-K

In this section, we will introduce a new algorithm, **Beat-By-K** (BBK), that can be analyzed using random walk techniques. The key difference between **Beat-By-K** and **Majority Vote** is that the number of samples BBK takes is a random variable and not a deterministic quantity. BBK is thus part of a larger class of strategies, and we will actually show in the following section that BBK strictly dominates strategies like Majority Vote that stop after a predefined number of samples.

To define the algorithm, let $\Delta^t = \#1\text{-samples} - \#0\text{-samples}$ be the difference between the number of 1-samples and the number of 0-samples from distribution $\mathcal{D}$ after $t > 0$ labels have been obtained. The Beat-By-K strategy takes one parameter $k(p, \delta)$ (to be determined) and repeatedly seeks labels for a given instance until the first time t when either $\Delta^t = -k$ (in which case the algorithm returns $\mathcal{D}_0$), or $\Delta^t = k$ (when it returns $\mathcal{D}_1$ as an answer).

Algorithm 2: Beat-By-K (p, δ):

- Given a parameter $k = \lceil \frac{\log \frac{1-\delta}{\delta}}{\log \frac{1-p}{p}} \rceil$, for every $t = 1, 2, \ldots$:
  1. Sample $x_t \sim \mathcal{D}$, update $\Delta^t = \#1\text{-samples} - \#0\text{-samples}$.
  2. Stop if either $\Delta^t = k$ (and recommend $\mathcal{D}_1$) or $\Delta^t = -k$ (and recommend $\mathcal{D}_0$) or continue otherwise.

The formal properties of Beat-By-K are stated in the following theorem:

Theorem 2.3.3. *The Beat-By-K algorithm with parameter* $k(p, \delta) = \lceil \frac{\log \frac{1-\delta}{\delta}}{\log \frac{1-p}{p}} \rceil = O(\frac{1}{1-2p} \log \frac{1}{\delta})$ *will return the correct answer to a* $SD(p, \delta)$ *problem with probability at least* $1 - \delta$ *and it will take* $m(p, \delta) \leq \frac{1}{(1-2p)^2} \log \frac{1-\delta}{\delta} + \frac{1}{1-2p} = O(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$ *samples in expectation.*

Proof. We can interpret the process of obtaining labels as a random walk on the number line with two absorbing barriers (at $-k$ and at k). The problem of deciding the identity of $\mathcal{D}$ is thus equivalent to a random walk, but with two possible (and hidden to the algorithm) probability distributions—one for a Bernoulli(p) distribution and the other for the Bernoulli$(1-p)$ instance—which bias it in opposite ways. Formally, Δ^t is either a positively biased random walk (with bias $1-2p$) or a negatively biased random walk (with bias $2p-1$). For more details about this interpretation the reader is referred to the transformations before corollary A.2.2 in Appendix A.2.

The accuracy of the Beat-By-K strategy can be described as:

$$
\begin{aligned}
f(k) &= P(k\text{-strategy is accurate}) && (2.3)\\
&= P(\Delta^\tau = -k | \mathcal{D} = \text{Bernoulli}(p))P(\mathcal{D} = \text{Bernoulli}(p)) + && (2.4)\\
&\quad P(\Delta^\tau = k | \mathcal{D} = \text{Bernoulli}(1-p))P(\mathcal{D} = \text{Bernoulli}(1-p))\\
&= 0.5P(\Delta_1^\tau = -k | \text{Bernoulli}(p)) + 0.5P(\Delta_1^\tau = k | \text{Bernoulli}(1-p)) && (2.5)\\
&= P(\Delta_1^\tau = k | \text{Bernoulli}(1-p)) && (2.6)\\
&= \frac{\beta^k}{\beta^k + 1} && (2.7)
\end{aligned}
$$

with $\beta = \frac{1-p}{p}$ and where Equality 2.6 is obtained by using the fact that the random walks are symmetric and thus have identical probabilities of reaching the barrier in the direction in which they are biased. Now, replacing $k_0 = k_1 = k$ in Theorem A.2.2(i), we get that $P(\Delta_1^\tau = k | \text{Bernoulli}(1-p)) = \frac{\beta^k}{\beta^k+1}$.

In the worst case, we would need to round k to $k' = \frac{\log \frac{1-\delta}{\delta}}{\log \frac{1-p}{p}} + 1$ to make sure the failure probability is as desired (since we can only pick discrete values for k). Now, if we replace this upper bound of k with the value from the theorem statement then $f(k') \geq 1-\delta$, which proves that the algorithm is correct with probability at least $1-\delta$.

Let $\tau(k')$ represent the number of labels obtained for an instance using the Beat-By-K strategy. Using Theorem A.2.2(ii) and again the property that the two random walks are symmetric we get:

$$\tau(k) \leq \tau(k') = \frac{k'(\beta^{k'} - 1)}{(1-2p)(\beta^{k'} + 1)} \tag{2.8}$$

$$\leq \frac{k'}{1-2p} \tag{2.9}$$

$$\leq \frac{\log \frac{1-\delta}{\delta}}{\log \frac{1-p}{p}(1-2p)} + \frac{1}{1-2p} \tag{2.10}$$

$$\leq \frac{1}{(1-2p)^2} \log \frac{1-\delta}{\delta} + \frac{1}{1-2p} \tag{2.11}$$

where for the last inequality we used the fact that $1/\log \frac{1-p}{p} \leq \frac{1}{1-2p}$. It is thus clear that $\tau(k) = O(\frac{1}{(1-2p)^2} \log \frac{1}{1-2p})$. □

2.3.3 Lower Bounds

The main goal in this section is to investigate the lower bounds on the sample complexity for solving stochastic dilemmas when the parameter p is known. We will prove that the two algorithms introduced in the previous sections are asymptotically optimal. The more interesting result is that when we analyze the two classes of strategies (that contain non-adaptive strategies like Majority Vote and adaptive strategies like Beat-By-K), we can prove a constant gap between the lower bound for Majority Vote and the upper bound for Beat-By-K, which formally shows that the latter is the dominant strategy (and this result is the key contribution in this chapter). The proofs also point to a connection with the literature on Probably Approximately Correct Multi Armed Bandits [Even-Dar et al., 2002] that will be explored in depth in Chapter 4.

The first theorem states a lower bound on the sample complexity for algorithms that always take a fixed number of samples. We call these algorithms

non-adaptive. The proof of theorem 2.3.4 is obtained by replacing the variables from Anthony and Bartlett [2009] in Lemma 5.1 like so: $\epsilon = 1 - 2p$ and $\delta = \delta$. The argument of identifying the type of the coin remains the same.

Theorem 2.3.4 (Anthony and Bartlett [2009])**.** *Any non-adaptive algorithm that correctly solves a $SD(p, \delta)$ must take at least $m(p, \delta)$ samples from distribution $\mathcal{D}$ where:*

$$m(p, \delta) = 2\lfloor \frac{1 - (1-2p)^2}{2(1-2p)^2} \log \frac{1}{8\delta(1-2\delta)} \rfloor = \Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta}) \tag{2.12}$$

As mentioned above, one issue with the lower bound is that it only applies to algorithms like Majority Vote but not to algorithms like Early Majority Vote or Beat-By-K, that have potentially different sample complexities for different runs. So, we need a more general lower bound, but this time on *expected* sample complexity (which contains such strategies as well):

Theorem 2.3.5. *Any algorithm that correctly solves a $SD(p, \delta)$ must take at least $m(p, \delta)$ samples from $\mathcal{D}$ on expectation, where:*

$$m(p, \delta) = \frac{C}{(1-2p)^2} \log \frac{1}{\delta} = \Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta}), \text{ for some constant } C > 0 \tag{2.13}$$

Proof. The proof is based on a reduction to a PAC-Bandit lower bound from Mannor et al. [2004]. Let us assume there exists an algorithm $A(p, \delta)$ that solves a stochastic dilemma with parameters p and δ in $o(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$. Consider a 2-armed bandit problem (label the arms $a1$ and $a2$) with known difference between the expected rewards of the best and the worst arms equal to $2(1 - 2p)$. Using the notation from Section 7 from Mannor et al. [2004], let $q^* = 1 - p$ and choose $\epsilon = \frac{1-2p}{2}$.

We can solve the 2-armed bandit problem in the following way: (1) Apply algorithm A for arm a_1 and then, (2) If A returns the distribution with a smaller

expected value, return arm a_2 as being the optimal arm. Otherwise return a_1. This strategy will have expected sample complexity $m = o(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$ since all it does is execute $A(p, \delta)$.

It is known from Theorem 13 of Mannor et al. [2004] that $m = \Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$. We remark that the result holds for the special case of 2 arms, which is what we need for the reduction.

But, we know from Theorem A.3.1 (applied to functions $m(p, \delta)$ and $g(p, \delta) = \frac{1}{(1-2p)^2} \log \frac{1}{\delta}$ for the complexity parameters $\frac{1}{\delta}$ and $\frac{1}{1-2p}$) that the two sets functions $o(g)$ and $\Omega(g)$ are disjoint and we thus get the desired contradiction. □

2.3.4 Beat-By-K vs (Early) Majority Vote

While the two lower bounds are asymptotically tight, there is a natural question as to whether we can get any benefit by relaxing the class of strategies from non-adaptive to *adaptive* algorithms, that is algorithms that take decisions based on the history of samples. A trivial positive answer to this question is that Early Majority Vote (which is, by definition, an adaptive strategy) has an expected sample complexity that is upper bounded by the performance of Majority Vote. The answer is not completely satisfactory because, as we will show, the expected ratio between the sample complexity of Early Majority Vote and Majority Vote actually converges to 1 as $p \nearrow 0.5$ (p approaches 0.5 from below) and $\delta \searrow 0$ (δ approaches 0 from above). It is thus the case that for *hard* Stochastic Dilemma problems the benefit of Early Majority Vote is insignificant. We consider a Stochastic Dilemma problem *hard* when p is "close" to 0.5 and δ is "close" to 0 due to the lower bound on sample complexity from theorem 2.3.5.

However, there exists, a class of strategies that provably gain a constant

multiplicative factor with respect to the lower bound on non-adaptive algorithms. We will prove in this chapter that the upper bound on the sample complexity of Beat-By-K is a constant factor smaller than the lower bound for Majority Vote.

We begin this section by proving that, for large enough p and small enough δ, the ratio between a lower bound on the sample complexity of Majority Vote and an upper bound on the sample complexity of Beat-By-K is larger than 1.

Theorem 2.3.6. *For any $p \geq 0.49$ and $\delta \leq 0.01$, the expected sample complexity of Beat-By-K for solving a $SD(p, \delta)$ problem is smaller than the lower bound on sample complexity for Majority Vote.*

Proof. We know from Theorem 2.3.4 that a lower bound on the sample complexity of Majority Vote is $m_0 = 2\lfloor \frac{1-(1-2p)^2}{2(1-2p)^2} \log \frac{1}{8\delta(1-2\delta)} \rfloor$ and is valid for $\delta \leq 0.25$. The value m_0 is discrete and thus a valid lower bound on m_0 is $m_0 \geq 2(\frac{1-(1-2p)^2}{2(1-2p)^2} \log \frac{1}{8\delta(1-2\delta)} - 1)$. Since $\delta \leq 0.01$, it follows that $\log \frac{1}{8\delta(1-2\delta)} > 2$ and then:

$$m_0 > m' = \frac{1-2(1-2p)^2}{(1-2p)^2} \log \frac{1}{8\delta(1-2\delta)} \text{ with } m' \in \mathbb{R}. \tag{2.14}$$

We know from the proof of Theorem 2.3.3 that the accuracy of BBK is $f(k) = \frac{\beta^k}{\beta^k+1}$ with $\beta = \frac{1-p}{p}$ and the expected sample complexity is $\tau(k) = \frac{k(\beta^k-1)}{(1-2p)(\beta^k+1)}$. The key idea of Beat-By-K is to compute the smallest k such that $f(k) \geq 1-\delta$ (since such a k corresponds to the smallest expected sample complexity as well).

We pick $k_0 = \lceil \frac{\ln \frac{1-\delta}{\delta}}{\ln \frac{1-p}{p}} \rceil = \lceil \log_\beta \frac{1-\delta}{\delta} \rceil$. We can only pick k_0 to be discrete, so in the worst case we would need to round k_0 to the closest highest integer (so that we can still guarantee the minimal failure probability). To satisfy the worst case scenario, we will pick $k' = \log_\beta \frac{1-\delta}{\delta} + 1$ with $k' \in \mathbb{R}$. Since f is increasing,

$f(k_0) = 1 - \delta < f(k')$, so Beat-By-K with k' as a parameter correctly solves the SD(p, δ) problem. Moreover, the expected sample complexity corresponding to k' is $\tau(k') = \frac{log_\beta \frac{\beta(1-\delta)}{\delta}}{1-2p} \frac{\beta(1-\delta)-\delta}{\beta(1-\delta)+\delta} > \tau(k_0)$.

Since $\beta > 1$, $\frac{1}{\beta(1-\delta)+\delta} < 1$. Since $p > 0.49$, $\beta < \frac{51}{49}$. So, we get that:

$$\tau(k_0) < \tau(k') < \tau' = \frac{\log(\frac{51}{49}\frac{1-\delta}{\delta})}{\log \frac{1-p}{p}(1-2p)}(\frac{51}{49}(1-\delta) - \delta) \tag{2.15}$$

Now, let $R(p, \delta) = \frac{m'}{\tau'}$:

$$R(p,\delta) = \frac{(1-2(1-2p)^2)\ln\frac{1-p}{p}}{1-2p} \frac{\ln \frac{1}{8\delta(1-2\delta)}}{(\frac{51}{49}(1-\delta)-\delta)\ln(\frac{51}{49}\frac{1-\delta}{\delta})} = R_1(p)R_2(\delta) \tag{2.16}$$

We want to show that $R(p, \delta) > 1$ for "most" p and δ values.

Next, $R_1(p) = \frac{(1-2(1-2p)^2)\ln\frac{1-p}{p}}{1-2p}$ is a strictly increasing function for $p \in [0, 0.5)$. Let's take $R_1(p = 0.49) > 1.99$. Then for any $p \geq 0.49, R_1(p) > 1.99$.

Next, $R_2(\delta) = \frac{\ln \frac{1}{8\delta(1-2\delta)}}{(\frac{51}{49}(1-\delta)-\delta)\ln(\frac{51}{49}\frac{1-\delta}{\delta})}$ is a strictly decreasing function for $\delta \in (0, 0.2)$. Take $R_2(\delta = 0.01) > 0.53$. Then, for any $\delta \leq 0.01, R_2(\delta) > 0.53$.

So, for any combination of $p \geq 0.4$ and $\delta \leq 0.01$, $R(p, \delta) = \frac{m'}{\tau'} > 1.05 > 1$. But, from Equations 2.14 and 2.15 we get that $R(p, \delta) < \frac{m_0}{\tau(k_0)}$ and thus we have shown that $m_0 > \tau(k_0)$ for any $p \geq 0.49$ and $\delta \leq 0.01$. □

Our goal was to find some constants p_0 and δ_0 such that Beat-By-K dominates Majority Vote for any $p \geq p_0$ and $\delta \leq \delta_0$. We did not try to find the best values of p_0 and δ_0. It is possible to prove a stronger result: For any constant factor $\alpha < 2$, there exists a large class of domains such that the sample complexity of Majority Vote is at least α times larger than the expected sample complexity of Beat-By-K.

Theorem 2.3.7. *For any $\alpha < 2$, there exists $p_0 < 0.5$ and $\delta_0 > 0$ such that for any $SD(p, \delta)$ with $p > p_0$ and $\delta < \delta_0$, the ratio between the lower bound on sample complexity for Majority Vote and the upper bound on expected sample complexity of Beat-By-K is at least α.*

Proof sketch. The proof uses the same quantity for $m' < m_0$. But, instead of using the upper bound $\tau' > \tau(k') > \tau(k_0)$ we will work directly with $\tau(k')$. We then define $R(p, \delta) = \frac{m'}{\tau(k')}$. We then compute $\lim_{p \nearrow 0.5, \delta \searrow 0} R(p, \delta) = 2 > \alpha$ which implies the result. □

In the rest of the section, we will show how we can use the proof of Theorem 2.3.6 to prove that Beat-By-K is also better than Early Majority Vote. Since Early Majority Vote is an adaptive strategy, we cannot directly use the lower bound from Theorem 2.3.4.

It is worth noting that because of the magnitude of the constants in the proof of Theorem 2.3.7 (that we used to prove that $R(p, \delta) > 1$), we cannot directly assume that the expected sample complexity of Early Majority Vote is at most 2 times better than the lower bound on Majority Vote. Since $R(p, \delta) < 2$, the ratio between the lower bound for Early Majority Vote and the upper bound of Beat-By-K would be smaller than 1 which is not enough to show that Beat-By-K dominates Early Majority Vote.

The high level strategy of the proof is to show that, for large enough p and small enough δ, Early Majority Vote(m) (for some fixed parameter m that depends on p and δ[2]) gains much less than a factor of 2 in terms of its expected sample complexity as compared to Majority Vote(m). We will actually prove that in the limit the ratio between the performance of Early Majority Vote and

[2] Sometimes we will explicitly state the dependency on p and δ and sometimes we will just use m as a parameter to increase readability

Majority Vote converges to 1. We thus obtain a stronger result: There is no constant factor α such that the ratio between the performance of Majority Vote and Early Majority Vote is larger than α for all hard Stochastic Dilemma problems.

Theorem 2.3.8. *For any $\alpha > 1$, there exists $p_0 < 0.5$ and $\delta_0 > 0$ such that for all $SD(p, \delta)$ with $p > p_0$ and $\delta < \delta_0$, the ratio between the sample complexity of Majority Vote($m(p, \delta)$) and the expected sample complexity of Early Majority Vote($m(p, \delta)$) is at most α.*

Proof. Let m_E be a random variable denoting the sample complexity of Early Majority Vote with parameter m (odd) for solving a SD(p, δ) problem. The parameter m corresponds to the parameter of a Majority Vote strategy with an identical failure probability (see Fact 2.3.1). Let $\gamma \in (0.5, 1)$ be a constant that we will fix later in the proof. Thus:

$$\mathbb{E}[m_E] = \sum_{t=\frac{m+1}{2}}^{m} tP(m_E = t) \tag{2.17}$$

$$= \sum_{t=\frac{m+1}{2}}^{\gamma m} tP(m_E = t) + \sum_{t=\gamma m+1}^{m} tP(m = t) \tag{2.18}$$

$$\geq \frac{m+1}{2} P(m_E \leq \gamma m) + (\gamma m + 1) P(m_E > \gamma m) \tag{2.19}$$

$$= \gamma m + 1 - P(m_E \leq \gamma m)(\gamma m + 1 - \frac{m+1}{2}) \tag{2.20}$$

where the last equality holds because $P(m_E \leq \gamma m) + P(m_E > \gamma m) = 1$.

We will now upper bound $P(m_E \leq \gamma m) = \sum_{t=\frac{m+1}{2}}^{\gamma m} P(m_E = t)$. The event "$m_E = t$" is equivalent to the event "Early Majority Vote stops at step t", which in turn is included in the event "$\frac{m+1}{2}$ samples out of the current total number of samples t are 0 or 1". Assume that the distribution $\mathcal{D} = \text{Bernoulli}(1-p)$ (the other case is treated similarly) and let $\hat{x}_t$ be the average of the samples at

step t. Then:

$$P(m_E = t) \leq P(\frac{m+1}{2} \text{ out of } t \text{ samples are 0 or 1}) \tag{2.21}$$

$$\leq 2P(\frac{m+1}{2} \text{ out of } t \text{ samples are 1}) \tag{2.22}$$

$$\leq 2P(\hat{x}_t = \frac{m+1}{2t}) \tag{2.23}$$

$$\leq 2P(\hat{x}_t \geq 1 - p + (\frac{m+1}{2t} - 1 + p)) \tag{2.24}$$

$$\leq 2e^{-2(\frac{m+1}{2t} - 1 + p)^2 t} \tag{2.25}$$

The second inequality follows from the union bound and captures the idea that the probability of sampling $\frac{m+1}{2}$ 1's is higher than seeing $\frac{m+1}{2}$ 0's (since $\mathcal{D} =$ Bernoulli$(1-p)$ with $p < 0.5$). The last inequality follows by an application of the Hoeffding inequality (Theorem A.1.1(i)). Then:

$$P(m_E \leq \gamma m) \leq 2 \sum_{t=\frac{m+1}{2}}^{\gamma m} \frac{1}{e^{\frac{(m+1-2t(1-p))^2}{2t}}} \leq \frac{2(\gamma m - \frac{m+1}{2})}{e^{\frac{(m+1-2\gamma m(1-p))^2}{2\gamma m}}} \leq \frac{m(2\gamma - 1)}{e^{m\frac{(2(1-p)\gamma - 1)^2}{2\gamma}}} \tag{2.26}$$

Now, we can replace the bound on $P(m_E \leq \gamma m)$ in Equation 2.20 and dividing both sides by m we get: $\frac{\mathbb{E}[m_E]}{m} \geq \gamma + \frac{1}{m} - \frac{(2\gamma-1)(\gamma m + 1 - \frac{m+1}{2})}{e^{m\frac{(2(1-p)\gamma-1)^2}{2\gamma}}}$ or alternatively:

$$\frac{m}{\mathbb{E}[m_E]} \leq \frac{1}{\gamma + \frac{1}{m} - \frac{(2\gamma-1)(\gamma m + 1 - \frac{m+1}{2})}{e^{m\frac{(2(1-p)\gamma-1)^2}{2\gamma}}}} \tag{2.27}$$

Since from Theorem 2.3.4, $m = m(p, \delta) = \Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$, we get that $\lim_{p \nearrow 0.5, \delta \searrow 0} m(p, \delta) = \lim_{m \to \infty} m = \infty$ and thus:

$$\lim_{p \nearrow 0.5, \delta \searrow 0} \frac{m(p,\delta)}{\mathbb{E}[m_E]} \leq \lim_{p \nearrow 0.5, \delta \searrow 0} \frac{1}{\gamma + \frac{1}{m} - \frac{(2\gamma-1)(\gamma m + 1 - \frac{m+1}{2})}{e^{m\frac{(2(1-p)\gamma-1)^2}{2\gamma}}}} \tag{2.28}$$

$$= \lim_{m \to \infty} \frac{1}{\gamma + \frac{1}{m} - \frac{(2\gamma-1)(\gamma m + 1 - \frac{m+1}{2})}{e^{m\frac{(\gamma-1)^2}{2\gamma}}}} \text{ (as } 1 - p \to 0.5) \tag{2.29}$$

$$= \frac{1}{\gamma} \tag{2.30}$$

We note that it is indeed necessary for γ to be strictly smaller than 1 for the limit to have the value $\frac{1}{\gamma}$.

Setting $\alpha = \frac{1}{\gamma} > 1$ (which can be chosen to be arbitrarily close to 1), we get the desired result based on the limit proven above. □

We can now state the result regarding the comparison between Beat-By-K and Early Majority Vote:

Theorem 2.3.9. *For any $\alpha < 2$, there exists $p_0 < 0.5$ and $\delta_0 > 0$ such that for any $SD(p, \delta)$ with $p > p_0$ and $\delta < \delta_0$, the ratio between the lower bound on the sample complexity of Early Majority Vote and the upper bound on expected sample complexity of Beat-By-K is at least α.*

The proof is a combination of the Theorems 2.3.7 and 2.3.8.

Proof. Denote by $m_{\text{LB}}(p, \delta)$ the lower bound on the sample complexity of Majority Vote for a SD(p, δ) problem. Similarly, let $m_{\text{BBK}}(p, \delta)$ and $m_{\text{EMAJ}}(p, \delta)$ be the expected sample complexity of the optimal Beat-By-K (and respectively the optimal Early Majority Vote) algorithm for a SD(p, δ) problem. Let m be the parameter of the Early Majority Vote strategy.

From Theorem 2.3.8, we know that $\forall \alpha_1 > 1, \exists p_1$ and δ_1 such that $m \leq \alpha_1 m_{\text{EMAJ}}(p, \delta)$ for all $p \geq p_1, \delta \leq \delta_1$.

From Theorem 2.3.7, we know that $\forall \alpha_2 < 2, \exists p_2$ and δ_2 such that $m_{\text{LB}}(p, \delta) \geq \alpha_2 m_{\text{BBK}}(p, \delta)$ for all $p \geq p_2, \delta \leq \delta_2$.

We now claim that $m_{\text{LB}}(p, \delta) \leq m$, that is the lower bound on the optimal parameter for the Majority Vote algorithm for a fixed SD(p, δ) problem is a lower bound for the parameter m of any parameter of an Early Majority Vote algorithm that correctly solves the same Stochastic Dilemma. We can prove this assertion by contradiction: Assume that $\exists m_0 < m_{\text{LB}}$ a parameter for Early

Majority Vote that solves the Stochastic Dilemma. Based on Fact 2.3.1, Majority Vote(m_0) has failure probability identical to Early Majority Vote(m_0), which is $\leq \delta$ (since Early Majority Vote(m_0) correctly solves SD(p, δ)). But, the sample complexity of Majority Vote(m_0) is $m_0 < m_{\text{LB}}$ while correctly solving the SD(p, δ) problem, which contradicts the assumption that m_{LB} is a lower bound for Majority Vote.

Putting all the facts together, we get that for all $p \geq \max(p_1, p_2)$ and $\delta \leq \min(\delta_1, \delta_2)$:

$$\alpha_2 m_{\text{BBK}}(p, \delta) \leq m_{\text{LB}}(p, \delta) \leq m \leq \alpha_1 m_{\text{EMAJ}}(p, \delta) \tag{2.31}$$

Now, for any $\alpha < 2$, let $\alpha_2 = \frac{\alpha+2}{2} < 2$ and $\alpha_1 = \frac{\alpha+2}{2\alpha} > 1$ and using relation 2.31, we get that $\frac{m_{\text{EMAJ}}(p,\delta)}{m_{\text{BBK}}(p,\delta)} \geq \frac{\alpha_2}{\alpha_1} = \alpha$. □

2.4 Unknown Parameters

When the parameter p is unknown, the stochastic dilemma problem naturally becomes more difficult. Due to our initial assumption that $p < 0.5$, the problem is still solvable in finite time for any failure probability δ (as we will see in the rest of this section).

It is important to note that we are departing from the standard way this type of setting is studied. In similar problems (see the PAC Bandit model [Even-Dar et al., 2002]), there is usually an extra accuracy parameter $\epsilon \in (0, 1)$ that establishes a limit with respect to which the difference between the two distributions is relevant (in other words, if the difference $1 - 2p < \epsilon$, any answer to the Stochastic Dilemma problem is acceptable). We will also study this extension (in a more general setting) in the following chapter (see Section 3.2.1

for a detailed discussion on the role of ϵ), but we chose to ignore ϵ (or alternatively set $\epsilon = 0$) for the Stochastic Dilemma problem. The reason is that we wanted to design algorithms that scale with the natural complexity parameters of the problem (which are p and δ) and study their theoretical properties. As will become apparent in the following chapters, simplifying the problem in this manner helps emphasize a type of algorithmic strategy that will prove very useful (both theoretically and empirically) in a variety of more complex settings.

2.4.1 Hoeffding Rejection

The **Hoeffding Rejection** (HR) algorithm is inspired by the Hoeffding Races [Maron and Moore, 1997] algorithm introduced in the context of model selection. Hoeffding Races has received increasing attention in the past few years in the machine-learning community in the context of optimal stopping [Mnih et al., 2008a] or policy search in reinforcement learning [Heidrich-Meisner and Igel, 2009]. A very similar algorithm was introduced by Even-Dar et al. [2002] under the name Successive Elimination in the context of finite PAC Bandits.

The idea of the algorithm is the build increasingly tight confidence intervals based on the Hoeffding inequality (hence the algorithm's name). The high level goal is to define the intervals in such a way that the true expected value of $\mathcal{D}$ is contained in the current interval after every sample. Then, the algorithm stops as soon as 0.5 falls outside the confidence interval (which is guaranteed to happen in finite time) and recommends the option that is on the same "side" of 0.5 as the last empirical average.

One difference with the Hoeffding Races algorithm introduced by Maron and Moore [1997] is that we do not know the number of steps ("models" in

their setup) *a priori*. The reason is that the number of steps until "stopping" (defined as the moment when the confidence interval doesn't contain 0.5 anymore) depends on p (as we will soon see), which is unknown. Moreover, since the algorithm can "fail" at any step, it is an adaptive strategy as opposed to a non-adaptive one. These factors lead to a more involved technical analysis, although the algorithmic idea is similar.

The difference as compared to the Successive Elimination [Even-Dar et al., 2002] strategy is the incremental nature of Hoeffding Rejection. Successive Elimination samples all arms in a round-robin manner at the beginning and increasingly decreases the set of arms it considers to be potentially optimal until only the optimal arms remains "active". In contrast, Hoeffding Rejection (and its extensions in Chapter 4) focus on taking decisions for one arm at a time, which leads to a different analysis (although the key idea remains unchanged). While the difference is not essential in the Stochastic Dilemma setting, this algorithmic idea will become important in future chapters.

Algorithm 3: Hoeffding Rejection (δ):

- For every $t = 1, 2, \ldots$:
  1. Sample $x_t \sim \mathcal{D}$, update $\hat{x}_t = \frac{\sum_{i=1}^{t} x_i}{t}$ and set $\alpha_t = \sqrt{\frac{2 \log t + \log \frac{1}{\delta} + \log 4}{2t}}$.
  2. Build a two-sided confidence interval $(\hat{x}_t - \alpha_t, \hat{x}_t + \alpha_t)$.
  3. Stop if either $\hat{x}_t + \alpha_t < 0.5$ or $\hat{x}_t - \alpha_t > 0.5$ and recommend the distribution consistent with the value of $\hat{x}$ or continue otherwise.

We are now ready to formally state the properties of **Hoeffding Rejection**:

Theorem 2.4.1. *For a stochastic dilemma with an unknown parameter p, Hoeffding*

Rejection will return the correct answer with probability $\geq 1-\delta$ *and with an expected sample complexity* $m(p,\delta) = O(\frac{1}{(1-2p)^2}(\log\frac{1}{\delta} + \log\frac{1}{1-2p}))$.

Before proving the theorem, one important observation to be made is that there is an extra dependency on p in the log term as compared to the setting when the parameter p is known. As we conjecture in Section 2.4.4, we think an extra dependency of a similar type is necessary (although HR has a suboptimal dependency on p), which would prove that the problem is intrinsically more difficult when p is unknown.

We will prove the theorem in a sequence of three lemmas. The first lemma deals with the correctness of the algorithm:

Lemma 2.4.2. *Hoeffding Rejection is correct with probability at least* $1-\delta$.

Proof. Let q be the parameter for the unknown distribution $\mathcal{D}$. Then, for a fixed time step, the error probability based on the Hoeffding Inequality A.1.1 is:

$$P(\hat{x}_t \notin (q-\alpha_t, q+\alpha_t)) \leq 2e^{-2\alpha_t^2 t} \tag{2.32}$$

$$\leq \frac{\delta}{2t^2} \text{ (by substituting the value of } \alpha_t) \tag{2.33}$$

Since the confidence interval monotonically shrinks after every time step, and since $p \neq 0.5$, there exists a finite time t_f after which 0.5 will drop outside the interval. The algorithm *can* be incorrect if at any step before and including t_f, $\hat{x}$ drops outside the confidence interval. Thus, a (loose) upper bound on the probability that the algorithm is incorrect is the probability that the empirical average will not be in the correct confidence interval over the infinite horizon. So $P(error) \leq \sum_{t=1}^{\infty} \frac{\delta}{2t^2} < \delta$ as desired. □

We will use this trick of using an infinite series to bound the total error probability of an algorithm throughout the dissertation.

The second lemma describes the sample complexity of HR in the scenario in which the algorithm never makes an error:

Lemma 2.4.3. *Conditioned on $\hat{x}_t \in (q - \alpha_t, q + \alpha_t), \forall t \geq 1$, Hoeffding Rejection will stop in at most $m(p, \delta) = \frac{8}{(1-2p)^2}(2 \log \frac{1}{1-2p} + \log \frac{8}{\delta})$ steps for any $\delta \leq \frac{1}{4}$.*

Proof. Let's assume wlog that $q = p$ (the other case is identical). We know that as soon as $\alpha_t < \frac{1-2p}{2}$, the confidence interval will not contain 0.5 anymore (since we assumed that $\hat{x}_t \in (p - \alpha_t, p + \alpha_t), \forall t \geq 1$) and will stop.

Now, with some simple manipulation of α_t's formula, we get that $\alpha_t \leq \sqrt{\frac{\log \frac{t}{\delta}}{t}}$. The inequality holds for any $\delta \leq \frac{1}{4}$. If $\delta > \frac{1}{4}$, we will solve the problem for $\delta = \delta_0 = \frac{1}{4}$ and the sample complexity will be guaranteed to be an upper bound for any larger failure probability $\delta > \delta_0$. Let's choose a time $t_f = x \log \frac{x}{\delta}$, for some x to be defined. Then:

$$\alpha_{t_f}^2 \leq \frac{\log \frac{x \log \frac{x}{\delta}}{\delta}}{x \log \frac{x}{\delta}} \leq \frac{1}{x} + \frac{\log \log \frac{x}{\delta}}{x \log \frac{x}{\delta}} < \frac{2}{x} \tag{2.34}$$

where the last inequality holds for any $\frac{x}{\delta} > 1$ (which trivially holds given the choice of x below).

Let's pick $x = \frac{8}{(1-2p)^2}$ and thus:

$$\alpha_{t_f} < \frac{1-2p}{2}, \forall t \geq t_f = \frac{8}{(1-2p)^2}(2 \log \frac{1}{1-2p} + \log \frac{8}{\delta}) \tag{2.35}$$

and the algorithm will stop as required after m samples. □

In the third lemma, we will show that the bound above is tight asymptotically even when HR "fails" and thus cover the general case for sample complexity. One difference with the proof from Theorem 2.4.3 is we need to consider that the sample complexity of HR is a random variable and must show that its expected value is bounded.

Lemma 2.4.4. *The expected number of samples HR takes is at most* $m(p,\delta) = O(\frac{1}{(1-2p)^2}\log\frac{1}{\delta(1-2p)})$.

Proof. Same as before, let's assume wlog that $q = p$ (the other case is identical). In the proof of Lemma 2.4.3 for the formula of t_f let's now take $x = \frac{32}{(1-2p)^2}$ which leads to $\alpha_t < \frac{1-2p}{4}$ for $t \geq t_f = \frac{C}{(1-2p)^2}\log\frac{1}{(1-2p)\delta}$ (where we replace the constants by a fixed constant C to make the proof more readable).

If $m(p,\delta)$ is the expected number of samples of HR then:

$$m(p,\delta) = \sum_{t=1}^{\infty} tP(m=t) \tag{2.36}$$

$$= \sum_{t=1}^{t_f} tP(m=t) + \sum_{t>t_f} tP(m=t) \tag{2.37}$$

$$\leq \frac{C}{(1-2p)^2}\log\frac{1}{(1-2p)\delta}P(t \leq t_f) + \sum_{t>t_f} tP(m=t) \tag{2.38}$$

For any $t > t_f$, $P(m=t)$ is the probability that the algorithm stops at step t. If the algorithm stops at step t it means the confidence interval around $\hat{x}_{t-1}$ contains 0.5 (otherwise the algorithm would have stopped at step $t-1$). So,

$$P(m=t) \leq P(HR \text{ doesn't stop at } t-1) \tag{2.39}$$

But, $\alpha_{t-1} < \frac{1-2p}{4}$, since $t-1 \geq t_f$, which means that $\hat{x}_{t-1} \notin (p - \frac{1-2p}{4}, p + \frac{1-2p}{4})$. The event "the confidence interval around $\hat{x}_{t-1}$ contains 0.5" is contained in the event that "$\hat{x}_{t-1} \notin (p - \frac{1-2p}{4}, p + \frac{1-2p}{4})$" and thus

$$P(HR \text{ doesn't stop at } t-1) \leq P(\hat{x}_{t-1} \notin (p - \frac{1-2p}{4}, p + \frac{1-2p}{4})) \tag{2.40}$$

Putting Equations 2.39 and 2.40 together and applying the Hoeffding inequality A.1.1 in the right side of Equation 2.40 we get:

$$P(m=t) \leq 2e^{-\frac{(1-2p)^2 t}{8}}, \forall t > t_f \tag{2.41}$$

Now, from Equations 2.38 and 2.41:

$$m(p,\delta) \leq \frac{C}{(1-2p)^2}\log\frac{1}{(1-2p)\delta}P(t \leq t_f) + \frac{1}{2}\sum_{t>t_f}\frac{t}{e^{\frac{(1-2p)^2}{8}t}} \tag{2.42}$$

and since the infinite series on the right side of the inequality converges to a constant and $P(t \leq t_f) \leq 1$, by absorbing the constants into C we get as desired that:

$$m(p,\delta) \leq \frac{C}{(1-2p)^2}\log\frac{1}{(1-2p)\delta} \tag{2.43}$$

□

The proof of Theorem 2.4.1 is simply the combination of Lemmas 2.4.2 and 2.4.4.

2.4.2 Lazy Hoeffding Rejection

Lazy Hoeffding Rejection is a variant of Hoeffding Rejection that is more efficient in terms of its expected sample complexity. By exponentially decreasing the number of "decisions" it takes, Lazy Hoeffding Rejection improves the asymptotic dependency of the sample complexity on the parameter p while maintaining the desired failure probability guarantee. As we will show in Section 2.4.4, Lazy Hoeffding Rejection is in fact asymptotically optimal. The same idea for improving Successive Elimination (the algorithm corresponding to Hoeffding Rejection from the PAC Bandit setting) is mentioned by Even-Dar et al. [2002] (Remark 1 in Section 3.1).

The key idea of the algorithm is to apply a doubling trick to the number of steps between checking whether the dynamic confidence interval built using the Hoeffding inequality still contains 0.5. This process allows us to apply the union bound for the failure probability on an exponentially smaller number

of steps as compared to the Hoeffding Rejection algorithm. In other words, instead of checking at every step that the confidence interval contains 0.5, we will only do so at steps that are a power of 2. This scheme gives us the opportunity to define tighter confidence intervals α_t as compared to Hoeffding Rejection and we will prove that this construction will decrease the expected sample complexity for solving the Stochastic Dilemma problem.

Algorithm 4: Lazy Hoeffding Rejection (δ):

- Take one initial sample from $\mathcal{D}$ and then, for every $t = 1, 2, \ldots$
  1. Take 2^{t-1} samples from $\mathcal{D}$, let $T = 2^t$ be the total number of samples taken so far and update $\hat{x}_T = \frac{\sum_{i=1}^{T} x_i}{T}$.
  2. Set $\alpha_T = \sqrt{\frac{2 \log\log T + \log\frac{1}{\delta} + \log 9}{2T}}$.
  3. Build a two-sided confidence interval $(\hat{x}_T - \alpha_T, \hat{x}_T + \alpha_T)$.
  4. Stop if either $\hat{x}_T + \alpha_T < 0.5$ or $\hat{x}_T - \alpha_T > 0.5$ and recommend the distribution consistent with the value of $\hat{x}_T$ or continue otherwise.

Note how, as compared to Hoeffding Rejection, the algorithm only makes decisions about whether to continue or stop at Steps $1, 2, 4, 8, \ldots$.

The theorem stating the properties of Lazy Hoeffding Rejection is:

Theorem 2.4.5. *For a Stochastic Dilemma with an unknown parameter p, Lazy Hoeffding Rejection will return the correct answer with probability $\geq 1 - \delta$ and with an expected sample complexity $m(p, \delta) = O(\frac{1}{(1-2p)^2}(\log\frac{1}{\delta} + \log\log\frac{1}{1-2p}))$.*

The proof follows closely the proof the Theorem 2.4.1, so we will sketch the steps that are similar and focus on the differences.

Proof. **Correctness.** In a similar manner to Lemma 2.4.2, we can compute the

failure probability for the decision step T as:

$$P(\hat{x}_T \notin (q-\alpha_T, q+\alpha_T)) \leq 2e^{-2\alpha_T^2 T} \tag{2.44}$$

$$\leq \frac{2\delta}{9\log^2 T} \text{ (by substituting the value of } \alpha_T) \tag{2.45}$$

$$\leq \frac{\delta}{2t^2} \text{ (since } T = 2^t) \tag{2.46}$$

and thus the total failure probability is bounded: $P(error) \leq \sum_{t=1}^{\infty} \frac{\delta}{2t^2} < \delta$. **Sample Complexity**. In the first part, we will assume as in the proof of Lemma 2.4.3 that $\hat{x}_T \in (q-\alpha_T, q+\alpha_T), \forall T = 1, 2, 4, \ldots$ and wlog that $q = p$. We can get that $\alpha_T \leq \sqrt{\frac{\log \frac{\log T}{\delta}}{T}}$. Choose a time $T_f = x \log \frac{\log x}{\delta}$. Then:

$$\alpha_{T_f} \leq \frac{\log \frac{\log(x \log \frac{\log x}{\delta})}{\delta}}{x \log \frac{\log x}{\delta}} = \frac{\log(\frac{\log x}{\delta} + \frac{\log \log \frac{\log x}{\delta}}{\delta})}{x \log \frac{\log x}{\delta}} \leq \frac{\log \frac{\log x}{\delta} + \log \frac{\log \log \frac{\log x}{\delta}}{\delta}}{x \log \frac{\log x}{\delta}} \tag{2.47}$$

where the last inequality follows from $\log(a+b) \leq \log a + \log b$ for any $a, b \geq 2$ and the corresponding terms are indeed ≥ 2 for any non-trivial SD(p, δ). Then, as in Lemma 2.4.3, $\alpha_{T_f} \leq \frac{2}{x}$ and if we choose $x = \frac{32}{(1-2p)^2}$, we get that $\alpha_{T_f} < \frac{1-2p}{4}, \forall T \geq T_f = \frac{C}{(1-2p)^2}(\log \log \frac{1}{1-2p} + \log \frac{1}{\delta})$ for some positive constant C. We remark that we actually have to choose T_f to be the closest higher number that is a power of 2 (so that we can have a decision at that time), which increases the sample complexity by a factor of at most 2 (which is absorbed in the constant C in the bound above).

The rest of the proof (corresponding to Lemma 2.4.4) is very similar to what was presented earlier with the only difference being that we need to do the summations in the expected value of m only for the steps that are powers of 2. □

2.4.3 Confidence Sequences

The algorithm we will present in this section (we are labeling it *Confidence Sequences*) was introduced in the statistics literature [Darling and Robbins, 1967b], [Darling and Robbins, 1967a] in the context of studying consequences of the law of iterated logarithm[3]. While the algorithm we will introduce is suboptimal in its dependency on the failure probability, it provides a complementary algorithmic perspective to Lazy Hoeffding Rejection. Moreover, it proved to be very competitive empirically in a variety of experiments so we believe it is worth discussing it in detail.

The algorithm's behavior has two epochs. In the initial epoch, Confidence Sequences takes a number of samples from $\mathcal{D}$ that is only a function of the failure probability δ. During this epoch, no decisions are made with respect to stopping. In the second epoch, similar to Hoeffding Rejection and Lazy Hoeffding Rejection, a confidence interval is built around the empirical average and a decision to stop is taken as soon as 0.5 drops outside the confidence interval. The key component of the second epoch is its use of a tighter interval as compared to the algorithms we have previously seen in this section.

An intuitive (but inaccurate) perspective is that Confidence Sequences compresses all the steps from Lazy Hoeffding Rejection when Lazy Hoeffding Rejection doesn't take a decision in an initial phase, with the advantage of having a tighter confidence interval for decisions at every step after this initial epoch ends and with the disadvantage of having a potentially very long initial epoch.

The analysis and the key ideas of Confidence Sequences were introduced byDarling and Robbins [1967b]. The main goal of this section is to translate

[3] Which states asymptotic guarantees with respect to the magnitude of fluctuations of a random walk.

their result into a solution for a Stochastic Dilemma. The proof is mostly technical, with the key idea from Darling and Robbins [1967b] remaining unchanged. We note that while the original proof has a different starting point, one of the key ideas is to use a trick similar to the doubling trick of Lazy Hoeffding Rejection (see Equation 9 from page 1189 in Darling and Robbins [1967b]).

Let's define the constants: $C_1 = e^{\frac{\log 2}{2}}, C_2 = \frac{14 \log 2}{40}, C_3 = 2.12, C_4 = 1.236$.

Algorithm 5: Confidence Sequences (δ):

- **Epoch 1** Take $k = C_1 e^{\frac{C_2}{\delta}}$ samples from $\mathcal{D}$.
- **Epoch 2** For $t = k, k+1, k+2, \ldots$:
  1. Sample $x_t \sim \mathcal{D}$, update $\hat{x}_t = \frac{\sum_{i=1}^{t} x_t}{t}$ and set $\alpha_t = C_3 \sqrt{\frac{\log \log t + C_4}{t}}$.
  2. Build a two-sided confidence interval $(\hat{x}_t - \alpha_t, \hat{x}_t + \alpha_t)$.
  3. Stop if either $\hat{x}_t + \alpha_t < 0.5$ or $\hat{x}_t - \alpha_t > 0.5$ and recommend the distribution consistent with the value of $\hat{x}$ or continue otherwise.

The actual exponential dependency on $\frac{1}{\delta}$ from the first epoch is mild (due to the constants) as long as δ is not very small ($k = 16$ for $\delta = 0.1$ for example) but it quickly becomes impractically large for small δ's ($k > 10^{10}$ for $\delta = 0.01$).

The second epoch is essentially identical to the Hoeffding Rejection algorithm, except for the $\log \log t$ dependency in the confidence interval formula.

We will begin the analysis by formally stating the original result from Darling and Robbins [1967a] (part $2(e)$ "Bounds on P_m"), which is an example of the general proof from Darling and Robbins [1967b] (in particular it is an application of part 3 "Other choices"). We note that in their papers $\log_2 n = \log \log n$.

Theorem 2.4.6 (Darling and Robbins [1967a]). *Let X be a random variable with mean 0 and moment generating function $\phi(s) \leq e^{\frac{s^2}{2}}$ (for $s > 0$) and let $\alpha_t = 2.12\sqrt{\frac{\log\log t + 1.236}{t}}$ be a function defined on natural numbers $t > 1$. If $x_i, i \geq 1$ are iid random variables with the same distribution as X and $\hat{x}_t = \frac{\sum_{i=1}^{t} x_i}{t}$ then $P_k = P(|\hat{x}_t| \geq \alpha_t \text{ for some } t \geq k) \leq \frac{14}{40}\frac{1}{\frac{\log k}{\log 2} - \frac{1}{2}}$.*

The theorem sets an upper bound on the probability (P_k) that the empirical average of a random variable (with certain properties) will *ever* be outside the confidence interval (α_t) after an initial number of steps k (P_k dropping logarithmically with k).

We note that the only difference with their formulation is a doubling of the probability P_k (which is obtained by a simple union bound) due to our goal of having a two-sided confidence interval around $\hat{x}_t$.

Let's now state the main theorem of this section that characterizes the performance of Confidence Sequences:

Theorem 2.4.7. *For a $SD(p, \delta)$, Confidence Sequences will return the correct answer with probability $\geq 1 - \delta$ and with an expected sample complexity $m(p, \delta) = O(e^{\frac{1}{\delta}} + \frac{1}{(1-2p)^2}\log\log\frac{1}{1-2p})$.*

Proof. Let's assume as in the other proofs in this section that the parameter of $\mathcal{D}$ is $q = p$.

Correctness. The goal in the first part of the proof is to connect the Stochastic Dilemma problem to Theorem 2.4.6. If X is Bernoulli(p), let $Y = \frac{X-p}{1-p}$ be an affine transformation of X. Then, $\mathbb{E}[Y] = 0$ and the moment generating function of Y is $\phi(s) = \mathbb{E}[e^{sY}] = pe^s + (1-p)e^{-\frac{sp}{1-p}}$. It follows that $\phi(0) = 1$ and it can be shown that $g(s) = \frac{\phi(s)}{e^{\frac{s^2}{2}}} \leq 1$ by showing that $g(s)$ is a decreasing function when $p < 0.5$ (as we constrain p in the definition of the Stochastic

Dilemma problem). To show this last fact we can compute the first derivative of $g(s)$ and show that it is negative on $\{s > 0\}$.

The above facts show that Y fulfills the constraints of Theorem 2.4.6 and we can thus apply the result that $P(|\hat{y}_t| \geq \alpha_t \text{ for some } t \geq k = C_1 e^{\frac{C_2}{\delta}}) \leq \delta$ (where we replace k with the value from the algorithm description and do the algebra to get δ on the right side of the inequality).

By replacing $y_i = \frac{x_i - p}{1-p}$ in the expression of $\hat{y}_t$, we get $P(|\hat{x}_t - p| \geq (1-p)\alpha_t \text{ for some } t \geq k = C_1 e^{\frac{C_2}{\delta}}) \leq \delta$. And since by increasing the size of the sequence of confidence intervals, we decrease the probability that the empirical average will exit them at some point, and $(1-p)\alpha_t < \alpha_t$, we get that $P(|\hat{x}_t - p| \geq \alpha_t \text{ for some } t \geq k = C_1 e^{\frac{C_2}{\delta}}) \leq P(|\hat{x}_t - p| \geq (1-p)\alpha_t \text{ for some } t \geq k = C_1 e^{\frac{C_2}{\delta}}) \leq \delta$.

Because with probability at least δ the empirical average will be inside the confidence interval around the true expected value of $\mathcal{D}$ and since α_t is strictly decreasing, we get that, in finite time, the algorithm will stop and recommend the correct answer with probability at least $1 - \delta$.

Sample Complexity. The second part of the proof is similar to the proofs of Hoeffding Rejection and Lazy Hoeffding Rejection. The first step is to compute the smallest time t_f such that $\alpha_{t_f} < \frac{1-2p}{4}$ for $t > t_f$ under the assumption that the empirical average always stays inside the confidence interval. In a very similar manner to the proof of Lazy Hoeffding Rejection, we get a bound of $t_f = O(e^{\frac{C_2}{\delta}} + \frac{1}{(1-2p)^2} \log\log \frac{1}{1-2p})$. The second and final step applies the same arguments as in lemma 2.4.4 to bound the expected sample complexity to the same asymptotic quantity (t_f) when failure is allowed. □

2.4.4 Lower Bounds

The Stochastic Dilemma problem with p unknown is clearly at least as hard as when the parameter p is known. It is thus obvious that the lower bound result from theorem 2.3.5 holds for this version of the problem as well:

Corollary 2.4.8. *Any algorithm that correctly solves a $SD(p, \delta)$ with unknown parameter p must take at least $m(p, \delta) = \Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$ samples from $\mathcal{D}$ on expectation.*

We think that the lower bound above is actually loose and a tighter dependency on p is possible. To be exact, we conjecture that the right dependency on p scales with $\Omega(\frac{1}{(1-2p)^2} \log \log \frac{1}{1-2p})$ (and coincides with the asymptotic dependency on p in the upper bounds of Lazy Hoeffding Rejection and Confidence Sequences).

The intuition behind the conjecture is based on the law of iterated logarithm, which establishes the tightest possible dynamic confidence interval (which scales with $\Theta(\sqrt{t \log \log t})$ with t being the number of samples) for the magnitude of fluctuations of an unbiased random walk. While we operate with Bernoulli(p) distributions with $p \neq 0.5$ (which correspond to biased random walks), we can transform these variables into unbiased random walks (using linear transformations as in the correctness proof of theorem 2.4.7) and the same type of lower bound should apply. Robbins [1970] for example discusses that the confidence interval described above is as tight as possible (up to constants) while still allowing a positive probability for the random walk to stay inside the dynamic confidence interval at each step over the infinite horizon (see Example 3, equation 14 at page 1400).

2.5 Summary and Discussion

2.5.1 Known Parameters

The three algorithms we proposed in Section 2.3 (Majority Vote, Early Majority Vote and Beat-By-K) are asymptotically optimal with an upper bound on the sample complexity for solving a SD(p, δ) problem identical to the lower bound $\Omega(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$.

One difference between (Early) Majority Vote and Beat-By-K is that while the upper bound for Beat-By-K is relatively tight up to constants, the upper bound for Majority Vote is loose, mainly due to the sufficient but not necessary conditions imposed by applying the Hoeffding inequality. This makes a formal comparison between the two classes of strategies relatively difficult (as the length of Section 2.3.4 demonstrates).

While Beat-By-K is shown to dominate both Majority Vote and Early Majority Vote, the proofs hold only for relatively large values of $\frac{1}{1-2p}$ and $\frac{1}{\delta}$. One reason is that we prove the dominance of Beat-By-K by comparing it directly with a loose lower bound on the performance of Majority Vote. Tightening this lower bound is a topic of active research. The number of papers proving increasingly tight lower bounds by increasing the constants (see the references in the introduction of this chapter) suggest that the problem is relatively difficult.

Numerical simulations suggest that the gap between Beat-By-K and Majority Vote is more significant than what we were able to prove and moreover holds uniformly over any p and δ values. It is for this reason that Beat-By-K can actually be a relevant practical alternative to Majority Vote-type approaches in various problems where reducing label noise is a challenge (like categorical data annotation or stochastic optimization). We will empirically compare the

two types of strategies in realistic problems in Chapter 3).

Finally, we would like to note that while the Beat-By-K algorithm is defined to take only a fixed parameter $k \in \mathbb{N}^*$, it is possible (and desirable in practice) to combine two successive k values in a randomized way to obtain a better granularity with respect to achievable failure probabilities. In more detail, since the k parameter is a natural number, for any desired failure probability that is higher than the failure probability achievable by k, we need to round the parameter for the Beat-By-K strategy to $k+1$ with the effect of increasing sample complexity. But it is actually possible to obtain a lower expected sample complexity by a convex combination of k and $k+1$ with a coin flip deciding (before the first sample is taken) which of the two should be used for solving a particular Stochastic Dilemma.

2.5.2 Unknown Parameters

All of the algorithms we discussed in Section 2.4 have a similar structure: they maintain a dynamic confidence interval around the empirical average. As the number of steps increases, the confidence interval decreases at each step at a rate that makes sure that the true average will be contained in the interval with high probability. The algorithms stop as soon as 0.5 drops outside the confidence interval which is interpreted as a "proof" that the true bias lies on the same side compared to 0.5 as the empirical average.

What differs between the algorithms is how tight these confidence intervals are and when are the algorithms checking for the stopping conditions. As we already saw these choices impact the theoretical performance and, as we will describe in the future chapters, they impact the empirical performance of the algorithms.

	Expected Sample Complexity	Theorem
Hoeffding Rejection	$O(\frac{1}{(1-2p)^2}(\log\frac{1}{\delta}+\log\frac{1}{1-2p}))$	2.4.1
Lazy Hoeffding Rejection	$O(\frac{1}{(1-2p)^2}(\log\frac{1}{\delta}+\log\log\frac{1}{1-2p}))$	2.4.5
Confidence Sequences	$O(e^{\frac{1}{\delta}}+\frac{1}{(1-2p)^2}\log\frac{1}{\delta})$	2.4.7
Lower Bound	$\Omega(\frac{1}{(1-2p)^2}\log\frac{1}{\delta})$	2.4.8

Table 2.1: Summary of the results for a SD(p,δ) for p unknown

In table 2.1 we summarize the upper and lower bounds discussed so far for the version of the Stochastic Dilemma problem where p is unknown. The important thing to remark is that Lazy Hoeffding Rejection asymptotically dominates the other two algorithms in the following sense: (1) there exists a class of SD(p,δ) problems for which Lazy Hoeffding Rejection and Hoeffding Rejection dominate Confidence Sequences and also (2) there exist another class of domains for which Lazy Hoeffding Rejection and Confidence Sequences dominate Hoeffding Rejection.

As an example of a class of the first type let's consider the case where $\frac{1}{\delta} = \Theta(\frac{1}{1-2p})$. Then the upper bounds for Lazy Hoeffding Rejection and Hoeffding Rejection are $O(\frac{1}{(1-2p)^2}\log\frac{1}{1-2p})$, whereas a lower bound on the performance of Confidence Sequences is $\Omega(e^{\frac{1}{1-2p}})$ which proves that Confidence Sequences has worse theoretical guarantees as compared to Lazy Hoeffding Rejection. The lower bound for the performance of Confidence Sequences follows from its definition of Epoch 1 in Section 2.4.3.

As for the example of a class of the second type let's consider the case where $\frac{1}{\delta} = \Theta(\log\frac{1}{1-2p})$. Then the upper bound for Lazy Hoeffding Rejection and Confidence Sequences is $O(\frac{1}{(1-2p)^2}\log\log\frac{1}{1-2p})$ whereas a lower bound on the performance of Hoeffding Rejection is $\Omega(\frac{1}{(1-2p)^2}\log\frac{1}{1-2p})$ which proves that

Lazy Hoeffding Rejection is better than Hoeffding Rejection. The lower bound on the performance of Hoeffding Rejection follows from lemma 2.4.3 which forces the expected sample complexity to be at least $(1-\delta)\frac{8}{(1-2p)^2}(2\log\frac{1}{1-2p} + \log\frac{8}{\delta}) = \Omega(\frac{1}{(1-2p)^2}\log\frac{1}{1-2p})$.

On the practical side, a key observation is that it is usually the case that the empirical failure probability is much smaller than the parameter of the $\mathrm{SD}(p,\delta)$ problem. Part of the reason is that the parameters of Hoeffding Rejection and Lazy Hoeffding Rejection are conservatively set as a function of concentration bounds, which are loose.

In the case of Confidence Sequences the main looseness comes because of the potentially very long initial epoch which appears to be the consequence of a loose analysis. In numerical simulations, Confidence Sequences tends to lead to much smaller failure probabilities than what the theory results predict for much shorter initial epochs.

Chapter 3
Label Identification

3.1 Introduction

The introduction of crowdsourcing platforms such as Amazon Mechanical Turk (mturk.com) has made it possible to harness cheap, albeit sometimes unreliable, human labor in a range of data annotation tasks. The most common approach for improving the quality of annotations, *consensus labeling* or *majority voting*, seeks multiple annotations, each from a different worker, and use their majority vote to assign a final output Smyth et al. [1994]; Sheng et al. [2008]. However, each annotation costs money, and this chapter targets the question of whether we can be smarter in allocating annotation tasks to workers so as to boost annotation accuracy in a cost-sensitive fashion.

We focus on what is known in the literature as categorical data annotation or label identification that concerns tasks in which the data annotation is itself the task of interest — for example, tasks that assess whether Web pages contain objectionable content. Unlike other work that models individual worker ability (Carpenter [2008] for example), we consider the common case when the vast majority of workers perform a single task and are never seen again. In such cases there is little need to model individual workers since the model will never have to be used in the future. We therefore model the entire crowdsourcing resource as one single, noisy oracle that assigns one of a fixed set of labels to

items in a dataset, but where each time the label may be corrupted by some random noise process.

For the purpose of this work we assume that each item is assigned a binary label, 0 or 1. In contrast to the static approach of consensus labeling, which always seeks m labels for an item, we present dynamic approaches based on the algorithms from chapter 2 that iteratively asks for labels for an item until some stopping criterion is met.

Finally, we note that our main interest in this chapter is to illustrate how the algorithmic ideas used to solve a Stochastic Dilemma can be used to solve realistic problems. So, in some sense, we are reducing the label identification problem to the Stochastic Dilemma problem without explicitly searching for optimal strategies in this new setting. We are thus changing the focus from the type of theoretical analysis from chapter 2 to one that is focused on applying the algorithms to solve actual problems.

3.2 Model

We first formally introduce the model and then discuss the meaning of the parameters and the connections to stochastic dilemmas.

We define an object o_r as a Bernoulli(r) distribution (for some fixed parameter $r \in [0, 1]$). The label of the object is defined to be 0 if $r < 0.5$ and 1 otherwise. Let's assume we are given the set $D = \{\text{Bernoulli}(r), \forall r \in [0, 1]\}$ (the set of all possible Bernoulli distributions) and access to an arbitrary distribution $\mathcal{P}$ over D (in other words, $\mathcal{P}$ is an arbitrary distribution with support $[0, 1]$).

The objects to be labeled are sampled according to $\mathcal{P}$. Every time a label for a fixed object is requested, the environment will return a sample from its

associated Bernoulli distribution. One way to interpret this process is that the "true" label of an object is transmitted via a noisy channel.

We now define the problem we study in this chapter.

Label Identification (LI$(\mathcal{P}, \epsilon, \delta)$). Given sampling access to the object generator $\mathcal{P}$, an accuracy parameter $\epsilon \in (0, 0.5)$, and a desired failure probability $\delta \in (0, 0.5)$, we are interested in developing algorithms that find the label of a sampled object o_r with failure probability at most δ if $r \leq 0.5 - \frac{\epsilon}{2}$ or $r \geq 0.5 + \frac{\epsilon}{2}$, while minimizing the expected number of sample $m(\mathcal{P}, \epsilon, \delta)$ from Bernoulli(r).

It is important to note that both the failure probability and the sample complexity are random variables that depend on the overall distribution $\mathcal{P}$. In other words we allow ourselves the flexibility of having variable failure probability as long as the overall failure with respect to both sampling o_r from $\mathcal{P}$ and sampling from Bernoulli(r) is at most δ.

In what follows we discuss the parameters, the motivation for the problem and the connection to stochastic dilemmas.

3.2.1 Parameter ϵ

There are several reasons for introducing the new parameter ϵ. If we return to the simple Stochastic Dilemma problem for a moment and ignore ϵ, the lower bound from theorem 2.3.5 still applies and thus one needs at least $\Omega(\frac{1}{(1-2r)^2} \log \frac{1}{\delta})$ samples to separate the two possible answers with high probability. In particular, when r is arbitrarily close to 0.5 the number of samples needed to solve the problem becomes arbitrarily large (without this fact being known a priori). Naturally, this situation is usually undesirable in practice. And this is one reason to have an approximation parameter like ϵ which establishes what is the threshold for which the difference between the two labels 0 and 1 is still

relevant.

Another way to interpret ϵ is as a proxy for the maximum number of samples one is willing to take to differentiate between the two labels. Of course, one could just stop the sampling process after some maximum budget of samples is spent for a fixed object. Given the type of strategies we design though, it will become clear that we can deterministically control for the budget by setting ϵ with the extra advantage of having clear guarantees with respect to what is the impact of a particular fixed budget.

There are other, more technical reasons, for having a parameter like ϵ that we will return to in section 3.4.2.

3.2.2 Distribution $\mathcal{P}$

The role of the distribution $\mathcal{P}$ is to simulate realistic processes that generate labels for objects like images or text. In particular, it is motivated by what is commonly observed when using crowdsourcing resources such as Amazon Mechanical Turk to label data.

Given the large number of workers available on such crowdsourcing platforms, each label for a fixed object can be interpreted as being generated iid from some joint distribution over worker abilities (e.g. various expertize for example) and object characteristics (e.g. various intrinsic difficulties in labeling). A fixed worker distribution and a fixed object will thus lead to an overall fixed probability p with which an object will be labeled 0 or 1.

And, since we target settings with heterogeneous, large populations of workers and sets of possible objects, we can interpret the process of obtaining an object as sampling from some arbitrary distribution $\mathcal{P}$ over r values.

Another way to interpret $\mathcal{P}$ is as a generalization of a hierarchical Bayesian

model (as the ones introduced by Carpenter [2008] in the context of categorical data annotation), where the probability distribution that generates the label in a Bayesian model is a particular instantiation of the arbitrary distribution $\mathcal{P}$.

3.2.3 Discussion

A connection to Stochastic Dilemma. The problem described above is closely related to the Stochastic Dilemma problem. A Stochastic Dilemma can be seen as a special case of Label Identification. To see this, we can consider $\mathcal{P}$ to be uniform over a support containing two values p and $1 - p$, take $\epsilon = 0$ and add the additional constraint that $p \neq 0.5$. Conversely, the Label Identification problems can be seen as generalization of Stochastic Dilemmas, where the goal of deciding the position of a parameter p with respect to 0.5 remains, while the process of generating p is more complex.

Noise process. One assumption in the way we defined the problem is that the label is determined strictly by the position of r with respect to 0.5. This corresponds to a type of noisy channel that only corrupts the true label with probability < 0.5 (and we ignore the case of $r = 0.5$ here). In such a setting, it is thus theoretically possible to always recuperate the true label of an object. However, in reality, it is not always possible to find the true label. Due to features of the object or to an adversarial labeling process, it is possible for the true label to be completely flipped. Unfortunately, lacking other information, this type of error is not recoverable irrespective of the number of labels taken for an object. We have thus chosen to define the *truth* strictly as a function of r (a parameter that we can estimate within arbitrary accuracy given enough samples).

Failure probability. In the definition of the problem, we require a failure

probability δ that is allowed to vary as a function of the sampled object o. This might seem strange as it means that for some objects, the failure probability can be close to 0.5 which seems to defeat the purpose of labeling an object (since a random decision can be taken in such scenarios with the same outcome). We could have added an extra constraint where δ is fixed for any sampled object (and in fact some algorithms do have that property - see section 3.4.2). However, we allow ourselves the aforementioned flexibility because the main use-case of the algorithms introduced in this section is to label datasets of N objects (for some possibly large N) where the goal is to have only a small percentage of the dataset (δN) labeled incorrectly. In such scenarios (with large values for N), due to the concentration of measure phenomenon, both the actual failure probability and the sample complexity will concentrate around their expected values.

We also note that the failure probability is computed only with respect to objects in the ranges $(0, \frac{1-\epsilon}{2})$ and $(\frac{1+\epsilon}{2}, 1)$. This avoids making the problem trivial in situations in which, for example, the distribution $\mathcal{P}$ for example has a probability mass of more than $1 - \delta$ in the range where any answer is acceptable (in $(\frac{1-\epsilon}{2}, \frac{1+\epsilon}{2})$).

Related problems Finally, we note that while our formulation of the Label Identification problem is relatively general, there are two extensions of practical interest that we do not cover in this chapter: (1) class-dependent failure probabilities as opposed to a single, general one and (2) more than two possible labels. We chose a simpler setting to simplify the presentation and we believe the results in this chapter can be extended to cover these extensions.

3.3 Related Work

Classical work in noisy data annotation has focused on estimating error rates of labelers. Dawid and Skene [1979] considered the problem where data labels are obtained from members of a set of noisy human labelers, and learned how to give a weight to each labeler so that their weighted vote more accurately assigned labels to future data. Much of the follow-up work in this area considers such cases where there is value in modeling worker abilities so as to be smart about how to select workers and how to combine their labels to assign labels to items more effectively [Smyth et al., 1994].

With the advent of crowdsourcing platforms in the past few years, renewed interest in the problem led to several papers using probabilistic models to characterize naturally occurring pools of experts [Carpenter, 2008; Whitehill et al., 2009; Raykar et al., 2010; Yan et al., 2010; Dai et al., 2011; Wauthier and Jordan, 2011]. It is well known that besides the flexibility of having a high number of workers available for various tasks, an important problem when interacting with crowds is the variability in expertize and the existence of spammers [Ipeirotis et al., 2010]. In this context, Snow et al. [2008] describes an interesting phenomenon: that using multiple noisy labelers is equivalent to using experts for annotation tasks.

One possible application of solutions for label identification is determined by its use in building classifiers that are robust to noise. As observed by Donmez and Carbonell [2008], the standard way a labeling oracle is modeled in classical machine learning (as a single source of possibly noisy labels) is often unrealistic. In reality, instead of being exposed to a single oracle, multiple oracles of different expertise and with different availabilities are usually interacting with the learner.

Most of the work on crowdsourced labeling for machine learning applications focuses on active learning. As it is described and motivated by Yan et al. [2010] and Kamar et al. [2012], a key assumption are that labelers provide a sufficient number of examples so that information about their abilities can be inferred from their history. A second key premise, assumed by most recent papers (see Donmez et al. [2009] for example) is that workers can be selected explicitly for a particular job from the pool of available labelers.

In contrast to the work described above, we address another common situation where neither of the two assumptions hold. As in Sheng et al. [2008]; Smyth et al. [1994] we assume that the set of workers is so large that most labels are obtained by a worker seen only a few times and moreover we are not able to explicitly select particular labelers. This setting is motivated by resources such as Amazon Mechanical Turk, where in many cases each worker appears and labels just one item. In such cases there is no point in learning about labelers, since most workers will never be seen again.

3.4 Algorithms

In this section we introduce a set of algorithms, analyze their theoretical properties and describe their performance in two regimes of the Label Identification problem:

1. *Interesting Regime*: $\epsilon < |1 - 2r|$ (only one label is "correct").

2. *Trivial Regime*: $\epsilon \geq |1 - 2r|$ (both labels are "correct").

The *Interesting Regime* corresponds to the range of r values for which we must find the correct label for an object with probability $\geq 1 - \delta$, whereas

in the *Trivial Regime* any answer is acceptable with any probability. One key difficulty is that we do not know which regime a particular object belongs to.

3.4.1 Naive Majority Vote

The first algorithm is based on the same idea and techniques as the **Majority Vote** algorithm from section 2.3.1. It assumes that $r = \frac{1-\epsilon}{2}$ and computes the number of samples $m(\epsilon, \delta) = \frac{2}{\epsilon^2} \log \frac{2}{\delta}$ under this assumption. We state the theorem for completeness and only sketch the proof as it is straightforward:

Theorem 3.4.1. *If the Naive Majority Vote algorithm takes* $m = \lceil \frac{2}{\epsilon^2} \log \frac{2}{\delta} \rceil = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ *samples, it will correctly solve a* $LI(\mathcal{P}, \epsilon, \delta)$ *problem with failure probability at most* δ *and accuracy at least* ϵ*.*

Proof. In the *Interesting Regime*, the proof is identical to the proof for Majority Vote and m is a sufficient quantity for the desired failure probability. In the *Trivial Regime*, the failure probability is 0 (since any answer is correct) and the sample complexity remains the same. We note that the failure probability is the same for any object o_r sampled from $\mathcal{P}$. □

The performance of the algorithm is thus identical in both regimes. The algorithm is asymptotically optimal in the *Trivial Regime*, but it is unnecessarily conservative in the *Interesting Regime* (since it doesn't take advantage of the fact that $|1 - 2r| > \epsilon$ to stop earlier).

The reason we present it is that it can be used as a stopping rule for other algorithms because it has two nice characteristics: (1) it guarantees the desired accuracy ϵ and the desired maximum failure probability δ in the *Interesting Regime* and (2) it sets a maximum bound on the number of samples that will prove to be useful.

3.4.2 Stopped Hoeffding Rejection

In this section we apply the Hoeffding Rejection strategy to solve the $\text{LI}(\mathcal{P}, \epsilon, \delta)$ problem. Interestingly, the algorithm works almost out of the box. We only need to carefully treat the case in which $|1 - 2r|$ is small. We also note that the analysis and comments in the rest of the section apply equally well to the other algorithms introduced in section 2.4—we chose Hoeffding Rejection as an illustrative example.

Using Hoeffding Rejection to solve $\text{LI}(\mathcal{P}, \epsilon, \delta)$ is straightforward: for any sampled object o_r, apply Hoeffding Rejection to solve a Stochastic Dilemma problem with unknown parameter $p = \min(r, 1 - r)$ and δ and decide 0 if $r < 0.5$ and 1 otherwise. As in the case of Naive Majority Vote, the failure probability is identical for each sampled object.

To illustrate the importance of the parameter ϵ and the need for a stopping rule like the one introduced in the previous section, we first describe the behavior of Hoeffding Rejection in a setting where we eliminate ϵ (or, alternatively set $\epsilon = 0$).

Let's first consider an extreme example where $\mathcal{P} = \text{Dirac}(0.5)$. In such a setting, any sampled r will be 0.5 and the expected sample complexity of Hoeffding Rejection is ∞ (from theorem 2.4.1). While this example is somewhat artificial, it emphasizes the key problem of vanilla Hoeffding Rejection: the inability to control the sample complexity when $|1 - 2r| \approx 0$.

Let's now consider a more natural case and prove the following corollary to 2.4.1 for $\mathcal{P} = \text{Uniform}(0, 1)$:

Corollary 3.4.2. *The expected sample complexity of Hoeffding Rejection for solving* $LI(Uniform(0, 1), 0, \delta)$ *is* ∞.

Proof. If we denote by m the expected sample complexity when the probability is taken over both sampling an object from $\mathcal{P}$ and the samples Hoeffding Rejection takes until stopping we get:

$$m(\text{Uniform}(0,1),0,\delta) = \mathbb{E}_{r\sim\text{Uniform}(0,1),x_i\sim\text{Bernoulli}(r)}[t] \tag{3.1}$$

$$= \int_0^1 \mathbb{E}_{x_i\sim\text{Bernoulli}(r)}(t|r)f(r)dr \text{ (total expectation)} \tag{3.2}$$

$$\geq \int_0^{\frac{1}{2}} m_{\text{HR}}(r,\delta)dr \tag{3.3}$$

since $f(r) = 1$ for the uniform distribution and $\mathbb{E}_{x_i\sim\text{Bernoulli}(r)}(t|r) = m_{\text{HR}}(r,\delta)$ (the expected sample complexity of Hoeffding Rejection for a SD(r,δ)).

But we know from theorem 2.4.8 that $m_{\text{HR}}(r,\delta) \geq \frac{C}{(1-2r)^2}$ for some positive constant C. Thus $m(\text{Uniform}(0,1),0,\delta) \geq \int_0^{\frac{1}{2}} \frac{C}{(1-2r)^2}dr = \infty$. □

The result above shows that even for a seemingly benign distribution over noise values the expected sample complexity of Hoeffding Rejection is unbounded. The same will hold for any $\mathcal{P}$ with significant probability mass around 0.5.

One solution to this issue is to combine the Hoeffding Rejection strategy with Naive Majority Vote and we call the new strategy **Stopped Hoeffding Rejection**. The resulting algorithm will run Hoeffding Rejection with the failure probability parameter $\frac{\delta}{2}$ and Naive Majority Vote with parameters ϵ and $\frac{\delta}{2}$ and will stop when one of the two algorithms decides to stop. Both strategies will use the same set of samples, and the overall failure probability will be δ by union bound.

In what follows we study the sample complexity of Stopped Hoeffding Rejection for some example $\mathcal{P}$ distributions:

Theorem 3.4.3. *Stopped Hoeffding Rejection will solve an LI$(\mathcal{P},\epsilon,\delta)$ problem in:*

(i) $m = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ *expected number of samples if* $\mathcal{P} = Dirac(0.5)$.

(ii) $m = O(\frac{1}{(1-2r)^2} \log \frac{1}{|1-2r|\delta})$ *expected number of samples if* $\mathcal{P} = Uniform\{r, 1-r\}$ *for some fixed* r *such that* $\epsilon < |1-2r|$.

(iii) $m = O(\frac{1}{\epsilon} \log \frac{1}{\epsilon\delta})$ *expected number of samples if* $\mathcal{P} = Uniform(0,1)$.

Before we prove the theorem, let's make several observations. First, an upper bound on sample complexity is always $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ which is tight (as point (i) of the theorem above shows). The second part of the theorem emphasizes that Stopped Hoeffding Rejection has the same upper bound on sample complexity as Hoeffding Rejection when the Label Identification problem is essentially a Stochastic Dilemma problem and we are in the *Interesting Regime*. This is a key component of how Stopped Hoeffding Rejection dominates Naive Majority Vote, since its sample complexity scales better with the problem parameters in the *Interesting Regime*, whereas Naive Majority Vote is very conservative. Finally, the third part of the theorem shows how we gain a factor of $\frac{1}{\epsilon}$ as compared to the worst case by using the properties of the uniform distribution.

Proof. Let's first fix a sampled object o_r and assume that $r \leq 0.5$ (the case of $r \geq 0.5$ is treated in the same manner). We know that Naive Majority Vote stops in at most $m = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ samples and Hoeffding Rejection in $m = O(\frac{1}{(1-2r)^2} \log \frac{1}{\delta(1-2r)})$ (theorem 2.4.1) expected number of samples.

Then Stopped Hoeffding Rejection will stop in $m(r, \epsilon, \delta) = O(\min(\frac{1}{\epsilon^2} \log \frac{1}{\delta}, \frac{1}{(1-2r)^2} \log \frac{1}{\delta(1-2r)}))$ expected number of samples. If we now

compute the expected number of samples with respect to the entire distribution $\mathcal{P}$ (where we assume $\mathcal{P}$ is absolutely continuous—the discrete case is similar):

$$m(\mathcal{P},\epsilon,\delta) = \int_0^1 m(r,\epsilon,\delta)f(r)dr \tag{3.4}$$

$$= \int_0^{\frac{1-\epsilon}{2}} m(r,\epsilon,\delta)f(r)dr + \int_{\frac{1+\epsilon}{2}}^1 m(r,\epsilon,\delta)f(r)dr + \int_{\frac{1-\epsilon}{2}}^{\frac{1+\epsilon}{2}} m(r,\epsilon,\delta)f(r)dr \tag{3.5}$$

Parts (i) and (ii) of the theorem follow directly from the upper bound of $m(r,\epsilon,\delta)$ and the expression of $m(\mathcal{P},\epsilon,\delta)$.

For part (iii) we know that $f(r) = 1$ and the first two terms are identical (due to the symmetry of the uniform distribution). We upper bound $m(r,\epsilon,\delta)$ in the first term with $\frac{C_1}{(1-2r)^2}\log\frac{1}{\delta(1-2r)}$ and in the second term with $\frac{C_2}{\epsilon^2}\log\frac{1}{\delta}$ (for some positive constants C_1, C_2).

$$m(\mathcal{P},\epsilon,\delta) \leq 2\int_0^{\frac{1-\epsilon}{2}} \frac{C_1}{(1-2r)^2}\log\frac{1}{\delta(1-2r)}dr + \int_{\frac{1-\epsilon}{2}}^{\frac{1+\epsilon}{2}} \frac{C_2}{\epsilon^2}\log\frac{1}{\delta}dr \tag{3.6}$$

$$= 2C_1\frac{\log\frac{1}{\delta(1-2r)} - 1}{2(1-2r)}\Bigg|_0^{\frac{1-\epsilon}{2}} + \frac{C_2}{\epsilon}\log\frac{1}{\delta} \tag{3.7}$$

$$\leq \frac{2C_1}{\epsilon}\log\frac{1}{\delta\epsilon} + \frac{C_2}{\epsilon}\log\frac{1}{\delta} \tag{3.8}$$

We thus get that $m(\mathcal{P},\epsilon,\delta) = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta})$. □

It is easy to get a general formula for the sample complexity of Stopped Hoeffding Rejection (using the proof above), but, lacking more details about $\mathcal{P}$, such a bound is not very informative.

3.4.3 Naive Beat-By-K

Similar to the extension of Majority Vote to Naive Majority Vote, we can extend Beat-By-K to an algorithm we label Naive Beat-By-K. The first step is to assume

that $r = \frac{1-\epsilon}{2}$ and derive the parameter $k(\epsilon, \delta) = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ (see theorem 2.3.3).

The second step is to use the stopping rule of Naive Majority Vote (similarly to Stopped Hoeffding Rejection) to ensure the algorithm does not invest a lot of samples when the sampled object is in the *Trivial Regime*. However, adding a stopping criterion is, interesting, not strictly necessary in this case. The reason is that compared to Hoeffding Rejection, a Beat-By-K strategy will always stop in finite time since even for $r = 0.5$ (corresponding to an unbiased random walk), because a random walk is guaranteed to hit one of the two absorbing barriers in finite time.

Before discussing the main theorem let's introduce another result:

Lemma 3.4.4. *For a fixed object o_r with $\epsilon < 1 - 2r$ (and we assume $r < 0.5$) the expected sample complexity of Naive Beat-By-K with parameter $k(\frac{1-\epsilon}{2}, \delta)$ is $m = O(\frac{1}{\epsilon(1-2r)} \log \frac{1}{\delta})$ and its failure probability is at most $\delta'(r, k) = \frac{1}{(\frac{1-r}{r})^k + 1} < \delta$.*

The lemma has an interesting interpretation: it shows that without any prior knowledge of the value of r and without explicitly trying to find it, Naive Beat-By-K improves on the Naive Majority Vote strategy and replaces one $\frac{1}{\epsilon}$ term with a $\frac{1}{|1-2r|}$ term (which can be significantly smaller). The algorithm is still worse than Stopped Hoeffding Rejection in this scenario—which is to be expected given that Stopped Hoeffding Rejection adapts dynamically to each r value. However, we introduce this provably suboptimal algorithm since it allows us to introduces the key ideas for the algorithm in the next section.

Proof. The proof follows the lines of theorem 2.3.3. Thus we know that $m \leq \frac{k(\frac{1-\epsilon}{2}, \delta)}{1-2r}$ and since $k(\frac{1-\epsilon}{2}, \delta) = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ which is what we wanted. Regarding the failure probability, we simply apply the formula $\delta' = 1 - f(k(\frac{1-\epsilon}{2}, \delta))$

where $f(k)$ is derived as in the proof of theorem 2.3.3 and get the desired relation. □

The following theorem states the properties of Naive Beat-By-K:

Theorem 3.4.5. *Naive Beat-By-K will correctly solve an LI$(\mathcal{P}, \epsilon, \delta)$ problem in:*

(i) $m = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ *expected number of samples if* $\mathcal{P} = Dirac(0.5)$.

(ii) $m = O(\frac{1}{\epsilon|1-2r|} \log \frac{1}{\delta})$ *expected number of samples if* $\mathcal{P} = Uniform\{r, 1-r\}$ *for some fixed r such that* $\epsilon < |1-2r|$.

(iii) $m = O(\frac{1}{\epsilon}(\log \frac{1}{\epsilon})(\log \frac{1}{\delta}))$ *expected number of samples if* $\mathcal{P} = Uniform(0, 1)$.

The proof is very similar to the one for theorem 3.4.3 (except for slightly different algebraic manipulations for part (iii)) and we omit it.

3.4.4 Averaged Beat-By-K

All the algorithms introduced so far in this chapter actually offer a stronger guarantee than we requested in the definition of the Label Identification problem: that for *any* sampled object, the failure probability is smaller than δ. Moreover, they are not taking direct advantage of possible knowledge of the distribution $\mathcal{P}$. Of course, as a function of $\mathcal{P}$, the expected sample complexities of a fixed algorithm vary, but the algorithms themselves are agnostic to various distributions.

The above observations lead to two questions: (1) can we do better if the failure probability is allowed to vary as a function of the sampled object? and (2) if we know the distribution $\mathcal{P}$, can we take advantage of such knowledge? We note that the answer to the second question is not immediate: even if we

know $\mathcal{P}$ it is not obvious how to use that knowledge to label a particular sampled object o_r.

The goal of this section is to present an algorithm that takes advantage of both opportunities. The key idea is to observe that when setting up the k value for Naive Beat-By-K, the actual failure probability for various r values can be significantly smaller than δ. The value of k that is set up in lemma 3.4.4 ensures that in the worst case, when all the probability mass of $\mathcal{P}$ is on $\{\frac{1-\epsilon}{2}.\frac{1+\epsilon}{2}\}$, the algorithm correctly solves the LI$(\mathcal{P}, \epsilon, \delta)$ problem. But, if $\mathcal{P}$ has a more spread out distribution we can actually pick a smaller k value in such a way that the failure probability averages to δ with the advantage of decreasing the sample complexity.

Formally, we can define a constrained optimization problem over the space of k values (that define the possible Beat-By-K strategies):

$$\text{Find } k \text{ that minimizes } m_{\text{Naive Beat-By-K}(k)}(\mathcal{P}, \epsilon, \delta) \text{ such that}$$
$$P(\text{failure}) = \frac{\int_0^{\frac{1-\epsilon}{2}} \delta(r,k) f_{\mathcal{P}}(r) dr + \int_{\frac{1+\epsilon}{2}}^1 \delta(r,k) f_{\mathcal{P}}(r) dr}{\int_0^{\frac{1-\epsilon}{2}} f_{\mathcal{P}}(r) dr + \int_{\frac{1+\epsilon}{2}}^1 f_{\mathcal{P}}(r) dr} \leq \delta \tag{3.9}$$

We first discuss the various terms and then the optimization objective. For a fixed but arbitrary k, Naive Beat-By-K is the combination of Beat-By-K with the Naive Majority Vote stopping strategy (similar to the previous section, except k is not determined by ϵ).

$m_{\text{Naive Beat-By-K}(k)}(\mathcal{P}, \epsilon, \delta)$ is the expected sample complexity of such a strategy when the expectation is taken with respect to sampling an object o_r from $\mathcal{P}$ and with respect the samples from Bernoulli(r).

$\delta(r,k)$ is the failure to correctly label an object o_r when using the algorithm with parameter k ($\delta(r,k) = \frac{1}{(\frac{1-r}{r})^k+1}$ for $r < 0.5$ according to lemma 3.4.4).

$f_{\mathcal{P}}(r)$ is the density of r according to distribution $\mathcal{P}$—where we assume, as before, that $\mathcal{P}$ is absolutely continuous, with the discrete case being handled similarly.

We only integrate the failure probability over the range $(0, \frac{1-\epsilon}{2}) \cup (\frac{1+\epsilon}{2}, 1)$ since the goal of getting the correct label with high probability applies only to objects in this range (see also the discussion about δ in section 3.2.3). The normalization factor makes sure that the failure probability is conditioned correctly.

It is not difficult to prove that the failure probability is a monotonically decreasing function of k and the expected sample complexity is an increasing function of k. The intuition is that for a fixed bias, the time it takes a random walk to reach one of two symmetric absorbing barriers increases with the increase of the absolute values of the barriers.

It is thus enough to find the minimal k value such that the constraint on the failure probability is fulfilled (since it also lead to the minimal achievable sample complexity for this class of strategies).

There are two main difficulties with performing the optimization mentioned above. The first difficulty is that we need to know $\mathcal{P}$ (or some approximation of it) to be able to perform the search for k. While assuming knowledge of $\mathcal{P}$ is not realistic, it is possible to approximate it with the empirical distribution over biases determined by labeling objects from a portion of the dataset of objects we want labeled. Then, the empirical distribution can replace the true distribution for deriving a reasonable k value.

The second challenge is technical and determined by the difficulty to analytically solve the inequality 3.9 for any non-trivial distribution $\mathcal{P}$. We address

this issue by searching for the best k value numerically (by simulating the sampling of objects from distribution $\mathcal{P}$) as described in the experimental results (section 3.5).

As a final observation, we note that in the same manner that Beat-By-K was modified for Averaged Beat-By-K, the (Early) Majority Vote algorithm can also be modified for an Average (Early) Majority Vote strategy that solves the Label Identification problem. We show empirically that the domination of Beat-By-K over (Early) Majority Vote from the Stochastic Dilemma setting holds true for this harder problem as well.

3.5 Experimental Results

The goal of this section is to investigate empirically the performance of the algorithms in two synthetic and two real domains. The key result is that Averaged Beat-By-K uniformly dominates competing strategies for all domains and settings of the parameters we attempted. A second interesting result is that numerically optimized versions of the Stopped Confidence Sequences algorithm perform very well in practice.

Stopping rules. In most of the experiments, the algorithms used as a stopping rule the Naive Majority Vote algorithm with parameters $\epsilon = 0.1$ and $\delta = 0.1$. This is essentially equivalent to establishing a maximum budget for labeling a fixed object for any algorithm to 600 samples. As we described in section 3.2.1, establishing a maximum budget is necessary for strategies like Stopped Hoeffding Rejection to be guaranteed to stop.

A common decision in practical situations is to set a maximum budget a

priori (with values usually between 10 and 100) and then stop a labeling strategy as soon as the number of samples reaches the budget. We verified that the empirical dominance of Naive Beat-By-K holds for this class of stopping strategies as well.

Algorithms. One key observation regarding strategies in the family of Stopped Hoeffding Rejection (which includes Stopped Lazy Hoeffding Rejection and Stopped Confidence Sequences which are defined in a similar manner to Stopped Hoeffding Rejection) is that they tend to be very conservative empirically in the following sense: whatever parameter δ they receive as an input, the empirical failure probability will often be much smaller than δ. The problem is that this benefit comes at the cost of a significant increase in sample complexity. This is not surprising given that the decisions of these algorithms are conservative, but it is an unwanted consequence in the sense that we dont have the desired fine tuned control over the failure probability parameter.

While (if we ignore setting a stopping strategy) strategies like Stopped Hoeffding Rejection do not have any parameters, they are unusable in a lot of practical problems due to their large sample complexity. To address this issue, we decided to numerically optimize (for a fixed problem) for the numerical constant in the confidence interval definitions of these strategies (for which the theory only sets a rough upper bound). This is roughly analogous to optimizing for the parameters k and m for the Averaged Beat-By-K or Averaged (Early) Majority Vote strategies in the sense that the optimization is only relevant for a fixed distribution $\mathcal{P}$ (corresponding to a fixed problem). Being able to perform this optimization requires access to distribution $\mathcal{P}$ (in the same way as for Averaged Beat-By-K).

Concretely, for Stopped Hoeffding Rejection (and similarly for Stopped

Lazy Hoeffding Rejection and Stopped Confidence Sequences), we change the definition of α_t to $\alpha_t = C\sqrt{\frac{\log \frac{t}{\delta}}{t}}$ where the constant C is to be determined by an exhaustive line search in some range of real numbers. The failure probability monotonically decreases as C increases. The reason is that larger C values correspond to looser confidence intervals, which in turn delays a stopping decision with the benefit of increasing accuracy but with the drawback of increasing sample complexity. Optimizing for C allowed us to have a fine-tuned control over the trade-off between the failure probability and the sample complexity for labeling a fixed object.

We note that Stopped Confidence Sequences actually has another parameter, besides C, for which we could try to optimize numerically: the size of the initial epoch. Since we set the parameter δ to 0.1, 16 samples is sufficient (according to Theorem 2.4.7) for the size of the first epoch. While we tried other values (and report setting the size of the first epoch to just 5 samples), the right tradeoff is unclear and a deeper theoretical understanding is needed to get better insights. Given the excellent empirical performance of the numerically optimized version of Stopped Confidence Sequences, we think this is an interesting problem to study further.

Notation. To improve the readability of the plots, we use the following acronyms for the algorithms in the rest of the section: BBK (Averaged BBK), (E)MAJ (Averaged (Early) Majority Vote), (L)HR (Optimized Stopped (Lazy) Hoeffding Rejection), CSQ16 and CSQ5 (Optimized Stopped Confidence Sequences with initial epoch set to 16 samples and respectively to 5 samples), Standard (L)HR (the standard version of Stopped (Lazy) Hoeffding Rejection) and Standard CSQ (the standard version of Stopped Confidence Sequences).

For readability purposes, we plot the comparison between the performance

of BBK with(E)MAJ separately from the comparison of BBK with (L)HR and CSQ.

Experimental protocol. All the algorithms have at most one free parameter for which we optimize. For every fixed problem, algorithm and parameter, we sampled from the distribution $\mathcal{P}$ (which is problem specific) 30k objects and used the algorithm to label each one. We computed for each such experiment the average failure probability and the average sample complexity.

In all situations both the average failure probability and sample complexity are monotonic functions (decreasing, respectively increasing) with respect to the parameter we are optimizing for. This characteristic of the algorithms allows us to plot the dependence of the failure probability as a function of the sample complexity as the parameter increases (or decreases) for a fixed algorithm. All the plots in this section have this form. Each plot allows one to infer for a fixed failure probability what is the algorithm with the minimum (expected) sample complexity or alternatively, for a fixed (expected) sample complexity, what is the algorithm that achieves the minimum failure probability.

Another way to interpret a curve for an algorithm in a plot is as a Pareto frontier for the solution of the algorithm to the 2-variable minimization problem with parameters δ and SC (sample complexity).

Finally, the distribution $\mathcal{P}$ corresponding to a real problems is actually an empirical distribution with a subset of the desired objects to be labeled sampled from the true distribution. To verify that we dont overfit this particular dataset, we kept a separate dataset and verified that the best choices of parameters (for a particular δ) lead to a similar performance on another dataset sampled from the same underlying distribution. We refer the reader to the

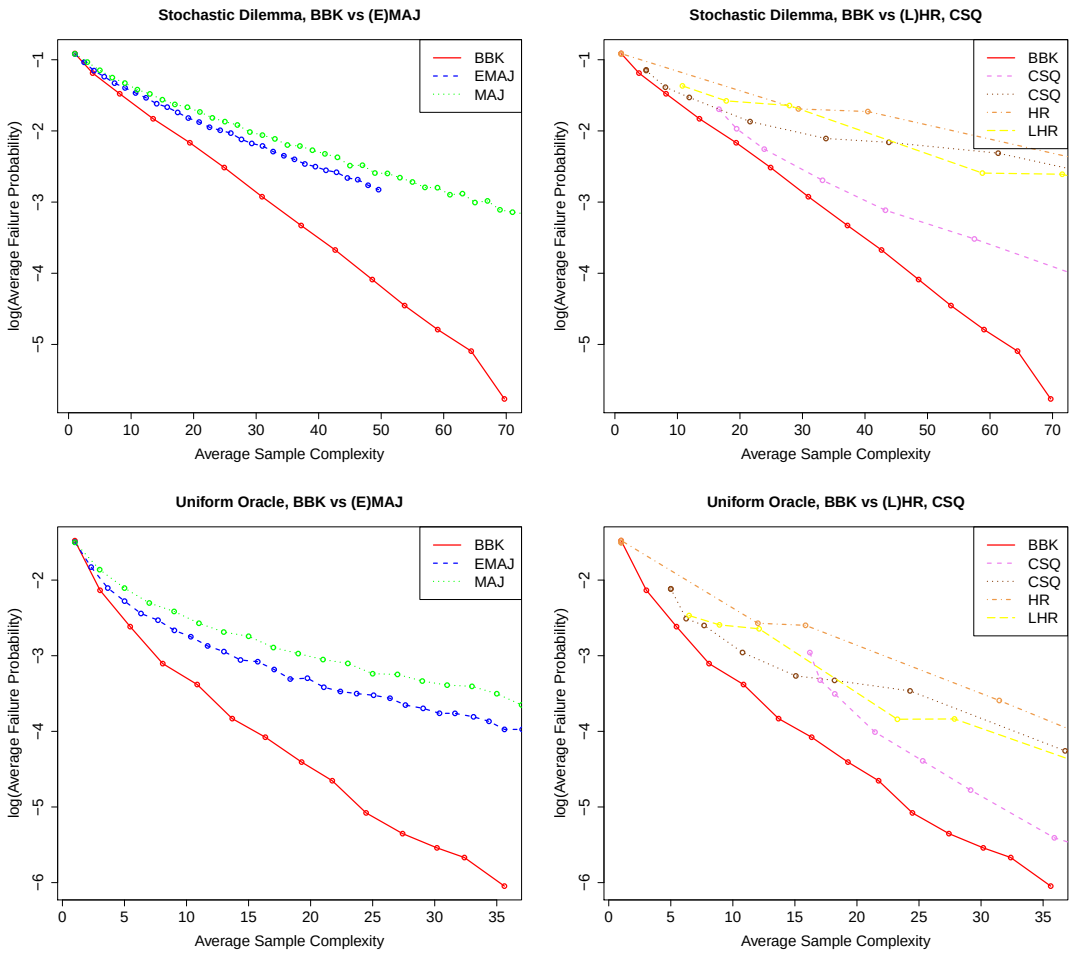

Figure 3.1: Results of comparing (a) BBK and (E)MAJ (top left), (b) BBK and (L)HR, CSQ (top right) for a Stochastic Dilemma oracle with parameter $p = 0.4$ and similarly (c) (bottom left), (d) (bottom right) for a uniform oracle.

results described in table 3.1 for more details.

3.5.1 Synthetic Domains

Stochastic Dilemma Oracle. In the first synthetic experiment every object is sampled from a Uniform distribution over $\{p, 1-p\}$ where $p = 0.4$ (which is a relatively high value, but small enough to be of practical relevance).

This Label Identification problem is essentially a stochastic dilemma and

the averaged versions of BBK and (E)MAJ are equivalent to their simpler versions introduced in the previous chapter. Even though they have an extra stopping rule as compared to their standard versions, we verified that this stopping rule does not get triggered for the range of the parameters we tested for BBK and (E)MAJ (due to the relatively small value of ϵ). The results of the comparison are plotted in figure 3.1(a). The experiment thus acts as a numerical simulation that verifies the theoretical results from chapter 2 and suggests that the dominance of Beat-By-K takes place for the entire range of δ and p values and not just for the ranges proven in theorems from section 2.3.4.

Standard CSQ ($SC = 587$, empirical $\delta = 0$), HR ($SC = 558$, empirical $\delta = 0$) and LHR ($SC = 471$, empirical $\delta = 0$), as described at the beginning of this section, tend to be conservative in terms of the number of samples they take (roughly an order of magnitude higher than for their optimized versions for $\delta = 0.1$). Moreover, the stopping rule gets triggered for a large percentage of the sampled objects ($86\%, 73\%, 29\%$ for CSQ, HR and LHR respectively). We observe that Standard LHR dominates the standard versions of HR and CSQ (which also confirms the discussion regarding the theoretical properties of these algorithms from section 2.5.2).

Finally, the performance of the optimized versions of CSQ and (L)HR is plotted in figure 3.1(b). Consistent with the results for the other domains, CSQ16 performs very well for small failure probabilities second only to BBK.

Uniform Oracle. In the second synthetic experiment we set distribution $\mathcal{P}$ to be Uniform$(0, 1)$. The results are qualitatively similar with the ones reported above. One intuitive difference (confirmed by the plots in figure 3.1(c,d)) is that the problem is slightly easier since the average object sampled from $\mathcal{P}$ is further away from 0.5 as compared to the setting of $p = 0.4$ from above.

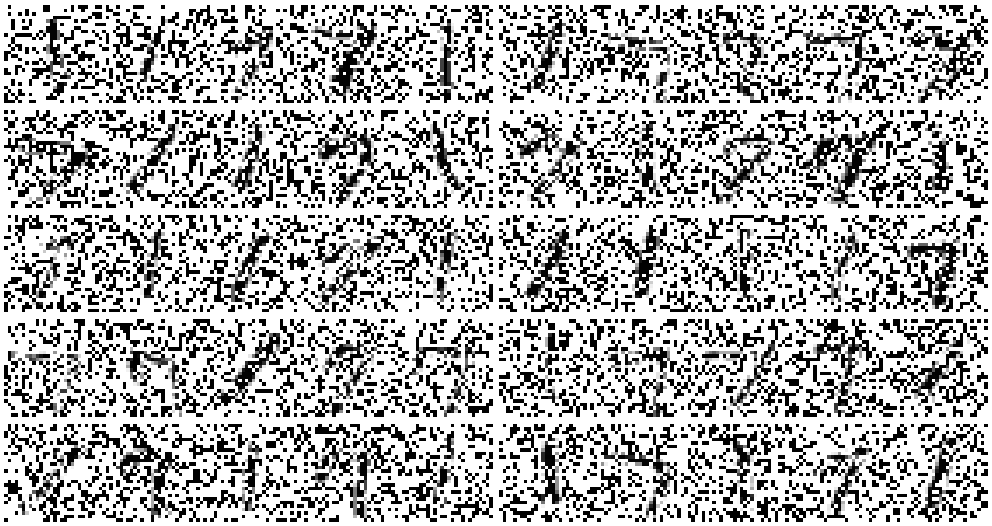

Figure 3.2: Example of noisy images submitted for labeling on Amazon Mechanical Turk.

3.5.2 Recognizing Digits

For the first non-synthetic experiment we used Amazon Mechanical Turk as a labeling oracle with the goal of comparing the algorithms in a setting involving a real crowdsourcing platform. We modified 2000 images of digits 1 and 7 from the MNIST dataset [Lecun et al., 1998] by adding random pixel noise to each image.

Every pixel in each image was flipped w.p. 31%. We chose this probability after initial experiments with various noise levels with a goal of finding a setting that lead to a difficult, but not impossible, annotation problem. We give several examples of modified images in Figure 3.2. We loaded the images as hits on Amazon Mechanical Turk and requested 11 labels for each. Every hit contained 5 images in a sequence and the workers were asked to make their best guess with respect to what was the sequence of 5 digits, while knowing that they can only be 1 and 7. We chose to group the images together to decrease the overall cost of the experiment. In total we uploaded 4400 hits on

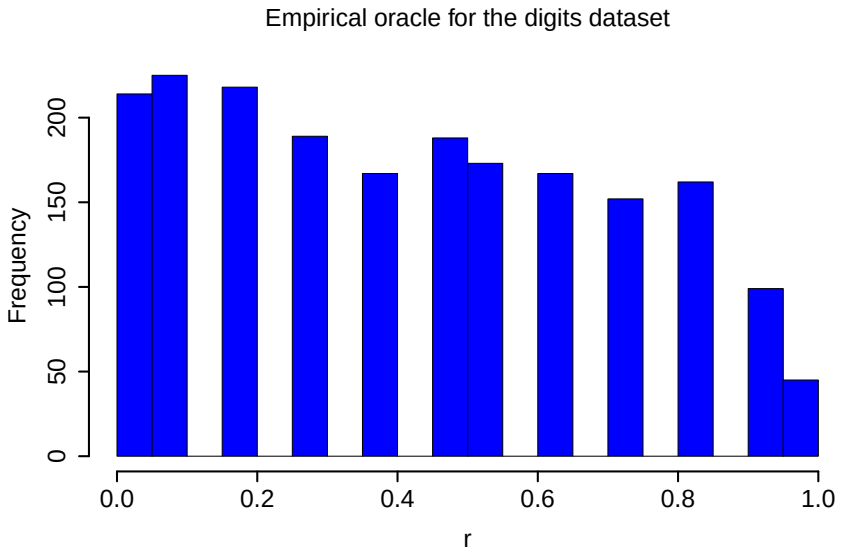

Figure 3.3: Empirical $\mathcal{P}$ oracle for the digits dataset (digit 7 is allocated to label 0 and digit 1 to label 1).

Amazon Mechanical Turk.

In an initial experiment we constrained each worker to only solve 1 hit, thus limiting the number of images labeled by a single person to 5 (with the goal of capturing as much of the diversity of workers on Amazon Mechanical Turk as possible). The problem with this approach is that it lead to the experiment being stuck for a long time (we gave up after 3 days during which only a small percentage of the hits were completed). We relaxed the constraint above to a maximum of 20 hits per worker and the outcome was that the entire set of 4400 hits was completed in 8 hours for a total cost of $66. A total of 417 workers labeled the dataset with most of the workers completing either 1 hit or 20 hits.

Given the noisy labels we obtained from the workers, we built an empirical distribution $\mathcal{P}$ (with the associated histogram plotted in figure 3.3 where a 0 label corresponds to digit 7 and a 1 label to digit 1). Concretely, for every fixed image, we considered the average vote as being the ”true” parameter for

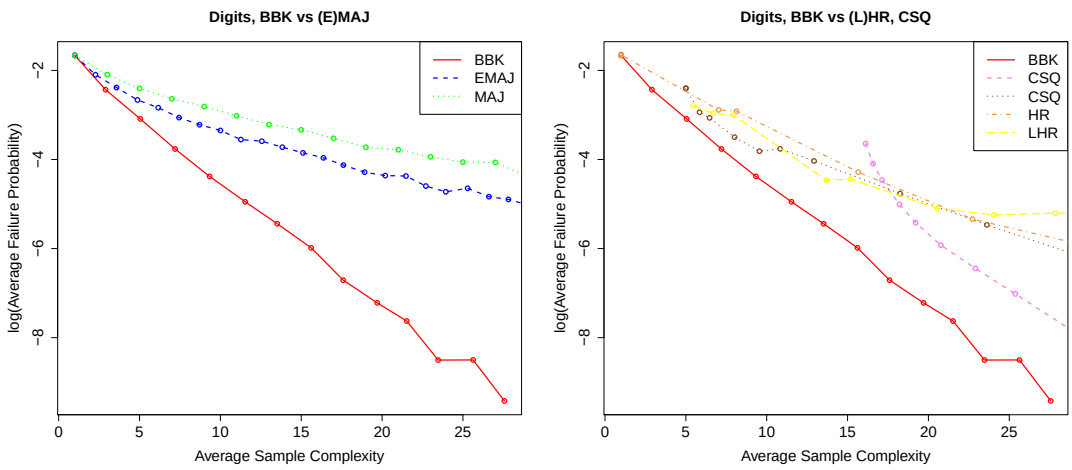

Figure 3.4: Results of comparing the algorithms on the digits dataset.

the Bernoulli distribution associated to the object with $\mathcal{P}$ being the distribution over these parameters. Even though 11 votes per image leads to only a rough approximation of the true noisy oracle, we thought it was reasonable to use $\mathcal{P}$ as an oracle when testing the algorithms. It is interesting to note that distribution $\mathcal{P}$ is asymmetric because people performed better at recognizing digit 7 than digit 1. Also, more images of digit 1 were incorrectly classified as 7, thus increasing the probability mass on objects with $r < 0.5$.

We present the results of the comparison between algorithms in figure 3.4. They are similar with the results for a uniform oracle and demonstrate BBK's strong performance in a real Label Identification problem.

3.5.3 Galaxy Zoo

The second non-synthetic experiment is concerned with labeling images of galaxies. We used the dataset Galaxy Zoo 2 [Willett et al., 2013] which was generated as part of a large scale crowdsourcing experiment concerned with obtaining labels for over 300000 galaxies from volunteers. The experiment was

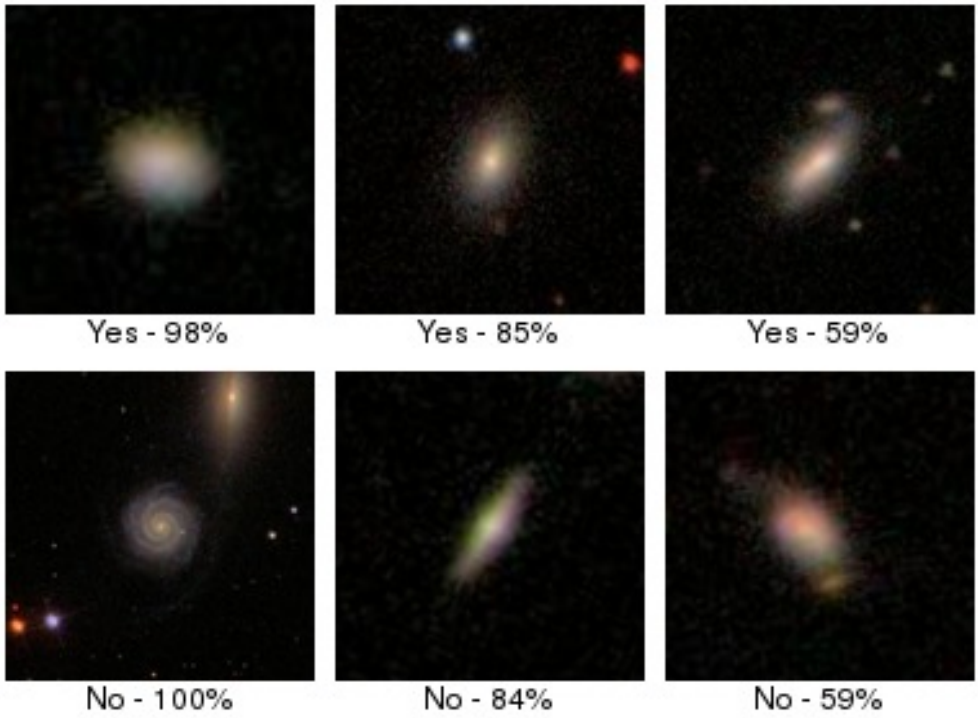

Figure 3.5: Several examples of the classification of galaxies as being smooth and round or not. The percentages represent average

designed so that for each image, several people would answer a set of questions (organized as decision tree) about basic characteristics of the galaxy in the image.

As a first example, the first question in the survey was whether a galaxy was smooth and rounded (label 1) or not (label 0) (task 1 from table 2 in Willett et al. [2013], which we transformed into a binary choice question). An average of 44 volunteers answered the question for each image in the dataset, with a volunteer being constrained to label a particular image at most once. In figure 3.5, we give several examples of images and state the percentage of agreement about the correct answer to the question, with some images being easier to label (images on the left), while others are intrinsically more difficult (images on the right). As a second example, we chose task 6 from Willett et al.

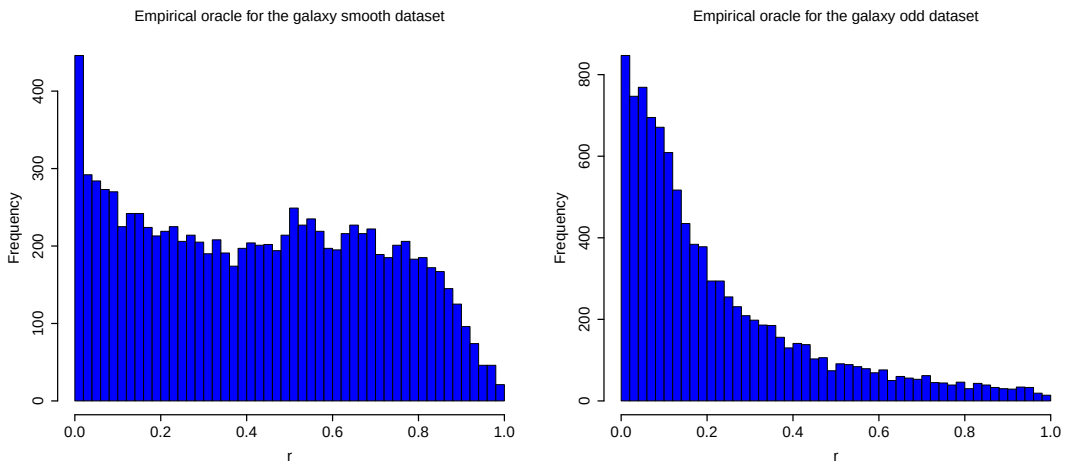

Figure 3.6: Empirical $\mathcal{P}$ oracle for the galaxy dataset for the (a) Smooth vs Non-Smooth question (left) and (b) Odd vs Non-Odd question (right).

[2013] which asked whether the images contains something odd (label 1) or not (label 0).

We sampled 10000 images from the entire dataset (and formed a training dataset) and associated with each the average vote on the correct label for the two questions described above, thus generating two different Label Identification problems. We plot the empirical $\mathcal{P}$ distributions for these two problems in figure 3.6 (both distributions are skewed towards 0-labeled objects). The results of comparing the algorithms are plotted in figure 3.7 and are essentially identical to the results for the other domains in this section.

Since we are optimizing each algorithm on the initial dataset of 10000 images, there is a possibility that we overfit the training dataset and the comparison of the algorithms is not representative of their true performance. To check whether it is indeed the case that Naive Beat-By-K dominates the other strategies, we sampled a different dataset of 10000 galaxies (the testing dataset) from the entire Galaxy Zoo dataset.

We chose a target failure probability of 3.4% (the results are consistent for

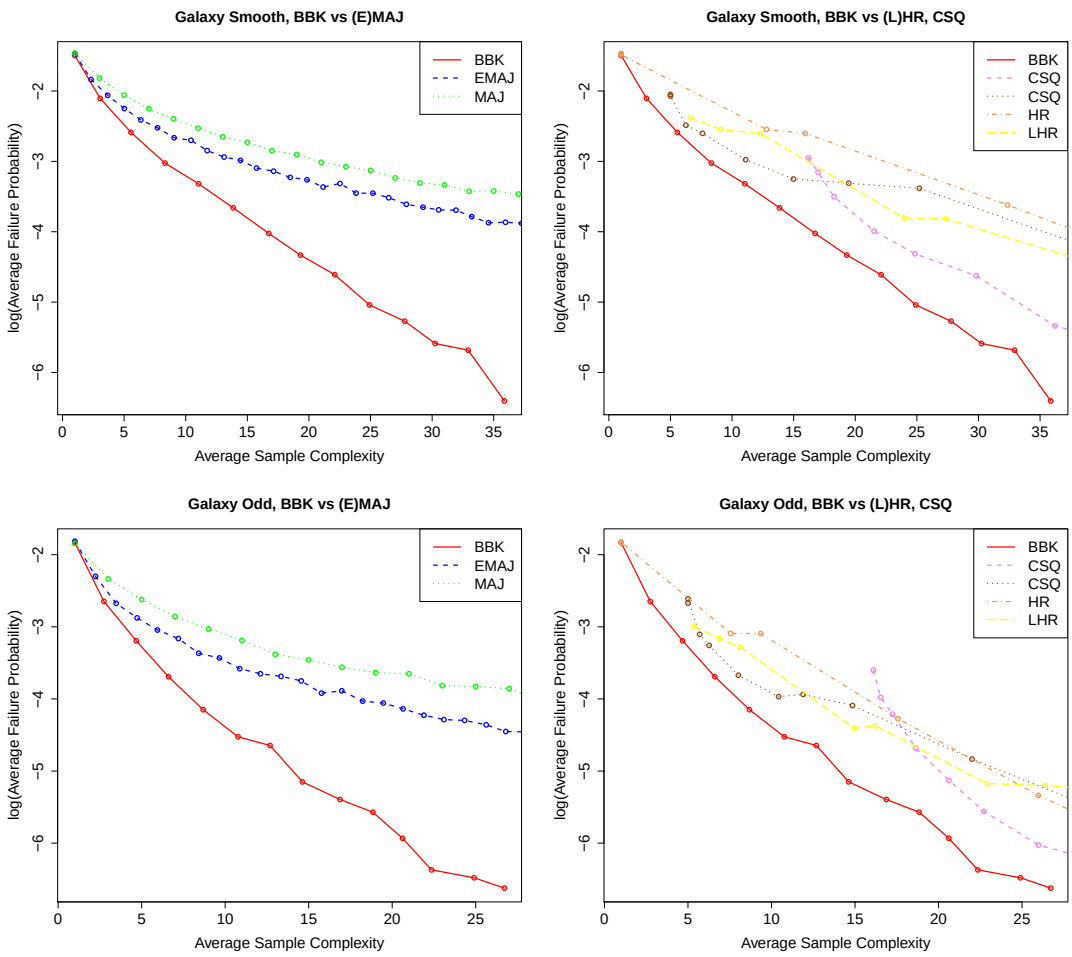

Figure 3.7: Empirical comparison of the algorithms on the galaxy smooth and galaxy odd datasets.

other choices) and picked for each algorithm the best parameter such that the empirical failure probability was at most 3.4% on the training dataset and the expected sample complexity was minimized under that constrained. We then executed each algorithm (with the parameters chosen as above) on the test dataset. We present the results in table 3.1.

The main observation is that BBK dominates the other algorithms on the test dataset both in the sense of having the minimal expected sample complexity and in the sense of having the minimal failure probability. The results confirm that we did not overfit and we can actually optimize the parameters

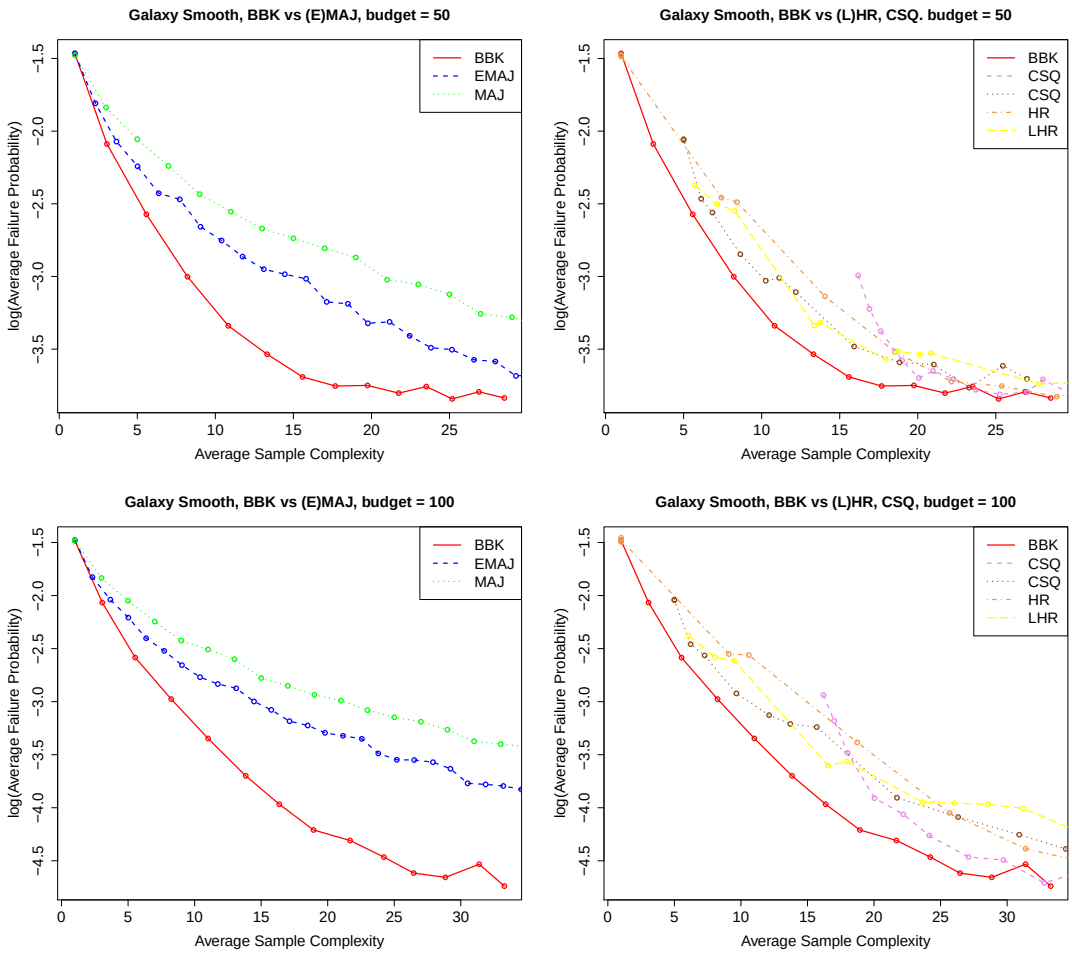

Figure 3.8: Empirical comparison of the algorithms when we set as a stopping rule a maximum budget of size 50 (top-left and top-right) and 100 (bottom-left and bottom-right)

of the algorithms based on some sampled set from $\mathcal{P}$ and then apply them for another dataset sampled from $\mathcal{P}$.

As a final experiment, we compared the performance of the algorithms in a setting where a maximum budget is set a priori to some fixed and relatively small value. We present the results for a budget of size 50 and respectively 100 in figure 3.8. The difference as compared to the previous experiments is that as the failure probability decreases, the algorithms converge to similar performances. In fact, the same effect would occur with a stopping rule based on

Algorithm	Train SC	Train δ	Test SC	Test δ
BBK	13.9	2.4%	14.1	2.3%
CSQ16	18.4	3.1%	18.5	3.2%
LHR	23.7	2.3%	24.1	2.4%
EMAJ	23.8	3.1%	23.9	3.3%
CSQ5	25.1	3.4%	25.2	3.3%
HR	31.6	2.9%	32.5	3%
MAJ	37	3%	37	3%
Std HR	187	0%	188	0%
Std LHR	162	0%	166	0%
Std CSQ	209	0%	214	0%

Table 3.1: The performance of the algorithms on the training and test datasets when the target failure probability is $\delta \leq 3.4\%$.

a parameter ϵ except for lower values of the failure probability). This phenomenon is not surprising as a limited budget constrains the set of achievable failure probabilities and sample complexities, with all the algorithms essentially converging to the performance of Majority Vote (with parameter $m =$ the budget size) as their sample complexities increase. The interesting result is that for intermediate values of δ, BBK dominates the other strategies, as in the previous experiments.

Chapter 4
Infinite Bandits

4.1 Introduction

Consider the following trivial problem. A huge jar of marbles contains some fraction ρ of black (success) marbles and the rest white (failure) marbles. We want to find a black marble as quickly as possible. If the black marbles are sufficiently plentiful in the jar, the problem is simple: Repeatedly draw marbles from the jar until a black one is found. The expected sample complexity is $\Theta(1/\rho)$. This kind of generate-and-test approach is simple, but can be extremely effective when solutions are common—for example, finding an unsatisfying assignment for a randomly generated CNF formula is well solved with this approach.

The corresponding noisy problem is distinctly more challenging. Imagine the marbles in our jar will be used to roll through some sort of obstacle course and (due to weight or balance or size) some marbles are more successful at completing the course than others. If we (quickly) want to find a marble that navigates the obstacle course successfully, how do we best allocate our test runs on the course? When do we run another evaluation of an existing marble and when do we grab a new one out of the jar? How do we minimize the (expected) total number of runs while still assuring (with high probability) that we end up with a good enough marble?

We formalize this problem as an infinite-armed bandit and provide lower and upper bounds on the number of arm pulls needed to find an arm with optimal payoff with high probability.

The focus of this chapter will be on algorithmic ideas and analysis tools that solve a simpler version of the general bandit problem. We will generalize the model in chapter 5 with the goal of applying the algorithms to solve non-trivial optimization problems.

4.2 Related Work

The framework used in our work is closely related to several models from the multi-armed bandit literature. While the case of a finite number of arms is well understood [Auer et al., 2002], in the past few years, papers discussing bandits with infinitely many arms have appeared [Kleinberg et al., 2008; Bubeck et al., 2008]. Our work extends the PAC-Bandit setting [Even-Dar et al., 2002] to an infinite number of arms [Wang et al., 2008]. It is worth noting here that a typical assumption in the literature on continuous-armed bandits (one class of infinite-armed bandits) is that the structure in the arm space induces structure in the space of expected rewards of the arms. The mean-reward function is usually assumed to be Lipschitz and algorithms are created that take advantage of this smoothness assumption. In some cases, including our example problems from chapter 5, this assumption does not hold and algorithms that depend on it can fail.

Regarding our chosen performance measure, we depart from the often used cumulative regret setting and choose a setting that only requires an agent to have a good answer after some finite experimentation. This setting is related

to some recent work [Bubeck et al., 2009; Audibert et al., 2010], but we chose the PAC-Bandit performance measure instead of the simple regret setting defined in the aforementioned papers. Besides existing Stochastic Dilemma (chapter 2) and PAC-algorithms, we also draw algorithmic inspiration from the empirical success of the Biased Robin algorithm from the Budgeted Learning framework [Madani et al., 2003], about which our analysis may offer some insight. Applicationwise, we target policy search and planning under uncertainty.

We chose to remove the parameter ϵ from the definition of the models for similar reasons as in the Stochastic Dilemma chapter: to allow us to focus on designing algorithmic strategies that are able to adapt to unknown p values. Most of the focus in the previous papers on PAC Bandits (with the exception of the Successive Elimination algorithm from Even-Dar et al. [2002]) was on the worst case scenario—where for an approximation parameter $\epsilon \in (0, 1)$, algorithms were designed to be optimal when the difference between the optimal and the suboptimal arms is exactly ϵ or less.

While this is of course interesting theoretically, the main challenge in practical settings is developing reasonable ways of handling non-worst case scenarios, where the difference between arms is significantly larger than ϵ. We note that the optimal algorithm in the finite PAC Bandit setting (Median Elimination from Even-Dar et al. [2002]) is non-adaptive in the sense that it will always take the same number of samples irrespective of the actual difference between the arms, which renders it too conservative for most practical applications.

While we will take a similar approach to Median Elimination in terms designing a new algorithm (Greedy Rejection) that focuses on being optimal the worst case scenario for an infinite armed bandit setting (section 4.6), we will

also discuss how to use the algorithm in the case where p is unknown (section 4.7) and we will show how the key insight leads to strong results empirically in the following chapter (5).

4.3 Models

We will discuss three bandit models in this section, in increasing order of difficulty in terms of the number of arms: finite bandits with 2 arms, finite bandits with $n > 2$ arms (for some parameter n) and infinite-armed bandits. Similarly to chapter 2, we chose to simplify the models as compared to the general bandit setting in the sense of only having two types of Bernoulli arms (suboptimal and optimal) with expectations symmetrically positioned with respect to 0.5.

As in the stochastic dilemma setting, this is an assumption regularly used to prove lower bounds for bandit algorithms. As we will see, this constraint also leads to a type of theoretical analysis that easily extends to more general models. The threshold 0.5 between suboptimal and optimal arms is chosen because it leads to the hardest possible setting for Bernoulli reward distributions—i.e. it leads to the largest number of samples needed to estimate the parameter of the reward distribution accurately.

4.3.1 2-Armed Bandit

We will use a similar notation as in section 2.2. Thus, we are given a set of distributions $D = \{\mathcal{D}_0 = \text{Bernoulli}(p), \mathcal{D}_1 = \text{Bernoulli}(1-p)\}, p \in (0, 0.5)$. And let's assume that we have two arms a_1 and a_2 available with associated reward distributions $\mathcal{D}_{a_1} \neq \mathcal{D}_{a_2} \in D$ and we are given sampling access to the two distributions. We are also given an acceptable failure probability $\delta \in$

$(0, 0.5)$ as a parameter.

The 2-Armed Bandit problem is to find which of the two arms has an associated reward distribution $\mathcal{D}_1$ (which corresponds to the higher expected reward and hence is the "optimal" arm) and return it with failure probability at most δ while minimizing the expected number of samples $m(p, \delta)$ from the distributions $\mathcal{D}_{a_1}$ and $\mathcal{D}_{a_2}$.

When referring to this model, for ease of notation, we will call it 2PAC(p, δ).

4.3.2 PAC Bandit

The PAC Bandit model is an extension of the 2-Armed Bandit problem from 2 to n arms. In this new setting, only one out of the n distributions is $\mathcal{D}_1$ and the rest are all $\mathcal{D}_0$. The goal remains the same as in the 2PAC(p, δ) model: to find the optimal arm with failure probability at most δ while minimizing the expected total number of samples from all distributions $\mathcal{D}_{a_i}$ (for $i \in [n]$).

We note that the model is essentially a constrained version of the general PAC Bandit model introduced in Even-Dar et al. [2002]. That being said, most lower and upper bounds in their setting are proved in the model we are using here. The intuition is that this model captures the worst case scenarios of the PAC Bandit problem. We will label this model nPAC(n, p, δ).

4.3.3 Infinite PAC Bandit

Finally, the nPAC(n, p, δ) model can be extended to an infinite number of arms. A natural extension is to assume that an infinity of arms are available with a certain proportion of them being optimal and with the rest being suboptimal.

Let's note the percentage of optimal arms (i.e. arms with a corresponding reward distribution $\mathcal{D}_1$) with $\rho \in (0, 0.5]$.

In the model, we assume that an algorithm is given sampling access to a categorical distribution $\mathcal{P}$ over D with probability ρ on $\mathcal{D}_1$ and $1 - \rho$ on $\mathcal{D}_0$. At every step an algorithm has three choices:

1. it can request a new arm from $\mathcal{P}$ in which case it receives a new arm a_i (represented through a unique id that is available for future samples from the arm's distribution) and a sample from the distribution $\mathcal{D}_{a_i}$ associated to the arm a_i.

2. it can request a new sample from the distribution associated to a previously seen arm.

3. it can stop and recommend one of the sampled arms as being the optimal one.

The goal of the algorithm is to find an optimal arm with failure probability at most $1 - \delta$ while minimizing the expected total number of samples. We will label this model IPAC(ρ, p, δ).

We note that an nPAC(n, p, δ) model can be viewed as an IPAC(ρ, p, δ) model by having $\mathcal{P}$ be a uniform distribution over the n arms and $\rho = \frac{1}{n}$. This points to a difference between the two models: in the finite model, we implicitly know the number of arms n, while in the infinite model, ρ may be unknown. Most of the algorithmic strategies we will discuss are not assuming knowledge of ρ and are thus more general. We will come back to this discussion in the following section.

4.4 On Reductions

In this section we will discuss several reductions between stochastic dilemmas, finite and infinite bandit models. While these reductions are not always optimal, they provide valuable insights about the relations between the problems and the common algorithmic and analysis ideas.

4.4.1 From Stochastic Dilemmas to 2-Armed Bandits

We will first show that 2PAC(p, δ) and SD(p, δ) can be reduced to each other and the reductions are optimal, thus proving that the two problems are essentially equivalent. The proof is straightforward in the direction SD$(p, \delta) \rightarrow$ 2PAC(p, δ) but slightly more involved in the opposite direction. We note that the proofs apply in both cases of known or unknown p (but for a fixed state of knowledge in each theorem statement).

Theorem 4.4.1 (SD$(p, \delta) \rightarrow$ 2PAC(p, δ))**.** *Given an algorithm A that solves a SD(p, δ) problem with expected sample complexity $m(p, \delta)$, we can use it to solve a 2PAC(p, δ) problem with sample complexity $m(p, \delta)$.*

Proof. We will pick arbitrarily one of the two arms (say a_1) from the 2PAC(p, δ) setting and use A to decide whether $\mathbb{E}[x]_{x \sim \mathcal{D}_{a_1}} < 0.5$ or > 0.5. In the first case arm a_2 is recommended as optimal while in the second case a_1 is recommended as optimal. The sample complexity remains m while the correctness of this 2PAC(p, δ) strategy is implied by the correctness of A. □

Theorem 4.4.2 (2PAC$(p, \delta) \rightarrow$ SD(p, δ))**.** *Given an algorithm A that solves a 2PAC(p, δ) problem with expected sample complexity $m(p, \delta)$, we can use it to solve a SD(p, δ) problem with sample complexity $m(p, \delta)$.*

Proof. We first note that the SD(p, δ) problem forces one to take all the samples from a fixed unknown distribution $\mathcal{D}$ that is uniformly sampled from the set of distributions D so we can't directly use an arbitrary 2PAC(p, δ) algorithm (since the algorithm might sample two arms according to some strategy, while we only have one arm available).

But we can use $\mathcal{D}$ to simulate the other possible distribution from D. Let's consider two arms: a corresponding to $\mathcal{D}$ and a 'virtual' arm a' corresponding to a distribution $\mathcal{D}'$ that is obtained by flipping the results of samples from distribution $\mathcal{D}$ (the two sets of samples are treated separately and are iid). We give as an input to algorithm A the two arms and let it decide which is the optimal one in m samples with failure probability $\leq \delta$. If a is returned as optimal, the answer to the SD(p, δ) problem is Bernoulli$(1 - p)$, while in the opposite case the answer will be Bernoulli(p). □

Finally, the reductions are optimal since for both problems the theorems hold for the special case where A is an optimal algorithm for one of the two settings.

4.4.2 From Stochastic Dilemmas to PAC Bandits

We will now focus on how to use stochastic dilemmas to solve the finite PAC Bandit problem. The following reduction is straightforward, but it is not optimal (as described in detail in Even-Dar et al. [2002], better algorithms exist for the nPAC(n, p, δ) problem). As in the stochastic dilemma case, the proof holds for a fixed state of knowledge of p (p is either known or unknown for both problems).

Theorem 4.4.3 (SD$(p, \delta) \rightarrow$ nPAC(n, p, δ))**.** *Given an algorithm A that solves a*

SD(p, δ) *problem with (expected) sample complexity* $m(p, \delta)$*, we can use it to solve a* *nPAC*(n, p, δ) *problem with (expected) sample complexity* $m' = nm(p, \frac{\delta}{n})$.

Proof. Let's apply A with parameter $\frac{\delta}{n}$ for each arm a_i. If only one arm is labeled Bernoulli$(1-p)$ by A, it will be recommended as optimal. If more than one arm or no arm is labeled Bernoulli$(1-p)$, any arm is chosen arbitrarily as optimal. The probability that at least one arm is "labeled" incorrectly by A is at most $\sum_{i=1}^{n} \frac{\delta}{n} = \delta$ via an application of union bound for individual failures. Thus the algorithm will return the correct answer w.p. at least $1-\delta$. The sample complexity is simply the sum of the sample complexities for applying A to the n arms (and we can use linearity of expectation in the case that m is the expected value of a random variable). □

4.4.3 From Stochastic Dilemmas to Infinite PAC Bandits

Finally, we will now cover the reduction of the IPAC(ρ, p, δ) problem to SD(p, δ). The key difference with the reduction from the previous section is that in general we don't know the value of ρ (whereas the corresponding complexity parameter n from nPAC(n, p, δ) is implicitly known to the algorithm). This implies that one needs to use SD(p, δ) solutions in a different manner.

The high level strategy we will outline is to transform the IPAC(ρ, p, δ) problem into an **iterative stochastic dilemma**. As the name suggests, arms will be sampled and "labeled" iteratively and as soon as an "optimal" arm is found the algorithm will stop. While this reduction is not optimal (see sections 4.6 and 4.7), it leads to practical algorithms with good empirical performance (see the experiments in chapter 5).

To be concrete, we start with index $i = 1$ and:

1. Sample an arm a_i from $\mathcal{P}$.

2. Use A to decide whether a_i is optimal w.p. $\frac{\delta}{2i^2}$.

3. If a_i is declared optimal, stop, else $i = i + 1$ and return to step 1.

Now, if N is a random variable that denotes the number of arms an algorithm samples from the distribution $\mathcal{P}$ over the set D until stopping, we will prove that:

Theorem 4.4.4. *Given an algorithm A that solves a $SD(p, \delta)$ problem with expected sample complexity $m(p, \delta)$, we can use it to solve a $IPAC(\rho, p, \delta)$ problem with expected sample complexity $m' \leq \mathbb{E}[Nm(p, \frac{\delta}{2N^2})]$ with $\mathbb{E}[N] \leq \frac{2}{\rho}$.*

where the expectations are taken both with respect to sampling arms from $\mathcal{P}$ and sampling from the reward distributions $\mathcal{D}_{a_i}$ associated with the sampled arms.

Proof. Using the same trick as in lemma 2.4.2, the probability the algorithm makes a wrong decision over the infinite horizon is bounded by $P(error) \leq \sum_{i=1}^{\infty} \frac{\delta}{2i^2} \leq \delta$ as desired.

The expected sample complexity is $m' = \mathbb{E}[\sum_{i=1}^{N} m(p, \frac{\delta}{2i^2}] \leq \mathbb{E}[Nm(p, \frac{\delta}{2N^2}]$ (where we assume that m is increasing when the failure probability is decreasing which is true for all the strategies we discussed in chapter 2).

Let S_{a_i} be the event of accepting the i'th sampled arm ($i \geq 1$), conditioned on rejecting the first $i - 1$ arms. Note that $P(S_{a_i}) = P(\text{accept arm } a_i | a_i \text{ is 'good' })P(a_i \text{ is 'good'}) + P(\text{accept arm } a_i | a_i \text{ is 'bad' })P(a_i \text{ is 'bad'}) \geq P(\text{accept arm } a_i | a_i \text{ is 'good' })P(a_i \text{ is 'good'}) \geq (1 - \frac{\delta}{2i^2})\rho \geq \frac{\rho}{2}, \forall i > 1$. Thus $E[N] \leq \frac{2}{\rho}$ (by the properties of the geometric distribution, where S_{a_i} stands for "success"). □

The theorem above implies a corollary for the performance of various iterative version of the stochastic dilemma algorithms introduced in chapter 2.

Corollary 4.4.5. *Given an IPAC(ρ, p, δ) problem:*

(i) Iterative Majority Vote and Iterative Beat-By-K use $m' = O(\frac{1}{\rho(1-2p)^2}\log\frac{1}{\rho\delta})$

(ii) Iterative Hoeffding Rejection uses $m' = O(\frac{1}{\rho(1-2p)^2}(\log\frac{1}{\rho\delta} + \log\frac{1}{1-2p}))$

(iii) Iterative Lazy Hoeffding Rejection uses $m' = O(\frac{1}{\rho(1-2p)^2}(\log\frac{1}{\rho\delta} + \log\log\frac{1}{1-2p}))$

samples on expectation to solve it.

Proof. We will only prove the result for Iterative Majority Vote as the proof is almost identical for the other algorithms. Using theorem 4.4.4 for m' and given that for Majority Vote $m = \frac{C}{(1-2p)^2}\log\frac{1}{\delta}$ for some constant $C > 0$ (from theorem 2.3.2) we get that:

$$m' \leq \mathbb{E}[\frac{CN}{(1-2p)^2}\log\frac{2N^2}{\delta}] \tag{4.1}$$

$$= \frac{C}{(1-2p)^2}\log\frac{2}{\delta}(\mathbb{E}[N] + 2\mathbb{E}[N\log N]) \text{ (linearity of expectation)} \tag{4.2}$$

We can show that:

$$\mathbb{E}[N\log N] \leq \sqrt{\mathbb{E}[N^2]\mathbb{E}[\log^2 N]} \tag{4.3}$$

$$\leq \sqrt{4\mathbb{E}[N]^2\mathbb{E}[\log^2 N]} \tag{4.4}$$

$$\leq \sqrt{4\mathbb{E}[N]^2\log^2\mathbb{E}[N]} \tag{4.5}$$

$$= 2\mathbb{E}[N]\log\mathbb{E}[N] \tag{4.6}$$

where 4.3 is an application of Cauchy-Schwarz, 4.4 is determined by the properties of the geometric distribution with success probability S_{a_i}. For 4.5 we use the fact that the function $\log^2 x$ is concave for $x \geq 3$ and we apply Jensen's inequality to $\log^2$ to get the inequality.

And, since we know from theorem 4.4.4 that $\mathbb{E}[N] \leq \frac{2}{\rho}$, we get that $m' \leq \frac{C}{\rho(1-2p)^2} \log \frac{2}{\delta}(\frac{2}{\rho} + \frac{4}{\rho} \log \frac{2}{\rho}) = O(\frac{1}{\rho(1-2p)^2} \log \frac{1}{\rho\delta})$. □

4.4.4 From PAC Bandits to Infinite PAC Bandits

The reduction we will discuss deals with the simplest version of the IPAC(ρ, p, δ) problem—when both p and ρ are known (reductions for other versions of the problem follow along similar lines). The high level strategy is to compute how many arms one needs to sample to get an optimal arm with high probability and then apply a PAC-Bandit algorithm to select the optimal arm from the sampled ones. The following algorithm implements this strategy.

1. Sample $n' = \frac{1}{\rho} \log(\frac{2}{\delta})$ arms.
2. Execute a correct nPAC(n, p, δ) algorithm A on the nPAC$(n', p, \frac{\delta}{2})$ problem.
3. Return the output of A.

Instead of applying the standard PAC Bandit algorithms we apply an algorithm that takes advantage of the knowledge of p—the version of Median Elimination from Section 7.1 of Mannor et al. [2004] (which we label as **Median Elimination with Known Bias** (MEKB)).

Theorem 4.4.6. *If the reduction algorithm is executed with MEKB as a nPAC(n, p, δ) algorithm, it solve the IPAC(ρ, p, δ) problem with expected sample complexity* $m' = O(\frac{1}{\rho(1-2p)^2} \log \frac{1}{\delta})$.

Proof. We will first prove that the reduction algorithm is correct. Let $a_1, a_2, ..., a_{n'} \sim \mathcal{P}$ i.i.d. Define event $A = \{\forall i \in \{1...n\}, a_i \text{ suboptimal}\}$ (in words, A stands for

the event that the expected value of all the sampled arms is smaller than 0.5). Then, $P_{a_i \sim \mathcal{P}, i \in \{1 \ldots n\}}(A) = (1-\rho)^n$. If we choose $n = \frac{1}{\rho}\log(\frac{2}{\delta})$, then $P(A) \leq \frac{\delta}{2}$.

When MEKB is executed with for the problem nPAC$(n, p, \frac{\delta}{2})$, the reduction strategy will fail with probability at most $\frac{\delta}{2}$. Thus, by the union bound on $P(A)$ and the failure probability of MEKB, the algorithm will fail with probability at most δ and will otherwise return an arm with the desired properties.

The sample complexity follows by using the result that MEKB takes $m = O(\frac{1}{(1-2p)^2}(n' + \log\frac{1}{\delta}))$ samples on expectation. □

While the reduction leads to a suboptimal algorithm (since the lower bound from 4.5 is a logarithmic factor smaller and is met by another algorithm we will discuss in section 4.6), it is instructive in terms of techniques of reusing algorithms from a finite to an infinite bandit problem (the other type of reduction is to apply a doubling trick). For a more detailed discussion about the connection between the two models, we refer the reader to the discussion section (4.8).

4.5 Lower Bounds

The goal of this section is to prove a lower bound for the IPAC(ρ, p, δ) problem. We will only prove a lower bound for the easiest version of the problem: when both p and ρ are known. The bound is thus a valid lower bound (although potentially loose) for the more difficult problems when at least one of the complexity parameters p or ρ are unknown.

One interesting fact is that in section 4.6 we will present an algorithm that has an identical asymptotic upper bound, even though the algorithm doesn't assume knowledge of ρ. This shows that the bound is asymptotically tight for the harder setting of p known and ρ unknown.

Theorem 4.5.1. *Any algorithm A that correctly solves a IPAC(ρ, p, δ) problem has an expected sample complexity of at least $m = \Omega(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta}))$.*

Proof. The proof is by contradiction. We will assume there exists a correct algorithm A that solves any IPAC(ρ, p, δ) problem with expected sample complexity $o(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta}))$. The goal is to show that A would imply a correct algorithm for the PAC-Bandit problem with expected sample complexity $o(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta}))$, which would contradict the known lower bound in the PAC-Bandit setting.

Let's define a PAC-Bandit problem as follows: assume we are given $n = \frac{1}{\rho}$ arms, $n-1$ of which have expected reward of p and one of which has expected reward of $1-p$. To be precise, it is worth mentioning that we allow the algorithms in the PAC-Bandit setting to resample arms and ignore any previous pulls taken for those arms (this actually makes the PAC-Bandit problem harder, so the lower bound still has to hold).

When we use A for the PAC-Bandit problem, each time the algorithm samples a new arm from the environment, it selects an arm uniformly at random, with replacement, from the n arms. Applying A, it will get the optimal arm with probability at least $1-\delta$ with an expected number of samples $m = o(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta})) = o(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta}))$.

From theorem 13 in Mannor et al. [2004], we know that $m = \Omega(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta}))$. But from theorem A.3.1 (applied to functions $m(p, \rho, \delta)$ and $g(p, \rho, \delta) = \frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta})$ for complexity parameters $\frac{1}{\delta}, \frac{1}{\rho}, \frac{1}{1-2p}$), we know that the two sets of functions ($o(g)$ and $\Omega(g)$) are disjoint and thus m can't belong to both which means we obtained the desired contradiction. □

4.6 A Novel Algorithm - Greedy Rejection

In this section we will introduce a new algorithm that has a better performance than the algorithmic reductions to Stochastic Dilemmas from section 4.4. We will formally study the algorithm in an infinite-armed bandit setting with a known p value and prove that it is asymptotically optimal in this version of the problem.

The high-level idea of the algorithm is to take advantage of the fact that one doesn't need to accurately label both suboptimal and optimal arms but only needs to find the one with a higher expected value. The algorithm is similar to the Beat-By-K strategy introduce in section 2.3.2 but it uses asymmetric barriers and a different manner of sampling arms which will lead to a different type of analysis.

Similarly to Beat-By-K, let $\Delta^t = \#1\text{-samples} - \#0\text{-samples}$. The algorithm has a parameter $k(i)$ that depends on the index of the currently sampled arm a_i. It starts by sampling with replacement an arm from a uniform distribution over D, and then it keeps sampling the arm until either 1. $\Delta^t = k(i)$ in which case the arm is labeled as "optimal" and the algorithm stops, or 2. $\Delta^t = -1$ in which case it labels the arm "suboptimal", it "throws" it away and samples a new one again according to a uniform distribution over D. The idea of having parameter $k(i)$ varying with index i follows from the need to control the failure probability over the entire execution of the algorithm (since we don't know a priori how many arms we need to sample to find an optimal one).

Algorithm 1: GreedyRejection(p, δ):

- Given a set of parameters depending the index i of the sampled arms $k(i) = \lceil \log_{\frac{1-p}{p}} (\frac{2i^2(1-2p)}{p\delta} + 1) - 1 \rceil$
- For every $i = 1, 2, \ldots$:
    1. Sample a new arm a_i from $\mathcal{P}$.
    2. For every $t = 1, 2, \ldots$:
        (a) Sample $x_t \sim \mathcal{D}$, update $\Delta^t =$ #1-samples $-$ #0-samples. .
        (b) If $\Delta^t = k(i)$ stop and recommend a_i as "optimal".
        (c) If $\Delta^t = -1$ declare a_i "suboptimal", set $i = i + 1$, go to step 1.
        (d) Otherwise set $t = t + 1$, go to step 2a.

Parts of the analysis are similar to Beat-By-K, but because arms are repeatedly sampled, as opposed to just one arm being "labeled", the proof becomes significantly more involved.

Theorem 4.6.1. *If the Greedy Rejection algorithm with the set of parameters* $k(i) = \log_{\frac{1-p}{p}} (\frac{2i^2(1-2p)}{p\delta} + 1) - 1$ *(where* i *is the index of the arm that is currently being sampled,* $i \in \mathbb{N}^*$*) is executed for an infinite bandit model* $IPAC(p, \delta, \rho)$ *(with* $\rho \leq 0.5$ *and* $p \geq 0.1$*), it will return the optimal solution with probability* $\geq 1 - \delta$ *and it will take* $m = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$ *samples on expectation.*

The key components of the proof are:

1. prove that all suboptimal arms are rejected with high probability (lemma 4.6.2).

2. prove that the expected time to reject a suboptimal arm is "small" as compared to the time to accept an optimal arm (lemma 4.6.4).

3. prove that even though "most" of the optimal arms are rejected as well (due to the aggressiveness of the rejection procedure), there remains a high probability of sampling and accepting an optimal arm (lemma 4.6.3).

4. combine the above facts with a careful handling of the upper bounds on the expected sample complexities for each scenario (i.e. rejected suboptimal arm, rejected optimal arm, accepted optimal arm) to prove the main result (lemma 4.6.7).

To get an intuition for the asymptotic dependency of k on the parameters p and δ in the setting the algorithm is intended for ("large" p, $p \geq 0.1$), we can compute k as follows: $k(i) \leq \frac{1}{\log \frac{1-p}{p}} \log \frac{4(1-2p)i^2}{p\delta} \leq \frac{1}{1-2p} \log \frac{40(1-2p)i^2}{\delta} = O(\frac{1}{1-2p} \log \frac{(1-2p)i}{\delta})$.

We chose to constrain $p \geq 0.1$ in the theorem statement, to have a cleaner analysis. If $p \leq 0.1$ we can obtain an asymptotic upper bound for the sample complexity that only depends on δ and ρ (which would not change the asymptotic upper bound in the theorem statement) since the "hard" IPAC(ρ, p, δ) problems are those for which p is close to 0.5 (due to the dependency of the sample complexity on $\frac{1}{(1-2p)^2}$).

We also note that we dropped the approximation of k_i to the nearest integer in the statement of the theorem to have a clearer analysis. The proof holds for discrete values of k_i with the intuition that for p close enough to 0.5 and small enough δ (for which k becomes "large"), the real value and the integer approximation become close with their difference having no impact on the asymptotic bound.

Before we continue, let's label several events and random variables that will prove useful for the various proofs related to the performance of Greedy Rejection:

- S_i = "Greedy Rejection declares arm a_i optimal".
- S_i^{opt} = "a_i is optimal AND S_i".
- S_i^{subopt} = "a_i is suboptimal AND S_i".
- R_i = "Greedy Rejection declares arm a_i suboptimal and rejects it".
- R_i^{opt} = "a_i is optimal AND R_i".
- R_i^{subopt} = "a_i is suboptimal AND R_i".
- N is a random variable that denotes the number of arms Greedy Rejection samples until one is labeled "optimal" or, alternatively, the number of sampled arms until Greedy Rejection stops.
- $\tau(i)$ is a random variable (r.v.) for the number of samples before an arm with index i is declared optimal or suboptimal.
- $\tau^{opt}(i)$ is a r.v. for the number of samples conditioned on a_i being optimal ($\tau(i)|a_i$ is optimal).
- $\tau^{subopt}(i)$ - same as above except for a suboptimal arm.

We will prove the main theorem in a sequence of lemmas. The first deals with the probability that the algorithm will fail to return the correct answer:

Lemma 4.6.2. *Greedy Rejection is correct with probability at least* $1 - \delta$.

Proof. The only way for Greedy Rejection to fail is if it recommends a suboptimal arm as being the optimal arm. The only way this event can happen is if a negatively biased random walk corresponding to the i-th sampled arm (obtained via the standard transformation from section A) corresponding to the distribution Bernoulli(p) hits the barrier at $k(i)$. Such an event would lead Greedy Rejection to recommend the arm as being "optimal". So we need to bound the total probability of such recommendations.

Let's denote by $\mathcal{N}$ the distribution over steps of a negatively biased random walk. Then, applying theorem A.2.2 with $k_0 = 1, k_1 = k(i), \beta = \frac{1-p}{p}$ and denoting $\tau(i)$ the time until stopping of the i-th arm, we obtain $P_{\mathcal{N}}(S_\tau(i) = k(i)) = \frac{\beta-1}{\beta^{k(i)+1}-1}$ and by replacing $k(i)$ with the value from the statement, we get that $P_{\mathcal{N}}(S_\tau(i) = k(i)) \leq \frac{\delta}{2i^2}$.

Let's assume the worst case scenario where the algorithm will only sample suboptimal arms over the infinite horizon. Then the probability of recommending one of them as being optimal is $P(error) \leq \sum_{i\geq 1} P_{\mathcal{N}}(S_\tau(i) = k(i)) = \sum_{i\geq 1} \frac{\delta}{2i^2} \leq \delta$. □

The next result determines the probability that an optimal arm will be correctly labeled "optimal" by the algorithm.

Lemma 4.6.3. *If the i-th sampled arm has a* Bernoulli$(1-p)$ *distribution, the probability that Greedy Rejection will stop and recommend it as optimal is* $P(S_i^{opt}) \geq 1 - 2p$.

The lemma has a strong connection with the well known Ballot Theorem from random walk theory [Addario-Berry and Reed, 2008], which states that the probability of a positively biased random walk to stay positive over the infinite horizon is positive and proportional to the bias of the random walk.

This connection is made more explicit in the following section in the proof of 4.7.3.

Proof. We again apply theorem A.2.2(i) for the positively biased random walk corresponding to the arm a_i and get that $P(S_i^{opt}) = P(S_\tau = k(i)) = \frac{\beta^{k(i)}(\beta-1)}{\beta^{k(i)+1}-1} = \frac{1-2p}{p}\frac{\beta^{k(i)}}{\beta^{k(i)+1}-1} \geq \frac{1-2p}{p}\frac{\beta^{k(i)}}{\beta^{k(i)+1}} = \frac{1-2p}{1-p} \geq 1-2p$. □

The following lemma computes the expected number of samples for a suboptimal arm.

Lemma 4.6.4. *If Greedy Rejection is executed for the i-th sampled arm and the arm a_i has an associated distribution* Bernoulli(p)*, then* $\mathbb{E}[\tau^{subopt}(i)] \leq \frac{1}{1-2p}$.

We note that the bound is a factor of $O(\frac{1}{1-2p})$ smaller as compared to the Beat-By-K expected sample complexity bound for a fixed suboptimal arm. The advantage is canceled when the sample complexity over the entire execution of the algorithm is considered (since, as we will see, an optimal arm can be rejected with significant probability), but it points to a significant difference between the two strategies.

Proof. We will deal with a negatively biased random walk corresponding to a Bernoulli(p) distribution, which can be treated symmetrically to a positively biased random walk but by reversing the barriers. Thus if we use theorem A.2.2(ii) with $k_0 = k(i) = k, k_1 = 1$ the time to hit a barrier at -1 or k is:

$$\mathbb{E}[\tau^{subopt}(i)] = \frac{\beta(\beta^k - 1) - k(\beta-1)}{(1-2p)(\beta^{k+1}-1)} \tag{4.7}$$

$$= \frac{\beta^{k+1} - 1 + 1 - \beta - k(\beta-1)}{(1-2p)(\beta^{k+1}-1)} \tag{4.8}$$

$$= \frac{1}{1-2p} - \frac{(k+1)(\beta-1)}{(1-2p)(\beta^{k+1}-1)} \tag{4.9}$$

and it follows that $\mathbb{E}[\tau^{subopt}(i)] \leq \frac{1}{1-2p}$. □

The next lemma computes the expected number of samples for an optimal arm.

Lemma 4.6.5. *If Greedy Rejection is executed for the i-th sampled arm then if the arm a_i has an associated distribution* Bernoulli(p)*, then* $\mathbb{E}[\tau^{opt}(i)] \leq 2k(i) + \frac{2k(i)}{E(i)} + \frac{1}{(1-2p)E(i)}$*, where* $E(i) = \frac{2i^2(1-2p)}{p\delta}$.

Proof. We will use theorem A.2.2 with $k_0 = 1, k_1 = k(i)$, $\frac{1-p}{p} = \beta$, and from the expression of $E(i)$ we get that $k = k(i) = \log_\beta (E+1) - 1$. Then, by using part (ii) from theorem A.2.2 for the expected time to hit a barrier at -1 or k, we get:

$$\mathbb{E}[\tau^{opt}(i)] = \frac{k\beta^k(\beta-1) - (\beta^k - 1)}{(1-2p)(\beta^{k+1}-1)} \tag{4.10}$$

$$= \frac{k\frac{E+1}{\beta}(\beta-1) - (\frac{E+1}{\beta} - 1)}{(1-2p)E} \tag{4.11}$$

$$= \frac{k}{1-2p}(1-\frac{1}{\beta}) + \frac{k}{(1-2p)E}(1-\frac{1}{\beta}) + \frac{1}{(1-2p)E} - \frac{E+1}{(1-2p)E\beta} \tag{4.12}$$

$$\leq \frac{k}{1-p} + \frac{k}{(1-p)E} + \frac{1}{(1-2p)E} \text{ (since } \frac{E+1}{(1-2p)E\beta} > 0) \tag{4.13}$$

$$\leq 2k + \frac{2k}{E} + \frac{1}{(1-2p)E} \text{ (since } p < 0.5) \tag{4.14}$$

We note that in these expressions, k and E are functions of i and we removed the dependency for clarity of exposition. □

We will now state a result about the expected number of arms that Greedy Rejection samples before it stops.

Lemma 4.6.6. *The expected number of arms Greedy Rejection samples before stopping is* $\mathbb{E}[N] \leq \frac{1}{\rho(1-2p)}$.

Proof. Using the notation at the beginning of the section, $P(S_{a_i}) = P(S_{a_i}^{opt} \vee S_{a_i}^{subopt}) = P(S_{a_i}^{opt}) + P(S_{a_i}^{subopt})$ (since the two events are mutually exclusive).

Then $P(S_{a_i}) \geq P(S_{a_i}^{opt})$. But since $P(a_i \text{ optimal})$ is ρ (since Greedy Rejection samples the two arms according to the distribution corresponding to the IPAC(p, δ, ρ) model), $P(S_{a_i}^{opt}) = P(a_i \text{ declared optimal} \,|a_i \text{ optimal}) P(a_i \text{ optimal}) = \rho P(a_i \text{ declared optimal} \,|a_i \text{ optimal}) \geq \rho(1-2p)$ (based on lemma 4.6.3)

Since N is a random variable that denotes the number of arms sampled until one is labeled "optimal", we can interpret N as the number of times we need to wait for a success in samples from a geometric distribution with success probability $P(S_{a_i})$ (since the first time an arm is labeled "optimal" by Greedy Rejection, the algorithm stops). Then $\mathbb{E}[N] = \frac{1}{P(S_{a_i})} \leq \frac{1}{\rho(1-2p)}$. □

We are now ready to prove the final result concerning the expected sample complexity of the Greedy Rejection algorithm. We note that the proof is relatively long and convoluted. The algebraic manipulations could be simplified significantly by removing the $1-2p$ term in the expression for $k = \log_{\frac{1-p}{p}}(\frac{2i^2(1-2p)}{p\delta} + 1) - 1$ at the cost of adding an extra $O(\log \frac{1}{1-p})$ in the sample complexity bound which would mean loosing the asymptotic tightness of the algorithm. The practical cost would probably be small in most situations, but from a theoretical perspective there is an obvious justification in looking for the tightest analysis possible.

Lemma 4.6.7. *The expected sample complexity of Greedy Rejection is* $m(p, \delta) = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$.

Proof. **Part 1**. In the first part of the proof we will derive an upper bound for the expected number of samples m. In what follows the expectations are taken with respect to both sampling arms from the distribution over the set of distributions D corresponding to the IPAC(p, δ, ρ) model (that contains only Bernoulli(p) and Bernoulli$(1-p)$) and to sampling a fixed arm with the goal

of deciding whether it is optimal or not.

$$m = \mathbb{E}[\sum_{i=1}^{N} \tau(i)] \tag{4.15}$$

$$= \sum_{n=1}^{\infty} \mathbb{E}[\sum_{i=1}^{N} \tau(i)|N = n]P(N = n) \tag{4.16}$$

by applying the law of total expectation (since the set of events $\{N = n|n \geq 1\}$ forms a partitioning of the sample space). We note that we can't apply Wald's identity (which would simplify the algebra) to split equation 4.15 (into a term that depends only on N and another term that depends only on $\tau(i)$) as the random variables $\tau(i)$ are not identically distributed. Then:

$$m = \sum_{n=1}^{\infty} \mathbb{E}[\sum_{i=1}^{n} \tau(i)|N = n]P(N = n) \text{ (due to conditioning)} \tag{4.17}$$

$$= \sum_{n=1}^{\infty} (\sum_{i=1}^{n} \mathbb{E}[\tau(i)|N = n])P(N = n) \text{ (linearity of expectation)} \tag{4.18}$$

We note that $\tau(i), \forall i \leq n$ is not independent of $N = n$ as the latter event constrains what arms are accepted or not at the indices $i \leq n$. So we will split the sum in two parts:

$$m = \sum_{n=1}^{\infty} (\sum_{i=1}^{n-1} \mathbb{E}[\tau(i)|N = n \wedge i < n])P(N = n) + \sum_{n=1}^{\infty} \mathbb{E}[\tau(n)|N = n]P(N = n) \tag{4.19}$$

Now, the only two pieces of information in the event $N = n \wedge i < n$ that $\tau(i)$ is dependent upon are 1. the index of the arm i and 2. the fact that since the stopping time of the algorithm is at an index higher than the current index $i < n$, it has to be the case that the arm at index i was rejected. In other words,

$$\mathbb{E}[\tau(i)|N = n \wedge i < n] = \mathbb{E}[\tau(i)|a_i \text{ declared suboptimal}], \forall i < n \tag{4.20}$$

$$\leq \frac{\mathbb{E}[\tau(i)]}{P(a_i \text{ declared suboptimal })} \tag{4.21}$$

where the inequality is due to the law of total expectation.

Similarly, $\tau(n)$ is dependent on the 1. index of the arm n and 2. the fact that arm a_n had to be accepted since the stopping time $N = n$. So, as above:

$$\mathbb{E}[\tau(n)|N = n] = \mathbb{E}[\tau(n)|a_n \text{ declared optimal }] \tag{4.22}$$

$$\leq \frac{\mathbb{E}[\tau(n)]}{P(a_n \text{ declared optimal })} \tag{4.23}$$

We can replace the two expressions in 4.19 and get:

$$m \leq \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{\mathbb{E}[\tau(i)]P(N = n)}{P(a_i \text{ declared suboptimal })} + \sum_{n=1}^{\infty} \frac{\mathbb{E}[\tau(n)]P(N = n)}{P(a_n \text{ declared optimal })} \tag{4.24}$$

Part 2. In the second part of the proof, we will upper bound the first term in the summation 4.59 $T_1 = \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{\mathbb{E}[\tau(i)]P(N=n)}{P(a_i \text{ declared suboptimal })}$.

First, since the events "a_i optimal" and "a_i suboptimal" partition the space of samples, we can apply the law of total expectation to $\mathbb{E}[\tau(i)]$ and get:

$$\mathbb{E}[\tau(i)] = P(a_i \text{ is optimal})\mathbb{E}[\tau^{opt}(i)] + P(a_i \text{ is suboptimal})\mathbb{E}[\tau^{subopt}(i)] \tag{4.25}$$

$$= \rho(2k(i) + \frac{2k(i)}{E(i)} + \frac{1}{(1-2p)E(i)}) + (1-\rho)\frac{1}{1-2p} \tag{4.26}$$

where the second inequality is determined by applying lemmas 4.6.5 and 4.6.4.

We now bound $P(a_i \text{ declared suboptimal }) \geq P(R_i^{subopt})P(a_i \text{ is suboptimal}) \geq (1-p)(1-\rho)$ where the last inequality holds because $P(R_i^{subopt}) \geq 1-p$ since the probability of a suboptimal arm being rejected is at least the probability that it is rejected in the first trial which is $1-p$. And since $1-p \geq 0.5$ and $1-\rho \geq 0.5$ we get that $P(a_i \text{ declared suboptimal }) \geq \frac{1}{4}$.

So the formula for T_1 now becomes:

$$T_1 \leq \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} 8\rho k(i)P(N = n) + \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{8\rho k(i)P(N = n)}{E(i)} + \tag{4.27}$$

$$\sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{4\rho P(N = n)}{(1-2p)E(i)} + \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{4(1-\rho)P(N = n)}{1-2p} \tag{4.28}$$

Let's consider the four terms in the summation in turn and upper bound them separately.

Part 2.1 Let's first upper bound the fourth term in the summation:

$$T_{1.4} = \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{4(1-\rho)P(N=n)}{1-2p} \tag{4.29}$$

$$= \frac{4(1-\rho)}{1-2p} \sum_{n=1}^{\infty} (n-1)P(N=n) \tag{4.30}$$

$$\leq \frac{4(1-\rho)}{1-2p} \mathbb{E}[N] \tag{4.31}$$

$$\leq \frac{4(1-\rho)}{\rho(1-2p)^2} \text{ (from lemma 4.6.6)} \tag{4.32}$$

Part 2.2 The third term is (by replacing the formula for $E(i)$:

$$T_{1.3} = \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{4\rho P(N=n)}{(1-2p)E(i)} \tag{4.33}$$

$$= \frac{2p\delta\rho}{(1-2p)^2} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{P(N=n)}{i^2} \tag{4.34}$$

$$\leq \frac{4p\delta\rho}{(1-2p)^2} \tag{4.35}$$

since $\sum_{i=1}^{n-1} \frac{1}{i^2} < 2$ and $\sum_{n=1}^{\infty} P(N=n) = 1$.

Part 2.3 By replacing the formulas for $k(i)$ and $E(i)$, the second term becomes:

$$T_{1.2} = \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{8\rho k(i)P(N=n)}{E(i)} \tag{4.36}$$

$$= \frac{1}{\log\beta} \frac{4p\delta\rho}{1-2p} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{P(N=n)\log(\frac{2i^2(1-2p)}{p\delta}+1)}{i^2} \tag{4.37}$$

$$\leq \frac{4p\delta\rho}{(1-2p)^2} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{P(N=n)\log(\frac{4i^2(1-2p)}{p\delta})}{i^2} \tag{4.38}$$

$$\leq \frac{4p\delta\rho}{(1-2p)^2} \log\frac{4(1-2p)}{p\delta} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{P(N=n)}{i^2} + \frac{8p\delta\rho}{(1-2p)^2} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} \frac{P(N=n)\log i}{i^2} \tag{4.39}$$

$$\leq \frac{8p\delta\rho}{(1-2p)^2} \log\frac{40}{\delta} + \frac{24p\delta\rho}{(1-2p)^2} \tag{4.40}$$

where 4.38 follows from $\frac{1}{\log\beta} \leq \frac{1}{1-2p}$, and 4.40 from (1). $p \geq 0.1$, (2). $\sum_{n=1}^{\infty} P(N = n) = 1$, (3). the fact that $\sum_{i=1}^{n-1} \frac{1}{i^2} < 2$ and (4). $\sum_{i=1}^{n-1} \frac{\log i}{i^2} < 3$. To prove the last relation, one can observe that $\sum_{i=1}^{n} \frac{\log i}{i^2} \leq \sum_{i=1}^{\infty} \frac{\log i}{i^2} \leq \sum_{i=1}^{\infty} \frac{1}{i^{1.5}} \leq 3$, since $\frac{i^2}{\log i} = i^{2-o(1)} \geq i^{1.5}$.

Part 2.4 Finally, the first term can be bounded as:

$$T_{1.1} = \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} 8\rho k(i) P(N = n) \tag{4.41}$$

$$\leq \frac{8\rho}{1-2p} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} P(N = n) \log \frac{4i^2(1-2p)}{p\delta} \tag{4.42}$$

$$\leq \frac{8\rho\mathbb{E}[N]}{1-2p} \log \frac{4}{p\delta} + \frac{16\rho}{1-2p} \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} P(N = n) \log i + \frac{8\rho\mathbb{E}[N]}{1-2p} \log (1-2p) \tag{4.43}$$

But $\sum_{n=1}^{\infty} \sum_{i=1}^{n-1} P(N = n) \log i \leq \sum_{n=1}^{\infty} \sum_{i=1}^{n-1} P(N = n) \log n \leq \sum_{n=1}^{\infty} n \log n P(N = n) = \mathbb{E}[N \log N]$.

Now, similarly to the proof of theorem 4.4.5, we can show that $\mathbb{E}[N \log N] \leq 2\mathbb{E}[N] \log \mathbb{E}[N]$.

We can then replace the formula for $\mathbb{E}[N \log N]$ in 4.43 and using the result that $\mathbb{E}[N] \leq \frac{1}{\rho(1-2p)}$ from lemma 4.6.6 we get:

$$T_{1.1} \leq \frac{8}{(1-2p)^2} \log \frac{40}{\delta} + \frac{16}{(1-2p)^2} \log \frac{1}{\rho(1-2p)} + \frac{16}{(1-2p)^2} \log(1-2p) \tag{4.44}$$

$$= \frac{8}{(1-2p)^2} \log \frac{40}{\delta} + \frac{16}{(1-2p)^2} \log \frac{1}{\rho} \tag{4.45}$$

Now we can finally put together the results from 4.32, 4.35, 4.40 and 4.45 into the inequality for T_1 from 4.28 to get the desired relation that
$T_1 = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log \frac{1}{\rho} + \log \frac{1}{\delta})) = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$.

Part 3. In the third part of the proof we will upper bound the second term from 4.59, $T_2 = \sum_{n=1}^{\infty} \frac{\mathbb{E}[\tau(n)]P(N=n)}{P(a_n \text{ declared optimal})}$. The algebraic manipulations are

similar to Part 2, with the difference being the way $P(a_n \text{ declared optimal})$ is handled.

We will first give a lower bound on $P(a_n \text{ declared optimal})$:

$$P(a_n \text{ declared opt.}) \geq P(a_n \text{ declared opt.}|a_n \text{ is opt.})P(a_n \text{ is opt.}) \tag{4.46}$$

$$= P(S_i^{opt})\rho \tag{4.47}$$

$$\geq (1-2p)\rho \text{ (based on lemma 4.6.3)} \tag{4.48}$$

The second fact we will use is $P(N=n) \leq P(a_n \text{ declared opt.})$. The reason is that the event $\{N=n\}$ is more specific than the event $\{a_n \text{ declared opt.}\}$ since $\{N=n\}$ is a conjunction of events that contains the rejection events of the first $n-1$ arms in addition the event $\{a_n \text{ declared opt.}\}$.

Now, similarly to Part2:

$$\mathbb{E}[\tau(n)] = \rho(2k(n) + \frac{2k(n)}{E(n)} + \frac{1}{(1-2p)E(n)}) + (1-\rho)\frac{1}{1-2p} \tag{4.49}$$

so the formula for T_2 becomes:

$$T_2 = \sum_{n=1}^{\infty} \frac{2\rho k(n) P(N=n)}{P(a_n \text{ declared opt.})} + \sum_{n=1}^{\infty} \frac{2\rho k(n) P(N=n)}{P(a_n \text{ declared opt.})E(n)} + \tag{4.50}$$

$$\sum_{n=1}^{\infty} \frac{\rho P(N=n)}{(1-2p)E(n)P(a_n \text{ declared opt.})} + \sum_{n=1}^{\infty} \frac{(1-\rho)P(N=n)}{(1-2p)P(a_n \text{ declared opt.})} \tag{4.51}$$

First we upper bound the fourth term and use the lower bound on $P(a_n \text{ declared opt.})$: $T_{2.4} \leq \frac{1-\rho}{(1-2p)^2}\sum_{n=1}^{\infty} P(N=n) = \frac{1-\rho}{\rho(1-2p)^2}$.

For the third term, we use the fact that $P(a_n \text{ declared opt.}) \geq P(N=n)$ and get: $T_{2.3} \leq \sum_{n=1}^{\infty} \frac{\rho}{(1-2p)E(n)} \leq \frac{p\delta\rho}{2(1-2p)^2}\sum_{n=1}^{\infty}\frac{1}{n^2} \leq \frac{p\delta\rho}{2(1-2p)^2}$ (since $\sum_{n=1}^{\infty}\frac{1}{n^2} < 2$).

For the second term, we use again $P(a_n \text{ declared opt.}) \geq P(N=n)$ and,

with a similar derivation and arguments as in Part 2.3, we get:

$$T_{2.2} = \frac{p\delta\rho}{2(1-2p)^2}\log\frac{4(1-2p}{p\delta}\sum_{n=1}^{\infty}\frac{1}{n^2} + \frac{p\delta\rho}{(1-2p)^2}\sum_{n=1}^{\infty}\frac{\log n}{n^2} \tag{4.52}$$

$$\leq \frac{p\delta\rho}{(1-2p)^2}\log\frac{40}{\delta} + \frac{3p\delta\rho}{(1-2p)^2} \tag{4.53}$$

For the first term, using similar manipulations to Part 2.4 we get:

$$T_{2.1} = \frac{2\rho}{1-2p}\log\frac{4(1-2p)}{p\delta}\sum_{n=1}^{\infty}\frac{P(N=n)}{P(a_n \text{ declared opt.})} \tag{4.54}$$

$$+ \frac{4\rho}{1-2p}\sum_{n=1}^{\infty}\frac{P(N=n)}{P(a_n \text{ declared opt.})}\log n \tag{4.55}$$

$$\leq \frac{2}{(1-2p)^2}\log\frac{4(1-2p)}{p\delta} + \frac{4}{(1-2p)^2}\sum_{n=1}^{\infty}\log nP(N=n) \tag{4.56}$$

where we used $P(a_n \text{ declared opt.}) \geq \rho(1-2p)$ and $\sum_{n=1}^{\infty}P(N=n) = 1$. The term $\sum_{n=1}^{\infty}\log nP(N=n) = \mathbb{E}[\log N] \leq \log\mathbb{E}[N]$ (using same argument as in Part 2.4) and so:

$$T_{2.1} \leq \frac{4}{(1-2p)^2}\log\frac{4(1-2p)}{p\delta} + \frac{4}{(1-2p)^2}\log\frac{1}{\rho(1-2p)} \tag{4.57}$$

$$\leq \frac{4}{(1-2p)^2}(\log\frac{40}{\delta} + \log\frac{1}{\rho}) \tag{4.58}$$

Putting the results of all subterms together we get that $T_2 = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\rho} + \log\frac{1}{\delta})) = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta}))$ which is the same asymptotic bound as for T_1 and since from 4.59 $m \leq T_1 + T2 = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \log\frac{1}{\delta}))$ as intended.

□

4.6.1 Applications of Greedy Rejection

In this section we will state two interesting consequences of the sample complexity analysis of Greedy Rejection.

The first corollary proves that Greedy Rejection applied to a PAC Bandit model is asymptotically optimal. This is a relatively surprising result as the

only known algorithm (a variant of Median Elimination [Even-Dar et al., 2002]) with this property (from Mannor et al. [2004]) uses the (implicit) knowledge of n, an information that Greedy Rejection doesn't need.

Corollary 4.6.8. *If Greedy Rejection is applied with a uniform distribution over the n arms of a PAC Bandit(n, p, δ), it will return the optimal arm with probability $\geq 1 - \delta$ and with an expected sample complexity $m = O(\frac{1}{(1-2p)^2}(n + \log \frac{1}{\delta}))$.*

The proof is a simple application of theorem 4.6.1 with the observation that $\rho = \frac{1}{n}$ since only one arm is optimal in the worst case of a PAC Bandit(n, p, δ) model. The asymptotic optimality follows due to the lower bound from Mannor et al. [2004] (theorem 13). The algorithm is numerically better than the alternative which is not surprising given the huge constant in the sample complexity bounds of the Median Elimination strategies.

The second corollary proves that Greedy Rejection applied to a 2-armed bandit problem with parameters p and δ is also asymptotically optimal.

Corollary 4.6.9. *If Greedy Rejection is applied with a uniform distribution over the* 2 *arms of a 2-armed bandit(p, δ) problem it will return the optimal arm with probability $\geq 1 - \delta$ and with expected sample complexity $m = O(\frac{1}{(1-2p)^2} \log \frac{1}{\delta})$.*

The result is a direct application of 4.6.8. It is an interesting consequence since, using the reductions from 4.4.1, it shows that Greedy Rejection can be used to solve a stochastic dilemma as well. An interesting open problem is explaining why, according to numerical experiments, Greedy Rejection outperforms Beat-By-K in a Stochastic Dilemma. The analysis of Greedy Rejection is too loose with respect to the constants to directly show that it indeed dominates Beat-By-K (and given the complexity of the proof, it is probably difficult to improve the constants significantly).

4.7 Greedy Hoeffding Rejection

The Greedy Rejection strategy from the previous section assumes the parameter p is known. As in the stochastic dilemma chapter, the natural question is what can be achieved when the parameter is unknown. The idea of this section is to combine the fast rejection idea with the algorithms from section 2.4 which leads to an alternative strategy to the reductions discussed in section 4.4. The new algorithm (Greedy Hoeffding Rejection) is the first step towards a version of Greedy Rejection that can be used to solve realistic optimization problems. As we will see in this chapter and the following one, this algorithm has both good theoretical properties and good empirical performance.

For simplicity of analysis, we will only present the combination of Greedy Rejection and Hoeffding Rejection (the analysis is very similar if instead of Hoeffding Rejection we would use Lazy Hoeffding Rejection or Confidence Sequences). We will use the same notation as in section 4.6. The algorithm will continue sampling an arm until either 1. $\Delta^t = -1$ in which case it labels the arm "suboptimal" and samples a new one or 2. Hoeffding Rejection declares the arm "optimal" (since the lower bound of the confidence interval around the empirical average grows to be larger than 0.5) and the algorithm stops.

The analysis of the algorithm is similar to the analysis of Greedy Rejection, except we need more general results regarding the expected time to reject a suboptimal arm and the probability to accept an optimal arm.

Theorem 4.7.1. *If the Greedy Hoeffding Rejection algorithm is executed for an infinite bandit model $IPAC(p, \delta, \rho)$ (with $\rho \leq 0.5$ and $p \geq 0.1$), it will return the optimal solution with probability $\geq 1 - \delta$ and it will take $m = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \frac{1}{1-2p} \log \frac{1}{\rho\delta(1-2p)}))$ samples on expectation.*

We will prove the theorem and in the discussion section (4.8) we will compare the properties of the algorithm with the other strategies that solve this problem. The correctness proof is identical to the proof of Greedy Rejection so we will skip it.

The first lemma we will prove sets an identical upper bound on the expected number of samples as lemma 4.6.4. The proof is different though, as we will not to be able compute the desired quantity directly.

Lemma 4.7.2. *If Greedy Hoeffding Rejection is executed for the i-th sampled arm and the arm a_i has an associated distribution* Bernoulli(p)*, then* $\mathbb{E}[\tau^{subopt}(i)] \leq \frac{1}{1-2p}$.

Proof. First let's assume that instead of executing Hoeffding Rejection for a suboptimal arm, we will only stop if and when the random walk hits the barrier at -1. But this is equivalent to executing a biased random walk with a negative bias $1-2p$ until it hits a unique absorbing barrier at -1. Using a similar result to theorem A.2.2(ii), we can compute the expected time of such a random walk which is exactly $\frac{1}{1-2p}$.

But having an extra stopping condition determined by the use of Hoeffding Rejection can only decrease the expected sample complexity, hence the result.

□

The second lemma deals with the probability that an optimal arm is eventually accepted. The result is similar to lemma 4.6.3, except now we have to apply a more general result.

Lemma 4.7.3. *If the i-th sampled arm has a* Bernoulli$(1-p)$ *distribution, the probability that Greedy Hoeffding Rejection will stop and recommend it as optimal is* $P(S_i^{opt}) \geq 1-2p$.

Proof. As in lemma 4.7.2, let's first assume that instead of executing Hoeffding Rejection for an optimal arm, we will only stop if and when the random walk hits the barrier at -1. This is equivalent to studying the probability that a positively biased random walk will ever hit -1. But the Ballot theorem for random walks (Theorem 3 from Addario-Berry and Reed [2008]) states that this last probability is $2p$.

Since the random walk must stay positive for a finite amount of time with probability 1 (since Hoeffding Rejection has finite expected sample complexity), it has to be the case that the probability that the algorithm stops without recommending the arm is optimal is at most the probability that it will stop by hitting -1 on the infinite horizon (which is at most $2p$). So, the complement of that event (which is that Hoeffding Rejection stops and recommends to arm as optimal) is at least $1 - 2p$. □

We will now state and prove the lemma for the expected sample complexity of Greedy Hoeffding Rejection. Since the proof is very similar to the proof of 4.6.7 we will only sketch it.

Lemma 4.7.4. *The expected sample complexity of Greedy Hoeffding Rejection is* $m = O(\frac{1}{(1-2p)^2}(\frac{1}{\rho} + \frac{1}{1-2p}\log\frac{1}{\rho\delta(1-2p)}))$.

Proof sketch. As in the proof of 4.6.7, we get an upper bound for m:

$$m \leq \sum_{n=1}^{\infty}\sum_{i=1}^{n-1}\frac{\mathbb{E}[\tau(i)]P(N=n)}{P(a_i \text{ declared suboptimal })} + \sum_{n=1}^{\infty}\frac{\mathbb{E}[\tau(n)]P(N=n)}{P(a_n \text{ declared optimal })} \quad (4.59)$$

with:

$$\mathbb{E}[\tau(i)] = P(a_i \text{ is optimal})\mathbb{E}[\tau^{opt}(i)] + P(a_i \text{ is suboptimal})\mathbb{E}[\tau^{subopt}(i)] \quad (4.60)$$

$$\leq \rho\frac{1}{(1-2p)^2}\log\frac{i^2}{\delta(1-2p)} + (1-\rho)\frac{1}{1-2p} \quad (4.61)$$

where for the last inequality we upper bound $\mathbb{E}[\tau^{opt}(i)]$ based on applying lemma 2.4.4 adapted for the level of error acceptable for the arm at index i and we upper bound $\mathbb{E}[\tau^{subopt}(i)]$ by using lemma 4.7.2.

Replacing $\mathbb{E}[\tau^{opt}(i)]$ (and similarly $\mathbb{E}[\tau^{opt}(n)]$) in the upper bound for m, the proof continues using similar techniques and tools as for 4.6.7 and the final result follows. We remark that the manipulations are actually easier as the upper bound for the expected time to accept an optimal arm has a simpler formula as compared to the proof for Greedy Rejection. □

4.8 Summary and Discussion

The high level goal of this chapter was to extend the published results from the finite PAC Bandit model to a setting where we have an infinite amount of arms. The key novel challenges as compared to the finite PAC bandit problem were:

1. Design algorithmic strategies that are able to cope with the inability to sample ”all” the arms.

2. Address the (potential) lack of knowledge of the concentration of optimal arms.

The first challenge can not be solved by a simple reduction to the finite version of the problem. The reason is that all of the algorithms introduced in the finite PAC Bandit setting (Naive, Successive Elimination, Median Elimination [Even-Dar et al., 2002] or their variants from Mannor et al. [2004]) are assuming all arms can be sampled at any time—or, equivalently, that the optimal arm is always part of an ”active” set of arms that the algorithm can access. This fact is no longer true in the infinite version of the problem. The challenge

Model	p known	Reference
2PAC(p,δ)	$O(\frac{1}{(1-2p)^2}\log\frac{1}{\delta})$ $\Omega(\frac{1}{(1-2p)^2}\log\frac{1}{\delta})$	*[1], 2.3.2,* *2.3.3* *[2]*
nPAC(n,p,δ)	$O(\frac{1}{(1-2p)^2}(n+\log\frac{1}{\delta}))$ $\Omega(\frac{1}{(1-2p)^2}(n+\log\frac{1}{\delta}))$	*[2], 4.6.8* *[2]*
IPAC(ρ,p,δ)	$O(\frac{1}{(1-2p)^2}(\frac{1}{\rho}+\log\frac{1}{\delta}))$ $\Omega(\frac{1}{(1-2p)^2}(\frac{1}{\rho}+\log\frac{1}{\delta}))$	4.6.1 4.5.1
	p unknown	
2PAC(p,δ)	$O(\frac{1}{(1-2p)^2}(\log\frac{1}{\delta}+\log\log\frac{1}{1-2p}))$ $\Omega(\frac{1}{(1-2p)^2}\log\frac{1}{\delta})$	*[1], 2.4.5* *[2]*
nPAC(n,p,δ)	$O(\frac{n}{(1-2p)^2}(\log\frac{n}{\delta}+\log\log\frac{1}{1-2p}))$ $\Omega(\frac{1}{(1-2p)^2}(n+\log\frac{1}{\delta}))$	*[1], 4.4.5(iii)* *[2]*
IPAC(ρ,p,δ)	$O(\frac{1}{\rho(1-2p)^2}(\log\frac{1}{\rho\delta}+\log\log\frac{1}{1-2p}))$ $O(\frac{1}{(1-2p)^2}(\frac{1}{\rho}+\frac{1}{1-2p}\log\frac{1}{\rho\delta}+\frac{1}{1-2p}\log\log\frac{1}{1-2p}))$ $\Omega(\frac{1}{(1-2p)^2}(\frac{1}{\rho}+\log\frac{1}{\delta}))$	*4.4.5(iii)* *4.7.1* *4.5.1*

Table 4.1: Summary of the expected sample complexity bounds for solving various PAC Bandit models. Color code: tight bounds (green background), gap between upper and lower bounds (red background), new result from this thesis (blue font), previous result (black font). References: [1] is Even-Dar et al. [2002] and [2] is Mannor et al. [2004]

is addressed by designing incremental strategies that sample arms from the distribution over the space of arms and label them as optimal or suboptimal. This new approach requires a different type of analysis, although of a similar type (with the exception of the proof for the Greedy Rejection algorithm, which is of a different nature).

The second challenge makes the problem intrinsically harder as compared to the finite setting where the number of arms is considered implicitly known. It is thus remarkable it is actually possible to get very similar results for p known in the infinite setting as in the finite setting (see table 4.1). We remark

though that the strategy with this property (Greedy Rejection) is very different from the algorithm from the finite setting, which is a variant of Median Elimination (see Even-Dar et al. [2002]). The Median Elimination algorithm, while asymptotically optimal in the known p setting, is impractical due to its huge constant in the sample complexity bound. It is not clear whether Median Elimination can be modified for the infinite setting in a manner that would avoid adding complexity parameters to its bound (a doubling trick for example would add a logarithmic factor, which would render it suboptimal).

An interesting property of Greedy Rejection is that it suggests an algorithmic idea (fast rejection of seemingly suboptimal arms) that can be used in the harder, adaptive setting (where the parameter p is unknown) and which, as we will see in the next chapter, has excellent empirical properties. A second interesting property is that Greedy Rejection doesn't make use of (possible) knowledge about ρ. This implies that Greedy Rejection is asymptotically optimal in the p known, ρ unknown setting (given that we have a matching lower bound from theorem 4.5.1 for the easier problem of p known, ρ known).

As it is obvious from the summary of the results in table 4.1, while the PAC Bandit problems are essentially completely solved asymptotically in the setting where p is known, for the harder problem (with p unknown) there are still gaps between the upper and lower bounds. We think the key difficulty lies in proving a tight lower bound in the simplest possible setting - the 2 Armed Bandit problem. Since we've shown in section 4.4.1 that the SD(p, δ) and 2PAC(p, δ) problems are essentially identical, we also conjecture as in section 2.4.4 that the correct dependency on p in the lower bound for 2PAC(p, δ) has an extra $\log \log \frac{1}{1-2p}$ multiplicative factor.

We also observe that because we don't have an ϵ parameter as in the standard finite PAC Bandit problem definition, it is relatively difficult to translate the lower bounds from Mannor et al. [2004] to our unknown p problems. For this reason, we chose to list in table 4.1 (for the case of unknown p) the lower bounds from the easier p known case. We leave the tightening of these bounds as open problems.

One of the most interesting applications from this chapter is the Greedy Hoeffding Rejection algorithm (see the last rows in table 4.1). The algorithm is suboptimal in general—it is dominated by the Iterative Lazy Hoeffding Rejection strategy (4.4.5) when for example $\frac{1}{\rho} = \Theta(\sqrt{\frac{1}{1-2p}})$ or more generally when $\frac{1}{\rho} = o(\frac{1}{1-2p})$. But there is an interesting class of domains for which it is actually almost tight with the (possibly loose in the general case) lower bound: when $\frac{1}{\rho} = \Omega(\frac{1}{(1-2p)^2})$ or more generally when $\frac{1}{\rho} = \omega(\frac{1}{1-2p}\log\frac{1}{1-2p})$. The interpretation is that when the concentration of optimal rewards is significantly smaller than the difference between the bias of the optimal and suboptimal arms, the Greedy Hoeffding Rejection algorithm will be guaranteed to perform very well. We will see in the following chapter how an extension of this algorithm has indeed excellent empirical performance.

Chapter 5
Planning in Reward-Rich Domains via Infinite Bandits

5.1 Introduction

In this chapter we extend the infinite bandit model from chapter 4 with the goal of solving stochastic planning problems. We view planning as an optimization problem, with every possible plan being an 'arm' in a finite or infinite bandit model. We target a class of problems where standard local search approaches fail: in particular, relations between arms are either not predictive of relations between their associated reward values or such relations can be ignored without sacrificing much in terms of the number of evaluations needed to get an approximately optimal solution.

Our claim is that some apparently hard planning problems can be solved via a sampling and testing approach that cannot be solved by algorithms that depend on the existence of local structure for search. We documented this phenomenon in the video games *Infinite Mario* and *Pitfall!* where policies can be very similar but have vastly different outcomes (one different action in a long sequence can lead the agent to failure as opposed to successfully completing a level).

The model we discuss in this chapter is a variation of the PAC-Bandit model [Even-Dar et al., 2002] to an infinite number of arms. The connection to the

PAC-Bandit model was discussed at length in chapter 4. We extend the model from 4 by: **(1)** allowing a more general threshold between optimal and sub-optimal arms (r_0 instead of 0.5), **(2)** adding an approximation parameter ϵ, **(3)** having a more general support for the expected reward of the arms and a more general distribution over this support. By allowing the agent to aspire to any reward level, our definition of the performance measure is akin to the earlier work of Wang et al. [2008], but we target offline rather than online stochastic optimization.

5.2 Model

We define an arm a as a Bernoulli(r_a) probability distribution, for some $r_a \in [0, 1]$. When an arm is *pulled*, it returns a reward value sampled from Bernoulli(r_a). An arm a_i is preferred to another a_j if it has a higher expected reward value, $r_{a_i} > r_{a_j}$. Arms are sampled from a distribution $\mathcal{P}$ over an *arm space* $S \subseteq [0, 1]$, possibly infinitely large (countable or uncountable). The distribution $\mathcal{P}$ defines an infinite-armed bandit problem.

We seek algorithms that take a reward level r_0 as input and attempt to minimize the number of pulls needed to identify an arm with expected value of r_0 or more. This *sample complexity* has a dependence on $\mathcal{P}$ and r_0, as it may be likely or unlikely to encounter an arm with high enough reward. Specifically, define $\rho = P_{a \sim \mathcal{P}}(r_a \geq r_0)$ as the probability of sampling a "good enough" arm. We assume the domain is "reward rich"—specifically, that ρ is bounded away from zero.

Formally, we define an (ϵ, δ, r_0)-correct algorithm ALG for an an infinite bandit problem (that we will label IB$(\epsilon, \delta, r_0, \mathcal{P})$) to be an algorithm that after

a number of samples $T(\epsilon, \delta, r_0, \mathcal{P})$ (that is finite with probability 1) returns an arm a with expected value $r_a \geq r_0 - \epsilon$ with probability at least $1 - \delta$.

5.2.1 Distribution $\mathcal{P}$

Technically, distribution $\mathcal{P}$ plays a similar role as in the Label Identification problem: it acts as a generator for objects that in this case are bandit arms. Contrary to Label Identification though, we associate other interpretations to $\mathcal{P}$.

The first perspective is to view $\mathcal{P}$ as the (unknown) distribution over (expected) function values induced by doing random search over some input space.

In more details, let's assume we are given query access to a function $f : X \rightarrow Y$. Every time f is evaluated at a point $x \in X$ it returns $f(x)$. The standard approach for maximizing f is to make some parametric assumptions about f (e.g. f is linear, convex, etc.) and design algorithms that take advantage of these assumptions. In some case though, f lacks structure (e.g. there is no correlation between the similarity of inputs and their values), or we don't have enough information or we are simply not willing to make any assumptions about f. In such scenarios, random search is used as a first attempt to optimize the function, sometimes performing remarkably well (as we will also demonstrate in section 5.3).

Random search samples inputs uniformly from X, evaluates them, and stops as soon as a stopping criterion is fulfilled (e.g. a desired performance threshold is reached or some maximum budget of queries is spent or the maximum value in Y is obtained). Sampling uniformly (or from another distribution for that matter) from X is equivalent to sampling from an unknown

distribution over Y determined by the input function values $f(x)$. Distribution $\mathcal{P}$ can be seen as playing the role of this unknown distribution.

From another perspective, $\mathcal{P}$ is akin to distributions maintained by some optimization algorithms as components in their search process. Algorithms like the Cross Entropy Method or Genetic Algorithms for example (see chapter 6 for more details) proceed in epochs by updating a distribution over the input space at the end of each iteration with the goal of increasingly focusing the distribution on inputs that are close to optimal. During an epoch, the distribution is fixed and a set of samples is taken from it and then evaluated with the high level goal of finding the best inputs among the samples. When the evaluations are noisy, a fixed epoch is thus very similar to solving an infinite bandit problem of the type introduced in this chapter (with distribution $\mathcal{P}$ representing, as in the first perspective from above, the distribution over (expected) function values induced by the current distribution over the input space).

5.2.2 Parameter r_0

Whenever the input space X of a function f to be optimized is infinite (countable or uncountable), unless one makes some type of assumption about f, there will exist problems that are unsolvable. Consider for example a "needle in the haystack" scenario where f is maximum at a unique input $x^* \in X$, but there is no structure in f to discover x^* other than by random sampling. In this case the probability of finding x^* is 0 if X is infinite, thus rendering the optimization problem unsolvable.

The role of parameter r_0 is to define a particular optimization problem: it acts as a threshold between optimal and suboptimal values. Knowledge of

reasonable r_0 values is problem dependent and its range of values defines a family of related optimization problems (increasing r_0 from the minimal to the maximal reward naturally increases the difficulty of the problem).

To have a well defined optimization problem, our key assumption ("reward richness") is that $\rho = P_{a\sim\mathcal{P}}(r_a \geq r_0) > 0$—i.e. we assume that the probability mass on optimal values is bounded away from 0 (thus preventing the "needle in the haystack" scenario described above). The assumption is relatively mild (as it applies to a wide variety of optimization problems, including some with no structure in the input space for example) and, as we will see in this chapter, it can be exploited to solve non-trivial optimization problems.

5.2.3 Parameter ϵ

The role of the parameter ϵ is similar to the role played in the Label Identification problem: to establish an approximate stochastic optimization problem where an algorithm is allowed to avoid spending an inordinate amount of samples for separating two arms that are very close (see section 3.2.1 for a similar perspective).

To ground the concept in the context of multi-armed bandits, let's assume that we are given 3 arms with rewards 0, 0.5 and $0.5 + \Delta$ for some $\Delta \in (0, 0.5)$. Separating the last two arms requires $\Omega(\frac{1}{\Delta^2})$ samples which becomes impractical for very small Δ values, whereas separating the arms with expected rewards ≥ 0.5 from the 0 reward arm can be done in a few samples with high probability. In a lot of practical settings, for small Δ values, the last two arms can be considered essentially equivalent, so there is no point in spending effort deciding which one is better. The parameter ϵ thus encodes what is the minimal relevant difference between arms.

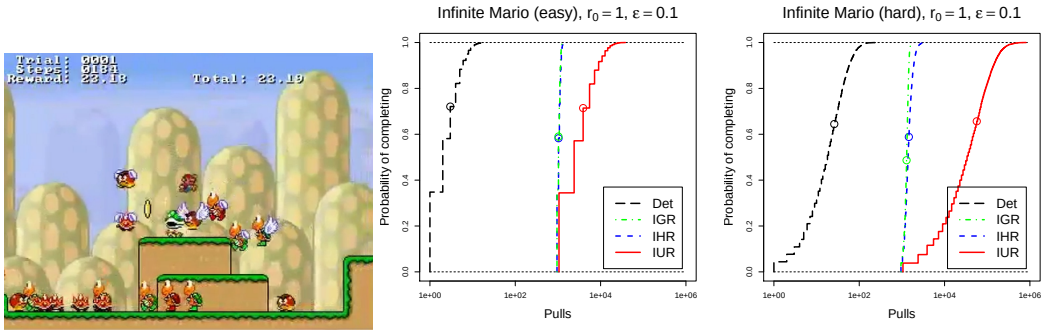

Figure 5.1: A screenshot of *Infinite Mario* and plots of the distribution of the sample complexity for an algorithm that pulls each arm once (x-axis log-scale). The distributions are plotted for 2 of the 50 *Infinite Mario* levels corresponding to the first (easy) and third (hard) quartiles. See the section 5.4 for more details.

5.3 Illustration - Infinite Mario

Our first experiment used a version of *Infinite Mario* (a clone of the *Super Mario* video game, see the left panel of Figure 5.1) that was modified for the Reinforcement Learning Competition [Whiteson et al., 2010]. It was also used for other competitions [Togelius et al., 2010] and it is considered to be an interesting benchmark for planning and learning. The game is deterministic and gives us an opportunity to present a natural problem that illustrates the "reward richness" phenomenon motivating our work.

We treated starting screens in *Infinite Mario* as bandits, where each arm encodes an action sequence 50-steps long. In the experiments, the agent's goal was to reach a threshold on the right side of the initial screen. The action set of the agent was restricted by removing the backward action (unnecessary for solving any level), resulting in 8 total actions and an arm space of size 8^{50}. Action sequences were tested in the actual game, assigning rewards of -1 if the agent was destroyed, 0 if it did not reach the goal in 50 steps, and a value of $100 - t$, otherwise (where t was the number of steps taken before reaching

the goal). Since the domain is deterministic, the agent simply sampled uniformly at random new arms until one was found with reward greater than 0. Sampling uniformly from the space of arms induces an unknown distribution $\mathcal{P}$ over the space of possible rewards, which is the distribution over arms described in section 5.2.1.

The average number of pulls needed to find a strategy for completing the first screen over a set of 50 levels ranged from 1 to 1000, with a median of 7.7 pulls and a mean of 55.7 (due to a few very difficult levels). Thus, testing just a handful of randomly generated action sequences was sufficient to find a successful trajectory in this game. The performance of the method is conveyed by the black (leftmost) lines in the plots in Figure 5.1 (the algorithms corresponding to the other curves are described in the next sections). The results show that nearly all screens were solved in well under 100 samples.

As an extension to this experiment, we "chained" trajectories together to completely solve each of the 50 levels (as opposed to just treating the starting screens). Using a cap of 3000 pulls for each screen in a level, this simple method was able to complete 40 out of the 50 levels. We recorded several videos of the performance of this simple strategy:

- Success video (resulting from stitching together solutions to consecutive screens), 0:53: http://www.youtube.com/watch?v=tH5DRNrRS8I
- Partial success video, 0:49: http://www.youtube.com/watch?v=Wh8HGKIp7PQ
- First screens for 50 levels, 9:51: http://www.youtube.com/watch?v=tcJSQcVzRkc

Regarding the implementation, we took the *Infinite Mario* code from RL-Competition source code: http://code.google.com/p/rl-competition/ and RL-Glue code from RL-Glue source:
http://glue.rl-community.org/wiki/Main_Page.

5.4 Algorithms

When the sampled arms are not deterministic, the problem of allocating pulls is more complex. The agent is faced with a choice between getting better accuracy estimates of previously sampled arms versus sampling new arms to find one with a higher value. In the following, we state and prove a lower bound on the expected sample complexity of a correct algorithm.

Theorem 5.4.1. *Any (ϵ, δ, r_0)-correct algorithm for an $IB(\epsilon, \delta, r_0, \mathcal{P})$ problem has an expected sample complexity of at least $T(\epsilon, \delta, r_0, \mathcal{P}) = \Omega(\frac{1}{\epsilon^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$.*

Most of the proofs from this chapter use very similar analysis ideas as the proofs from chapter 4 and thus we will leave them for Appendix B. The proof of theorem 5.4.1 can be found in B.1.

For the (general) case, when the concentration of rewards ρ is unknown, this section introduces three types of algorithms: one that is an incremental version of a naïve strategy [Even-Dar et al., 2002], one inspired by the Hoeffding Races framework [Maron and Moore, 1997; Heidrich-Meisner and Igel, 2009], and another that uses ideas from ballot-style theorems for random walks [Addario-Berry and Reed, 2008] to quickly reject unpromising arms.

These strategies are extensions of the algorithms we introduced in chapter 4 and we will discuss this connection in detail. The extensions are analogous to the modifications we made to the Stochastic Dilemma algorithms (chapter 2)

to solve the more general Label Identification problem (chapter 3).

All the algorithms we introduce have the structure of the **Generic Algorithm** (Algorithm 1): They sample an arm, make a bounded number of pulls for the arm, check if the arm should be accepted (and in this case, stop and return the arm) or rejected (sample a new arm from $\mathcal{P}$ and repeat). The decision rule for acceptance / rejection and when it can be applied is what differentiates the algorithms.

Algorithm 1: GenericAlgorithm $(\epsilon, \delta, r_0, RejectionFunction)$:

- Let $i = 1, found = FALSE$.
- While $found == FALSE$:
  1. Sample a new arm $a_i \sim \mathcal{P}$.
  2. $decision = RejectionFunction(a_i, i, \epsilon, \delta, r_0)$.
  3. If $decision == ACCEPT, found = TRUE, a_{found} = a_i$.
  4. If $decision == REJECT, i = i + 1$, continue.
- Return a_{found}.

5.4.1 Iterative Uniform Rejection (IUR)

Iterative Uniform Rejection (Algorithm 2) is an incremental version of the naïve strategy by Even-Dar et al. [2002]. The algorithm pulls an arm a fixed number of times to decide with high confidence if the arm has an expected reward less than or greater than $r_0 - \epsilon$. It samples arms in this manner until one with an estimated mean reward of at least $r_0 - \epsilon$ is found.

Algorithm 2: IterativeUniformRejection (ϵ, δ, r_0):

- Return $\mathit{GenericAlgorithm}(\epsilon, \delta, r_0, \mathit{UniformRejection})$
- Function $\mathit{UniformRejection}(a, i, \epsilon, \delta, r_0)$
  1. Let $n_0 = \frac{4}{\epsilon^2} \ln \frac{2i^2}{\delta}$.
  2. Pull the arm a_i n_0 times to get rewards $r_k \sim \text{Bernoulli}(r_{a_i}), k \in [n_0]$.
  3. Let $\hat{r}_{a_i} = \frac{1}{n_0} \sum_{k=1}^{n_0} r_k$.
  4. If $\hat{r}_{a_i} < r_0 - \frac{\epsilon}{2}$ return *REJECT*.
  5. Return *ACCEPT*.

The algorithm extends the Iterative Majority Vote algorithm from section 4.4.3 (see theorem 4.4.5) to handle arbitrary thresholds ($r_0 - \frac{\epsilon}{2}$ instead of 0.5) and for situations when the parameter p of a particular arm is unknown. We note that the extension is similar to the way the Majority Vote algorithm (section 2.3.1) is extended by Naive Majority Vote (section 3.4.1). In the following theorem we state the formal guarantees for IUR:

Theorem 5.4.2. Iterative Uniform Rejection *is an* (ϵ, δ, r_0)*-correct algorithm for any* $IB(\epsilon, \delta, r_0, \mathcal{P})$ *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\rho\delta})$.

The **IUR** algorithm is simple, correct, and achieves a bound close to the lower bound for the problem. We leave the proof of the theorem for Appendix B.2.

5.4.2 Iterative Hoeffding Rejection (IHR)

One problem with **IUR** is that it is very conservative in the sense of taking a large number of samples for each arm (with the dominant term being $\frac{1}{\epsilon^2}$). The algorithm does not take advantage of the fact that it may be possible to tell that an arm is highly unlikely to be better than $r_0 - \epsilon$ long before all n_0 pulls are performed. As a result, the algorithm wastes pulls deciding precisely *how* good or bad the arm is, when it just needs to know *whether* it is good or bad. **Iterative Hoeffding Rejection** (Algorithm 3) exploits the situation in which Δ_i, the difference between the expected reward of arm a_i and r_0, might be larger than ϵ, so an unpromising arm could be rejected before reaching the decision threshold from **IUR**—an insight from the Hoeffding Races framework [Maron and Moore, 1997]. The main idea of the **IHR** algorithm is to maintain confidence intervals built using the Hoeffding bound around the empirical average for the sampled arm and to reject the arm as soon as the upper bound of the confidence interval drops below a certain threshold. If this threshold is not reached after a particular number of pulls, the arm is accepted.

As the name suggests, the algorithm is a variant of the strategy with the same name from section 4.4.3 extended to handle arbitrary thresholds and with a predefined maximum number of samples (n_0) that can be taken for a fixed arm (which is analogous to the strategy Stopped Hoeffding Rejection from section 3.4.2).

Algorithm 3: IterativeHoeffdingRejection (ϵ, δ, r_0):

- Return *GenericAlgorithm*$(\epsilon, \delta, r_0, \textit{HoeffdingRejection})$
- Function HoeffdingRejection$((a, i, \epsilon, \delta, r_0)$
  1. Let $j = 1, \delta_0 = \frac{\delta}{2i^2}, n_0 = \frac{4}{\epsilon^2} \ln \frac{1}{\epsilon\delta_0}$.
  2. For $j = 1, 2, ..., n_0$
     (a) $r_j \sim \text{Bernoulli}(r_{a_i}), \hat{r}_{a_i,j} = \frac{1}{j} \sum_{k=1}^{j} r_k$
     (b) If $\hat{r}_{a_i,j} < r_0 - \sqrt{\frac{2\log(2j^2/\delta_0)}{j}}$ return *REJECT*.
     (c) $j = j + 1$
  3. Return *ACCEPT*.

We also note that in a similar manner that IHR extends the Hoeffding Rejection algorithm from the Stochastic Dilemma setting, we can define iterative versions for Lazy Hoeffding Rejection and Confidence Sequences. These algorithms (**Iterative Lazy Hoeffding Rejection** and **Iterative Confidence Sequences**) are extended in an identical manner as the algorithm described in this section so we will only focus on the intuition and analysis of IHR.

Theorem 5.4.3. Iterative Hoeffding Rejection *is an* (ϵ, δ, r_0)*-correct algorithm for any* $IB(\epsilon, \delta, r_0, \mathcal{P})$ *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\epsilon\rho\delta})$.

The proof of the theorem can be found in Appendix B.3. The analysis of the algorithm is tight in the worst case (consider the domain used to prove the lower bound in Theorem 5.4.1). Nevertheless, as we will show in the experiments section, the algorithm has a much better practical behavior than **IUR**. The reason can be understood by emphasizing the differences between the arms in the upper bound for **IHR**. Define $\Delta_a = r_a - r_0$ to be a random variable

that encodes the difference between r_0 and the expectation of an arm sampled from $\mathcal{P}$, and define Δ_- such that $\frac{1}{\Delta_-^2} = \mathbb{E}[\frac{1}{\max(\epsilon^2, \Delta_a^2)} | \Delta_a < 0]$. ($\Delta_-$ is lower bounded by ϵ and it encodes the relevant difference for rejecting an arm if the arm has an expected value smaller than r_0.) It can then be shown (see the proof in Appendix B.4) that:

Theorem 5.4.4. *The expected sample complexity of* **Iterative Hoeffding Rejection** *is upper bounded by* $O((\frac{1}{\epsilon^2} + \frac{1}{\rho\Delta_-^2}) \log \frac{1}{\epsilon\rho\delta})$ *with probability at least* $1 - \delta$.

For situations where Δ_- is larger than ϵ, the number of pulls needed to classify a 'bad' arm is actually much smaller (ignoring log factors, the difference is between $O(\frac{1}{\Delta_-^2})$ and $O(\frac{1}{\epsilon^2})$ pulls per arm). This difference is the reason why the algorithm has the potential to be much more useful then **IUR** in practice.

The algorithm can be improved by accepting an arm faster if the lower bound of the confidence interval for the empirical average of a particular arm becomes larger than r_0. Another immediate extension is to use Bernstein bounds [Mnih et al., 2008b] instead of Hoeffding bounds to take advantage of the case where the distributions associated with each arm have low variance. We leave these straightforward improvements for future work.

5.4.3 Greedy Hoeffding Rejection (GHR)

Both **IUR** and **IHR** carefully decide whether an arm is good or bad before deciding to reject. As a consequence, with high probability, the first time they encounter a 'good' arm, they accept it. This strategy is reasonable in general, but it has one disadvantage: when the proportion of good arms is relatively low, these algorithms will spend a long time sampling and discarding bad arms. In some cases, a better strategy could be to reject faster—without being sure

with high probability whether an arm is good or bad. This approach is permissible in our framework since failure to accept a good arm is only penalized in terms of sample complexity and does not compromise correctness. Related examples of empirically successful algorithms that quickly reject are Biased Robin [Madani et al., 2003] in the Budgeted Bandit setting and evolutionary algorithms in noisy settings [Fitzpatrick and Grefenstette, 1988].

We next describe an algorithm that implements such a strategy by constraining the empirical average of each sampled arm to stay above a certain threshold for it not to be rejected. The algorithm is a variant of the Greedy Hoeffding Rejection strategy introduced in section 4.7 modified to handle arbitrary thresholds between optimal and suboptimal arms (beyond 0.5) and with a maximum budget of samples for accepting or rejecting a particular arm.

Algorithm 4: GreedyHoeffdingRejection (ϵ, δ, r_0):

- Return $\mathit{GenericAlgorithm}(\epsilon, \delta, r_0, \mathit{GreedyRejection})$
- Function GreedyRejection $(a, i, \epsilon, \delta, r_0)$
    1. The function is identical to HoeffdingRejection with the exception of Line 2*b* which is replaced by: "If $\hat{r}_{a_i,j} < r_0 - \frac{\epsilon}{2}$ return *REJECT*".

Algorithm 4 is essentially a mix of two stopping rules: (1) Reject an arm as soon as the empirical average drops below $r_0 - \epsilon$, and (2) Accept an arm after n_0 samples.

Similarly to the observation we had about IHR, we can define "greedy" versions of the Lazy Hoeffding Rejection and Confidence Sequences strategies (**Greedy Lazy Hoeffding Rejection** and **Greedy Confidence Sequences**) that

quickly reject seemingly suboptimal arms but apply other strategies than Hoeffding Rejection for the definition of the dynamic confidence intervals around the empirical average. As in the previous section, we focus on Hoeffding Rejection but the analysis and results apply to these other strategies as well.

Theorem 5.4.5. Greedy Hoeffding Rejection *is an* (ϵ, δ, r_0)*-correct algorithm for any* $IB(\epsilon, \delta, r_0, \mathcal{P})$ *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^3} \log \frac{1}{\epsilon\rho\delta})$*, if* $r_0 > \epsilon$ *(if* $r_0 \leq \epsilon$*, an algorithm can simply return the very first arm, which is guaranteed to be 'good' because its reward r satisfies* $r \geq 0 \geq r_0 - \epsilon$*.)*

The key idea of the proof is to interpret the evolution of the empirical average for a good arm as a random walk and then apply a ballot-style theorem [Addario-Berry and Reed, 2008] to bound the probability that the average will always be higher than a fixed threshold. Doing so allows us to lower bound the probability of accepting a good arm, which is the key to upper bounding the expected sample complexity.

The proof can be found in Appendix B.5. The structure of the proof is similar to that of theorem 4.7.1 with the key difference being that we use a more general result about the properties of random walks (with steps being real numbers as opposed to just 1 or -1). While the difference is technical, it leads to a more involved analysis.

While the worst-case bound we prove is a factor of $O(\frac{1}{\epsilon})$ worse than that of the other algorithms, it can be tightened in a similar manner to the proof of theorem 4.7.1.

5.5 Experimental Results

Originally developed for the Atari 2600, *Pitfall!* is a game where the objective is to guide the protagonist through the jungle collecting treasure while avoiding items that harm him. In our experiments, interaction with the game was done via an emulator [JStella, 2008], which was modified to allow for software control. In the experiment, the agent's goal was defined simply as arriving at the right side of the screen on the top tier (some levels can be finished on the lower tier). The evaluation used the same 8 actions from the *Infinite Mario* experiment. Stochasticity was added by randomly changing the joystick input to a centered joystick with no button press 5% of the time.

In this game, constructing a policy as a mapping from states to actions is difficult because it is unclear exactly what representation to use. The Atari 2600 has 128 bytes of RAM, which means the actual size of the state space can be 8^{128}, much too large to effectively plan in directly. Treating the game as a collection of objects greatly simplifies the problem and has been used in *Pitfall!* for learning the dynamics of the first screen and navigating it successfully [Diuk et al., 2008], but requires prior domain knowledge to define what kind of interactions can occur between objects.

Because the issue of state in *Pitfall!* is problematic, one approach to planning in this domain is to not factor in state at all but to execute action sequences (which are policies) blindly (conditioned only on time, as opposed to state). The search space for sequences of 500 actions is 8^{500} possible plans. However, on average far fewer than 8^4 of the possible sequences actually need to be sampled uniformly at random before a successful one is found. This result is surprising, as the more difficult screens do not tolerate errors of more than a couple of pixels of placement. The success indicates that, like *Infinite Mario*,

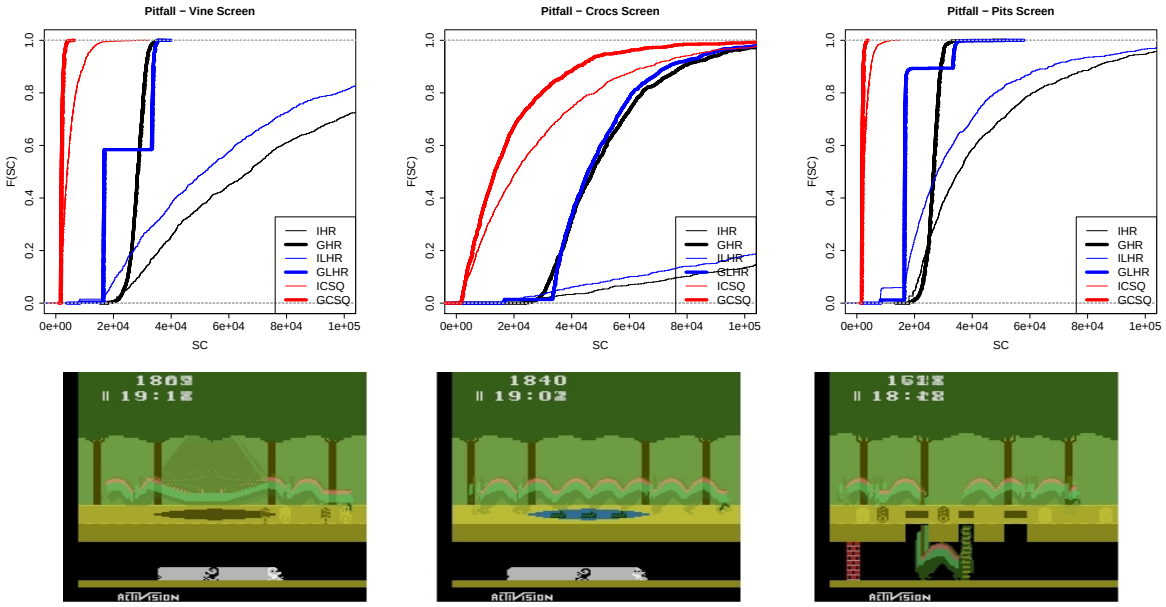

Figure 5.2: Plot of distribution of the sample complexity (pulls needed) of the algorithms over a set of 1000 repetitions. The distributions are plotted for 3 different *Pitfall!* levels (shown along with a representation of a successful policy in the lower half of the figure). All experiments used $\delta = 0.1, \epsilon = 0.1$ and $r_0 = 0.4$ for the screens on the left and right and $r_0 = 0.3$ for the screen in the middle.

Pitfall! is reward rich.

Figure 5.2 illustrates the results of running six algorithms on three *Pitfall!* levels (labeled *Vine*, *Crocs* and *Pits*). Three of the algorithms were introduced in section 5.4.2: Iterative Hoeffding Rejection (IHR), Iterative Lazy Hoeffding Rejection (ILHR), Iterative Confidence Sequences (corresponding to the thick lines in Figure 5.2) and the other three were introduced in section 5.4.3: Greedy Hoeffding Rejection (GHR), Greedy Lazy Hoeffding Rejection (GLHR), Greedy Confidence Sequences (GCSQ) (from section 5.4.3 (thin lines in Figure 5.2).

We plot the empirical cumulative distribution functions for the sample complexities of the algorithms resulting after 1000 repetitions of running each algorithm for a particular Pitfall screen.

For all three problems, the fast-rejection strategies (the algorithms that reject as soon as the empirical average drops below a threshold, i.e. GHR, GLHR, GCSQ) dominate by a large margin the corresponding iterative strategies (the algorithms that classify each arm with high probability, irrespective of whether it is "good" or "bad", i.e. IHR, ILHR, ICSQ). The same pattern can also be seen in Figure 5.1, where the algorithms were used on a deterministic domain without modification. Note that the deterministic strategy used for *Infinite Mario* is not successful in *Pitfall!* because of the noise we introduced. The deterministic algorithm assumes an arm is good if it is successful on the first pull, which can be very misleading. In *Crocs*, for example, the deterministic strategy results in over 90% of the runs erroneously returning a bad arm. Another interesting observation is that the strategies that quickly reject arms have a significantly smaller variance as compared to the iterative strategies (an order of magnitude in our experiments).

As it was the case in the case of the experiments from chapter 3, the algorithms that extend Confidence Sequences (GCSQ and ICSQ) dominate the other strategies. We note that for GCSQ we set the size of the initial stage (Epoch 1 in Algorithm 5) to always be 16 (corresponding to a $\delta = 0.1$) for every sampled arm). The reason, was that Confidence Sequences quickly becomes impractical if we continue to decrease the failure probability with every sampled arm (due to its theoretical—and probably very loose—exponential dependency on $\frac{1}{\delta}$ from theorem 2.4.7).

The empirical failure probabilities are significantly smaller than the parameter δ we used in the experiments. Even if $\delta = 0.1$, the empirical accuracy for finding an an optimal arm was above 0.99. Even when setting the same value for δ for every arm i for IHR, GHR, ILHR and GLHR, empirically we get

significantly smaller failure probabilities.

Of all the advantages of the infinite-armed bandit algorithms discussed here, the most significant may be the weak assumptions that are made: the only requirement is the probability of sampling a good enough arm be nonzero. It is thus an important topic of future research to compare the strategies described in this chapter with local search strategies and planning algorithms that are designed to take advantage of relations between arms or policies [Kleinberg et al., 2008; Bubeck et al., 2008; Bubeck and Munos, 2010].

5.6 Summary

We introduced an infinite-armed bandit framework that is tailored to optimization problems to which local search cannot be applied. It is closely related to the finite PAC multi-armed bandit model. We presented an almost-tight lower bound and three algorithms that solve the problem and provided analyses proving that these algorithms achieve polynomial sample complexity bounds. We showed how a decision maker can balance between allocating pulls to get high-accuracy estimates and sampling new arms to find ones with higher expected rewards.

The framework models applications where good solutions are plentiful—where a good arm can be found by random sampling. It was shown that some non-trivial planning problems (such as two encountered in established video games) can be solved handily by exploiting this insight, even in the face of stochastic outcomes.

Chapter 6
The Cross-Entropy Method Optimizes for Quantiles

6.1 Introduction

For real-life optimization problems in which function evaluations require physical measurement or complex simulations, the same inputs can be assigned different scores [Fitzpatrick and Grefenstette, 1988]. Optimizing in the face of such noisy functions adds several additional difficulties beyond the challenge of ordinary optimization. One well known fact is that these inconsistent evaluations can mislead an algorithm, causing it to spend too much or too little of its resources on an individual relative to its "worth". One common response is to repeat and average the evaluations, which provides more accurate estimations at the expense of additional evaluations [Heidrich-Meisner and Igel, 2009]. Note, however, that while this multiple-sample approach shrinks the variance of the noise distributions, it cannot reduce them to a single point—the problem of noisy optimization remains.

A second difficulty in noisy optimization is that the very notion of which individual is the best can have multiple interpretations. Given that an input can have more than one value, which one should "count" in the optimization process? The maximum? The mode? Depending on the form of the noise distribution, these options might not even make sense. A natural choice is the mean, or expected value. However, the median might be more appropriate for

some applications, say if the noise distribution has extreme outliers.

Existing analyses [Miller and Goldberg, 1995] and evaluations [Hansen et al., 2009] of global optimization algorithms optimizing noisy fitness functions focus on problems in which "reasonable" objective functions align. That is, the individual with the highest mean also has the highest median, 25th percentile, etc. In the case of the evaluation, this decision was made explicitly so that results remain comparable even if participants choose to optimize different objectives.

Consider, however, applications such as finance where the "right" decision depends critically on one's risk attitude. An optimization procedure applied to such a problem without concern to what objective it is attempting to optimize is of limited utility to the user. The user would want to choose an algorithm that optimizes a desired quantity, or, even better, would like to tell the algorithm what to optimize. Note that this notion of risk is relevant even in more traditional optimization problems such as jet-engine design where cost–safety tradeoffs play an important role.

In this chapter, we undertake an analysis and empirical study of exactly the issue of what objective function the cross-entropy (CE) method strives to optimize. The type of analysis we undertake holds for other global optimization techniques (e.g. genetic algorithms [Goschin et al., 2011]). but we chose to study the behavior of CE as its theoretical properties are relatively well understood.

Originally designed as a technique for the simulation of rare events in networks Rubinstein [1996], CE was later adapted to the task of optimization by casting optimal events as the rare events of interest Rubinstein [1999]. As an optimization strategy, CE has been successfully applied in a variety of tasks,

such as doing policy search in reinforcement learning (RL) Mannor et al. [2003] or performing supervised classification Mannor et al. [2005]. Although its theoretical properties are understood only in limited settings Margolin [2005]; Costa et al. [2007], CE's empirical success in a wide range of applications has put it among the state of the art methods for global optimization.

Contributions. Most prior work applies CE directly with no modifications when optimizing a stochastic evaluation function. Usually, the underlying assumption is that the noise is "well behaved" (input-independent Gaussian noise for example) and that CE performs well on average. The main focus of this chapter will be to show that this assumption of well behaved noise is commonly violated, and leads to solutions with poor expected value (sometimes worse than chance). Formally, it will be shown that this failure is due to the fact that CE optimizes for quantiles instead of expectation. Therefore, in domains where the ordering determined by the expectations is different than the one determined by the targeted quantile, performance is poor. We will propose a simple, alternative algorithm that accomplishes the correct task. Our formal proof will be based on a model used to establish similar properties for selection rules in simple genetic algorithms Vose [1998a].

Empirically, we show that noise distributions with the above property occur naturally in a variety of commonly studied stochastic optimization problems. In particular, we will use domains from operations research (Inventory Control), policy search in RL (Tetris) and games (Blackjack) to demonstrate this claim. We show that even in small Markov Decision Processes, the noise distributions over the returns of policies can have a wide variety of shapes, supports, and variances.

6.2 Related Work

As mentioned, CE has had significant empirical success in a number of settings, among them buffer allocation Alon et al. [2005], scheduling, and vehicle routing. More references and applications are described in the standard CE tutorial Boer et al. [2005] or in the detailed monographs on the topic Rubinstein and Kroese [2004]; Chang et al. [2007]. The first paper to apply the CE method in the context of RL for policy search was Mannor et al. [2005]. The idea of using CE to search in a parameterized policy space was subsequently used to obtain results that were orders of magnitude better than previous approaches in a challenging RL domain—Tetris Szita and Lörincz [2006]; Szita and Szepesvári [2010a], which we will also address here. More recent papers compare CE with standard RL techniques Kalyanakrishnan and Stone [2009] and establish interesting connections with other policy-search algorithms like CMA-ES and PI^2, generalizing several design choices made in standard CE Stulp and Sigaud [2012]. An interesting application of CE in the context of sample-based motion planning for robotics Kobilarov [2011] uses a mixture of Gaussians instead of the standard, unimodal distributions usually used for CE. The same idea also appears in the work of Bardenet and Kégl [2010].

On the theoretical side, several initial proofs Rubinstein and Kroese [2004]; Margolin [2005] established convergence properties of modified versions of CE under certain assumptions. In a more recent paper, Costa et al. [2007] prove the asymptotic convergence of the standard version of CE for discrete optimization. An underlying assumption of the results above is that there is no noise in the function to be optimized. Previous empirical evidence Rubinstein and Kroese [2004] suggests that standard CE behaves well in noisy settings at least for certain domains. To the best of our knowledge, the only theoretical

result that discusses the convergence of a modified CE algorithm in a noisy setting Chang et al. [2007], proposes sufficient, but impractical modifications to CE to address arbitrary noise, in addition to adding extra parameters to the algorithm. (See Section 6.3.2 for more details.)

6.3 Algorithms

In the previous chapters, we started by formally describing a model and an (optimization) problem and then described algorithms that solved it. The complementary approach (which we undertake in this chapter) is to fix an algorithm (or better said an algorithmic template) and choose a model (or several) to understand its properties. The reason for this type of approach is to help us understand algorithms or heuristics that have proven successful in practice, but for which the theoretical understanding is mostly lacking. In the context of stochastic optimization, CE is such an algorithm and its practical success in solving a variety of optimization problems is a good motivation to better understand its properties and limitations.

The key idea of CE is to maintain a distribution over a space of inputs and update that distribution iteratively so that its support focuses only on the optimal solutions. In the following subsections, we will discuss the standard CE algorithm, a natural modification to CE for noisy settings (that fails to address the main issues correctly) and a modified algorithm that we call *Proportional CE*.

6.3.1 The Cross-Entropy Method

For a fixed iteration t, a distribution $\mathcal{D}_t$ over an input space $\mathcal{X}$ (usually a subset of $\mathbb{R}^n$ or $\{0,1\}^n$), and query access to a (possibly noisy) function $F : \mathcal{X} \to \mathbb{R}$ to be optimized, CE proceeds in three phases that are executed iteratively. In the **first phase**, it samples a set of N inputs $x_i \sim \mathcal{D}_t, i \in [1, N]$ and evaluates them. In the **second phase**, it ranks the inputs according to their values $F(x_i)$ and selects a size ρN top (or "elite") subset (for some $\rho \in (0\%, 100\%)$) or, equivalently, the inputs with higher evaluations than the $1 - \rho$ sample quantile. Finally, in the **third step**, it uses the elite subset to set the new parameters for $\mathcal{D}_{t+1}$, most commonly by determining the maximum likelihood estimators for the elite set. CE is executed either for a fixed number of iterations or until the distribution is concentrated on a small subregion of the input space. In the RL setting, solutions are encodings of policies, and F executes the policy in the domain, yielding the return of that trajectory.

The algorithm is parameterized by the choice of N, ρ, the parameters of the initial distribution $\mathcal{D}_0$ over the input space and the family of distributions $\mathcal{D}_t$ (which includes the initial distribution). The distributions $\mathcal{D}_t, t \geq 0$ are usually part of the natural exponential family and the standard choice is the normal distribution (for continuous inputs) or the Bernoulli (or multinomial) distribution (for discrete inputs). To instantiate the algorithm for a particular distribution, one needs to specify the update rule for the third stage. In general the rule is determined by solving a stochastic program (for the general version the reader is referred to algorithm 2.1 in Boer et al. [2005]). We will give an example of update rules for the case of $\mathcal{D}_t$ being multi-variate Bernoulli distributions over $\{0,1\}^n$ (for the normal distribution see Stulp and Sigaud [2012] for example). In this case, $\mathcal{D}_t$ are thus parameterized by a vector of elements

$p_i^t \in [0,1], t \geq 0, i \in \{1,...,n\}$ (where p_i^t is the parameter for the ith Bernoulli distribution at generation t).

In the first stage of generation t, CE samples N Bernoulli vectors $\mathbf{x}_j$ and evaluates them. In the second stage CE computes the "elite" or the $1-\rho$ sample quantile F_t^ρ based on the evaluations and in the third stage it updates p_i's using the formula: $p_i^{t+1} = \frac{\sum_{j=1}^N I[F(\mathbf{x}_j) \geq F_t^\rho] \; I[x_{j,i}=1]}{\sum_{j=1}^N I[F(\mathbf{x}_j) \geq F_t^\rho]}$, where I is the identity function and $x_{j,i}$ is the ith component of the jth vector. So each component p_i^{t+1} is set to reflect the ratio of 1 values of the bits at position i among the elite sample.

A number of techniques have been used to address various practical observations regarding the behavior of the algorithm. One of the most common problems is that the distribution sometimes prematurely converges to a single point. This is because in practice, the variance of the "elite" population is much smaller than the population at large, leading to a decrease in the variance of $\mathcal{D}_t$. One solution is to artificially maintain the variance of the population high, which was one of the key ideas leading to the empirical success of CE in Tetris Szita and Lörincz [2006]. Another common technique is to smooth the updates of the parameters of the distribution over generations.

As already mentioned, the structure of the algorithm is based on ideas from rare event simulation Rubinstein [1996] and the key insight is that the distribution maintained by the algorithm is continuously updated to minimize the Kullback-Leibler distance to the ideal distribution that is focused on the correct solution. (Full details can be found in Boer et al. [2005].)

It is important to note that the algorithm is often applied "as is" in settings where the evaluation of the function F is corrupted by an arbitrary noise process. The key point of the chapter is that the algorithm optimizes a quantile

measure that, in certain situations of practical interest, is different from optimizing for the expected value of the function. In these cases, in addition to the natural difficulties in optimizing in high dimensional spaces, the algorithm is further hindered by the fact that it is explicitly attempting to find solutions of lower expected quality.

6.3.2 The m-Cross-Entropy Method (mCE)

An intuitive way to mitigate the impact that the optimization for quantiles has on the expectation of the solution is to take the mean of m samples for each queried input (for example this was applied by Mannor et al. [2003] to address the noise in evaluations). Then, the standard version of CE can be applied considering the value of an individual $\hat{F}(x) = \frac{\sum_{i=1}^{m} F_i(x)}{m}$ (where $F_i(x)$ are i.i.d samples from $F(x)$). By the central limit theorem, as m increases, the noise distribution for each evaluated input will become concentrated around its mean, thus eliminating (in the limit) the problem of inconsistent orderings for the mean and any quantile values.

One obvious problem with mCE is the need to choose a reasonable value for m when not enough information is available about the noise distributions. If m is too small, the undesired phenomenon can still occur. If, on the other hand, m is too large, for a fixed number of function evaluations per generation (thus counting the repeated evaluations of the same point), two problems can occur. On one hand, it is possible that not enough inputs are evaluated for mCE to succeed in finding the optimal (or a reasonable) solution. On the other hand, improvements in the expected value from resampling come at the cost of increased variance in the quality of the final population, as issues of early convergence are exacerbated when the set of sampled points shrinks. As has

been observed in early work on evolutionary methods **?**, the tradeoff between m, N, and the total number of generations is a complex one with no universally "right" answer. We will discuss an example in Section 6.7.4 to illustrate these tradeoffs in the context of mCE.

A different (but related to mCE) approach to modifying CE for optimizing in stochastic environments was proposed by Chang et al. [2007] under the name of "Model Reference Adaptive Search 2" (MRAS_2). In addition to other modifications, the algorithm requires the designer to specify a rule m_k for the number of times each input is evaluated at each generation k. For the algorithm to converge (under certain assumptions), m_k is required to increase with k and the authors suggest $m_k = \Omega(c^k)$ (for some $c > 1$) or $m_k = \Omega(k)$ as possible rules for several classes of noise. (See Section 4.2.3 in Chang et al. [2007].) While the above rules are sufficient for convergence in certain scenarios, they lead to impractical algorithms for all but the simplest domains, in addition to adding the need to specify the correct m_k sequence.

6.3.3 Proportional Cross-Entropy

We will now propose a variant of the standard CE method that seeks the input that optimizes the expected value of the evaluation function. In addition to seeking high expected value solutions, the method has the additional benefit of not requiring the parameter ρ. The main modification is a change to the second phase of the CE algorithm: Instead of selecting a subset of the samples from $\mathcal{D}_t$, the algorithm weights each input according to its value (normalizing with respect to the difference between the minimum and the maximum value to address negative evaluations). Then, in the third stage, it sets the parameters of distribution $\mathcal{D}_{t+1}$ according to these weighted inputs.

Concretely, for the same case of the multivariate Bernoulli distributions and using the same notation as for CE, the weights for the sampled inputs are $w_j = \frac{F(\mathbf{x}_j)-m}{M-m}$, where $m = \min_{j=1}^{N}\{F(x_j)\}, M = \max_{j=1}^{N}\{F(x_j)\}$ (the case of $M = m$ can be handled by setting all weights to be equal). Then, in the final stage, the new parameters for the distributions are set according to the equations: $p_i^{t+1} = \frac{\sum_{j=1}^{N} w_j I[x_{j,i}=1]}{\sum_{j=1}^{N} w_j}$.

The idea of modifying the definition of what an "elite" set represents is not new. In CMA-ES, the mechanism for deciding the relative importance of the top ρN samples can be chosen by the algorithm designer Hansen and Ostermeier [2001], but it is still the case that the samples outside the "elite" set have no influence in shaping the distribution for the next iteration. In PI^2 Stulp and Sigaud [2012], an exponential decay scheme weights all inputs according to their evaluation. Thus, the algorithm we propose can be viewed as being an instantiation of a general template for designing CE-like algorithms. Our contribution is to link the "elite" set selection mechanism (phase two of the CE algorithm) to the optimization objective of the algorithm. To simplify comparison and analysis, we keep all the other design choices unchanged and focus only on comparing the standard algorithm with Proportional CE.

6.4 Illustration - Tetris

To illustrate the main point of the chapter, we will describe a simple optimization example. We ran an experiment with a constrained version of the video game Tetris using a setup similar to Szita and Szepesvári [2010a]. We used a 10×8 board, the feature set from Bertsekas and Ioffe [1996], and allowed only the "S", "Z," and "I" tetrominoes to appear with probabilities of 45%, 45% and

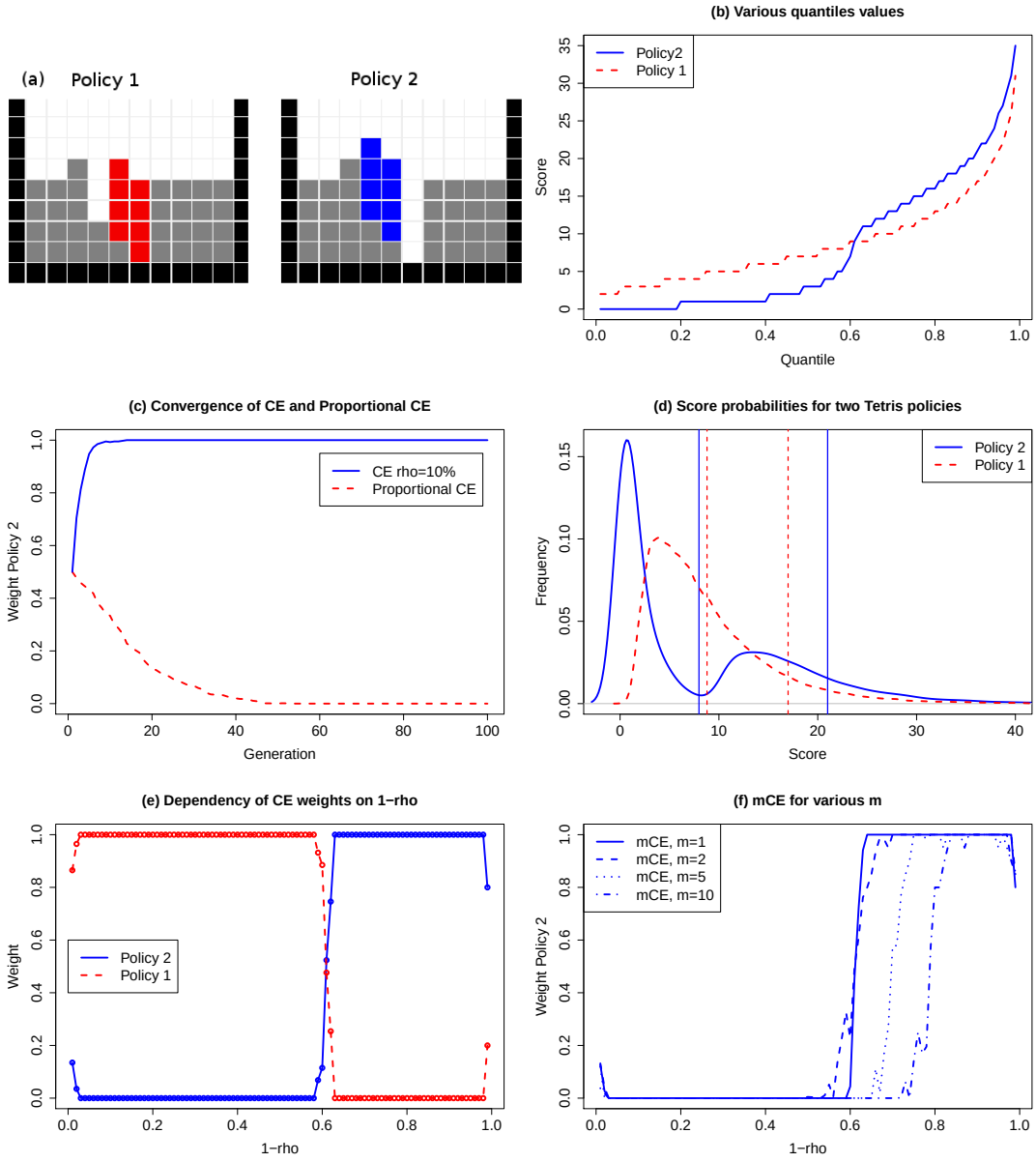

Figure 6.1: A simple experiment for two policies in Tetris.

10%. The score bonuses for clearing $1, 2, 3$, and 4 lines were $1, 2, 3$ and 10, respectively. The optimization problem is to pick among two policies the one that has the best average score. Policy 1 is less "risky" and it is represented in Fig. 6.1(a) (left). It receives a pair of S tetrominoes that it positions as shown in the diagram. Policy 2 positions the two S tetraminoes as in Fig. 6.1(a) (right). Both policies continue by executing the same fixed strategy (that was obtained offline by running a CE algorithm as in section 6.7.3) when exposed to random tetrominoes arriving according to the distribution specified above. We executed each policy for 50k steps and plotted the distribution over scores in Fig. 6.1(d). The ordering by the means is different from the ordering determined by the 90% quantile and in fact the same holds true for a wide range of quantiles (as can be observed in Fig. 6.1(b), where we plot the empirical quantile functions for the scores of Policy 1 & 2).

We ran CE and Proportional CE with a Bernoulli distribution with just one component that captures the binary choice between Policy 1 and 2. Every experiment was executed 100 iterations and was repeated 20 times with $N = 200$. The result in Fig. 6.1(c) shows the two algorithms converging to different solutions, with CE($\rho = 10\%$) converging to Policy 2 (which has a higher 90% quantile) and Proportional CE converging to Policy 1 (which has a higher expectation). To verify the phenomenon for a wide range of ρ values, we ran CE with $\rho \in (1\%, 99\%)$ and plotted the results in Fig. 6.1(e). The solution for CE(ρ) is consistent with the ordering of the quantiles from Fig. 6.1(b) with a transition stage for $\rho \approx 40\%$ (i.e. for the 60%th quantile), where the quantiles values for the two policies are similar.

To illustrate the behavior of mCE, we repeated the experiment for various values of m. For a fixed m, we varied $\rho \in (1\%, 99\%)$, keeping the number

of evaluations per generation fixed and plotted the results in Fig. 6.1(f). For increasing values of m, although the range of ρ values for which the suboptimal policy is chosen shrinks, the phenomenon does not disappear.

6.5 Model

We analyze the properties of the algorithms in the classical infinite population model Vose [1998b] extended to treat noisy functions. As the name suggests, the model assumes that an infinity of function evaluations are available at each iteration (we note that the algorithms are fixed and don't take advantage of such knowledge).

This model provides a useful abstraction that allows for analysis without the complication of finite sampling effects. And, while obviously unrealizable, it is a reasonable model for studying qualitative properties of some global optimization algorithms and is consistent with results for situations where the number of evaluations is large.

Of course, this simplification is a double-edged sword. While it reduces complications in proofs, it also hides possibly relevant details. Hence, Section 6.7 provides supporting evidence from computational simulations.

The model is very similar to a finite multi-armed bandit model in the sense of being defined for a finite set $I = \{1, 2, \ldots, n\}$ and a noisy function $F : I \to [0, 1]$ to be maximized. Whenever $F(i), i \in I$ is evaluated it will return a sample from a distribution $\mathcal{P}_i$ (that depends on i) over $[0, 1]$.

For simplicity of analysis, we will assume that $\mathcal{P}_i$ have continuous and strictly decreasing complementary cumulative distribution (ccdf) functions G_i and common support $[0, 1]$. We also denote by g_i the densities corresponding

to distributions $\mathcal{P}_i$. Since the ccdf's G_i are continuous and strictly decreasing, the quantiles $q_i(\rho)$ are uniquely defined for any probability $\rho \in [0\%, 100\%]$: $q_i(\rho) = G_i^{-1}(1-\rho)$ (an example meeting these constraints is if g_i are beta density functions with different parameters).

The standard goal for optimization in the above model is to find an $x^* = \arg\max_i \mathbb{E}_{F(i)\sim\mathcal{P}_i}[F(i)]$. An alternative goal is to optimize for quantiles: for a fixed ρ, find $x^\rho = \arg\max_i q_i(\rho)$. It is possible that the two optimization objectives are aligned ($x^* = x^\rho, \forall\rho$) as in the case of an additive noise distribution like Beta(α, α) that is input-independent for example. But, for general $\mathcal{P}_i$ and ρ, the objectives are different.

6.6 Theoretical Results

We will study the optimization objectives of CE and Proportional CE assuming they are allowed an infinite number of evaluations at every iteration. Informally, the idea of having an infinite number of evaluations is to allow the entire distribution over $F(x)$ values for a particular value x to be "present" in the set of samples for a fixed generation. This assumption naturally removes the need for a parameter N. But we still need to study the convergence properties for a particular family of distributions $\mathcal{D}_t$ and for a particular setting of the initial parameters $\mathcal{D}_0$. The reason is that in general, without fixing $\mathcal{D}_0$, there will always be a setting for which the algorithms are guaranteed not to converge:

Fact 6.6.1. *In the setting above, there exists an initial distribution $\mathcal{D}_0$ that forces both CE and Proportional CE to never find x^*.*

Proof. Assume x^* is unique in maximizing $\mathbb{E}[F]$. Let's consider a binary encoding of the input space in $\log(n)$ bits and consider $\mathcal{D}_t$ to be multivariate

Bernoulli distributions over vectors of size $\log(n)$. Let's assume wlog that bit 0 of x^* is set to 1. Let's now choose the initial distribution $\mathcal{D}_0$ to have a Bernoulli($p = 0$) distribution on the first bit of the representation. Then x^* will never be sampled from $\mathcal{D}_t, \forall t \geq 0$. □

Since the results are distribution-dependent, we will prove the convergence properties for a natural choice of a distribution and initial parameters and conjecture that the results can be extended to other distributions as well. Both algorithms will use a multinomial distribution $\mathcal{M}^t$ over the input space with all parameters $w_i^0 = \frac{1}{n}, i \in I$ initially (hence the subscript 0). Using a multinomial distribution is reasonable in this context since it encodes the degree of "belief" the algorithm has in a particular i being the optimal argument of the function.

In this context, a fixed infinite population is thus described by a real-valued weight vector w representing the proportions of the inputs, with $\sum_{i \in I} w_i^t = 1$ and $w_i^t \geq 0$ for $t \geq 0$, where t is the index of the iteration.

Given that $w_i^t \in [0, 1]$ represents the proportion of the population at time t occupied by the input i, $f^t = \sum_{i \in i} w_i^t g_i$ is the density of the mixture distribution that characterizes the entire population at time t. We call this combined distribution over possible function values the *population distribution*.

6.6.1 Proportional Cross-Entropy

We note that since in the model described above, the range of the function to be optimized (F) is $[0, 1]$, the weight of every sampled input is exactly its evaluation (the minimum value is 0 and the maximum value is 1).

Ignoring the normalization factor for a moment, in the infinite population

model, inputs i occupy w_i^t fraction of the population at time t and those inputs have their evaluations drawn from the density $g_i(x)$. Because we are imagining an infinite population (and since the support of g_i is $[0,1]$), each of these evaluations actually appears and the fraction of the time function value $x \in [0,1]$ appears due to input i is $w_i^t g_i(x)$. The total weight of this input in the resulting population is therefore $w_i^{t+1} = \int_{\mathcal{X}} w_i^t x g_i(x) dx / Z$.

As a result, if we define

$$v_i = E_{x \sim g_i}[x] = \int_{[0,1]} x g_i(x) dx \tag{6.1}$$

the expected value of the distribution over function values for input i, then the evolution of the weights (i.e. parameters of the multinomial distribution $\mathcal{M}^t$) while executing Proportional CE is given by:

$$w_i^{t+1} = \frac{w_i^t v_i}{\sum_{j \in 1 \ldots n} w_j^t v_j}, \forall i \in 1 \ldots n \text{ and } \forall t \geq 0. \tag{6.2}$$

For simplicity and without loss of generality, we assume that $v_1 < v_2 < \cdots < v_n$. That is, inputs are sorted in increasing expected function value order and all expected values are unique. (If v_i is not unique, we can consider the mixture of the distributions that have the same v_i as a single individual that has as its value distribution the weighted mixture.)

We will prove the Proportional CE indeed optimizes for expectation. The theorem states that the weight of the individual with the largest expected value will asymptotically converge to 1.

Theorem 6.6.2. *When running Proportional CE ($\mathcal{M}_0(w_i^0 = \frac{1}{n})$), the algorithm will asymptotically converge to x^*. Concretely*

$$\lim_{t \to \infty} w_i^t = 0, \forall i \in 1 \ldots n-1, \text{ and } \lim_{t \to \infty} w_n^t = 1. \tag{6.3}$$

Proof. Using induction, we show $w_i^t = w_i^0/(\sum_j w_j^0(v_j/v_i)^t)$. For the base case, note that $w_i^0 = w_i^0/(\sum_j w_j^0)$ because w^0 is normalized.

For the inductive step,

$$w_i^{t+1} = w_i^t v_i/(\sum_j w_j^t v_j) \tag{6.4}$$

$$= w_i^t/\sum_j w_j^t(v_j/v_i) \tag{6.5}$$

$$= \frac{w_i^0/(\sum_j w_j^0(v_j/v_i)^t)}{\sum_j[w_j^0/(\sum_k w_k^0(v_k/v_j)^t)](v_j/v_i)} \tag{6.6}$$

$$= \frac{w_i^0(v_i)^t/(\sum_j w_j^0(v_j)^t)}{\sum_j[w_j^0(v_j)^t/(\sum_k w_k^0(v_k)^t)](v_j/v_i)} \tag{6.7}$$

$$= w_i^0(v_i)^t/\sum_j w_j^0(v_j)^t(v_j/v_i) \tag{6.8}$$

$$= w_i^0/\sum_j w_j^0(v_j/v_i)^{t+1}. \tag{6.9}$$

For individual n and any $i < n$, $v_n > v_i$. Thus, using the result above,

$$\lim_{t\to\infty} w_n^t = \lim_{t\to\infty} w_n^0/(\sum_j w_j^0(v_j/v_n)^t) \tag{6.10}$$

$$= \lim_{t\to\infty} w_n^0/w_n^0 = 1. \tag{6.11}$$

Since $\sum_j w_j^t = 1$, for $i < n$, $\lim_{t\to\infty} w_i^t = 0$.

□ □

6.6.2 The Cross-Entropy Method

Before stating the theorem that characterizes The Cross-Entropy Method, we discuss the new notation: define $x_i \in \mathcal{X}$ such that $\Pr_{x\sim g_i(x)}(x \geq x_i) = \rho = G_i(x_i)$. Thus, x_i is the value of the quantile function $q_i(1-\rho)$ of distribution g_i and it is unique by our assumptions noted in the model description (i.e. that G_i

are continuous and strictly decreasing). It is the value x_i at which ρ fraction of the noisy function values are above x_i and $1-\rho$ fraction of the noisy function values are below x_i. Once again, we assume without loss of generality that $x_1 < x_2 < \cdots < x_n$—all of these thresholds are unique and sorted.

We will now prove that The Cross-Entropy Method optimizes for quantiles[1]:

Theorem 6.6.3. *When running CE($\rho, \mathcal{M}_0(p_0^x = \frac{1}{n})$) to optimize a function F, the algorithm will asymptotically converge to $x^{1-\rho}$ (i.e. to a multinomial with $w^{\infty}_{x^{1-\rho}} = 1$). Concretely:*

$$\lim_{t\to\infty} w_i^t = 0, \forall i \in 1 \dots n-1 \text{ and } \lim_{t\to\infty} w_n^t = 1. \tag{6.12}$$

Proof. **Part 1 - Derive formulas for the evolution of the parameters of $\mathcal{M}^t$ as t increases**.

The first idea of the proof is to use the fact that the third stage of the CE algorithm is maximum likelihood estimation for the multinomial distribution based on the top ρ percent (or equivalently, the $1-\rho$ quantile) of the evaluated, infinite population. For a motivation of the maximum likelihood claim, the reader is referred to Boer et al. [2005] (Remark 2.5 in particular). This perspective automatically provides closed form solutions for the updates of the parameters for $\mathcal{M}^t$ (as opposed to solving a potentially complicated stochastic program as it is the case in general). In particular:

$$w_i^{t+1} = w_i^t \frac{G_i(x_c^t)}{\rho} \tag{6.13}$$

where x_c^t is the threshold value in $[0,1]$ that separates the elite from the rest

[1] We note that given the usual values of ρ ($\rho < 0.5$), CE is thus risk-seeking which can be dangerous in practice.

of the population. In words, each component's new weight w_i^{t+1} is proportional to the relative tail probability mass (which also takes into consideration the previous weight w_i^t) with respect to the other components among the elite sample.

We label the threshold $x_c^t \in \mathcal{X}$ the *population common point*. It is the value at iteration t for which

$$\sum_{i \in 1 \ldots n} w_i^t G_i(x_c^t) = \rho. \tag{6.14}$$

Thus, x_c^t for the *population distribution* is analogous to x_i for individual i's noise distribution. This x_c^t threshold plays the important role of deciding the change in weights for the individuals in the next iteration. When evaluations are made for the w_i^t fraction of individuals belonging to individual i, $G_i(x_c^t)$ fraction of them will survive because their values will surpass x_c^t.

The rest of the argument consists of three major steps:

1. **(Part 2)** $\forall t > 0, x_c^t < x_c^{t+1}$: The population common point is increasing.

2. **(Part 3)** $\exists t' > 0$ s.t. $x_c^{t'} > x_{n-1}$: After a finite number of iterations, the population common point exceeds the second largest threshold (and will not go below it again due to Step 1).

3. **(Part 4)** $x_c^t > x_{n-1} \implies \lim_{t' \to \infty} w_n^{t'} = 1$: If the population common point is above the second largest threshold, then the population will converge to the individual with the largest threshold.

Putting these three facts together completes the proof. We prove each in turn.

Part 2 - The population common point always increases.

At iteration t, let $W = \{i \text{ s.t. } x_i > x_c^t\}$ be the "winners" (any input i with the threshold above the population common point at time t) and $L = \{i \text{ s.t. } x_i \leq$

$x_c^t\}$ be the "losers" (any input i with the threshold at or below the population common point at time t). (We suppress the dependence of W and L on t to simplify notation).

Define $\delta_i = G_i(x_c^t) - \rho$ if $i \in W$ and $\delta_i = \rho - G_i(x_c^t)$ if $i \in L$. It captures the amount that input i's G_i value deviates from ρ at the current population common point. Note that $\delta_i \geq 0$ for all i and that $\delta_i > 0$ for at least one $i \in W$ (otherwise, all inputs are tied in their order statistics). Also, note that $w_i^{t+1} > w_i^t$ for $i \in W$ and $w_i^{t+1} \leq w_i^t$ for $i \in L$. These facts follow from Equation 6.13 and the definition of W and L.

We proceed by contradiction. Assume the population common point remains the same or decreases, $x_c^{t+1} \leq x_c^t$. This assumption implies $G_i(x_c^{t+1}) \geq G_i(x_c^t)$ (by the fact that G_is are strictly decreasing). Now:

$$\begin{aligned}
\rho &= \sum_{i \in 1 \ldots n} w_i^{t+1} G_i(x_c^{t+1}) && (6.15)\\
&\geq \sum_{i \in 1 \ldots n} w_i^{t+1} G_i(x_c^t) && (6.16)\\
&= \sum_{i \in W} w_i^{t+1} G_i(x_c^t) + \sum_{i \in L} w_i^{t+1} G_i(x_c^t) && (6.17)\\
&= \sum_{i \in W} w_i^{t+1} (\rho + \delta_i) + \sum_{i \in L} w_i^{t+1} (\rho - \delta_i) && (6.18)\\
&= \rho + \sum_{i \in W} w_i^{t+1} \delta_i - \sum_{i \in L} w_i^{t+1} \delta_i && (6.19)\\
&> \rho + \sum_{i \in W} w_i^{t} \delta_i - \sum_{i \in L} w_i^{t} \delta_i && (6.20)\\
&= \sum_{i \in W} w_i^{t} (\rho + \delta_i) + \sum_{i \in L} w_i^{t} (\rho - \delta_i) && (6.21)\\
&= \rho && (6.22)
\end{aligned}$$

which is a contradiction ($\rho > \rho$). Thus, the population common point must increase.

Part 3 - The population common point eventually exceeds the second largest threshold.

Let $\Delta = G_n(x_{n-1}) - \rho$. Note that $\Delta > 0$ because $G_n(x_n) = \rho$ (by definition of x_n) and $G_n(x_{n-1}) > G_n(x_n)$ (because the x_is are sorted and the G_is are strictly decreasing). This quantity represents how likely it is for an evaluation for input n to fall between x_{n-1} and x_n.

Now, as long as $x_c^t \leq x_{n-1}$, $G_n(x_c^t) \geq G_n(x_{n-1})$ (because the G_is are strictly decreasing). By Equation 6.13,

$$w_n^{t+1} = w_n^t G_n(x_c^t)/\rho \tag{6.23}$$

$$\geq w_n^t G_n(x_{n-1})/\rho \tag{6.24}$$

$$= w_n^t(\Delta + \rho)/\rho \tag{6.25}$$

$$= w_n^t(1 + \Delta/\rho) \tag{6.26}$$

Thus, $w_n^{t'}$ grows without bound as long as $x_c^{t'} \leq x_{n-1}$. Therefore, there must be some time point t' when $x_c^{t'} > x_{n-1}$ (and, due to Step 1, it will never go below x_{n-1} again).

Part 4 - Once the population common point exceeds the second largest threshold, convergence to input n is guaranteed.

Let $\hat{t}$ be the first time the population common point goes over the second largest threshold (and this threshold is well defined as the time is discrete and the support of the noise distributions is continuous) and define a to be the probability that the evaluation of the input with the second largest threshold is larger than ρ and smaller than the resulting population common point: $a = \rho - G_{n-1}(x_c^{\hat{t}})$. For all $i < n$,

$$w_i^{t+1} \quad = w_i^t G_i(x_c^{\hat{t}})/\rho \tag{6.27}$$

$$\leq w_i^t G_{n-1}(x_c^{\hat{t}})/\rho \tag{6.28}$$

$$= w_i^t(\rho - a)/\rho \tag{6.29}$$

$$= w_i^t(1 - a/\rho). \tag{6.30}$$

As a result of the fact that w_i^t is multiplied by a number bounded away from 1, as t increases, w_i^t goes to 0, as desired. □

6.7 Experiments

The goal of this section is to present empirical results in support of the claim that CE fails to optimize for expectation in naturally occurring noisy environments. The domains presented below have diverse characteristics and are used as optimization benchmarks by various communities. For every domain, we will describe its modeling as a Markov Decision Process, the policy space we searched in, and the parameters we used. For CE, we focused our search for good ρ parameters in ranges that make sense given the algorithm's motivation and the way it is used in practice ($\rho \leq 50\%$).

6.7.1 Die4

Die4 is a simple game we created to study optimization under risk. The game is played with a regular die. At each point in time the player can decide to roll the die or to stop and accumulate the sum of all die values until the current time. If, however, the die comes up 4 at any roll, the game ends and the player gets 0 points. Depending on the attitude towards risk, policies can stop earlier and have a good chance of gaining a non-0 reward or stop later with a high risk of gaining nothing.

Model. The states are the possible sum values for a die (natural numbers $>= 2$), the actions are *roll* and *stop* (both can be terminal), the rewards are 0 for failure and the value of the state for success. The transitions for the *roll* action are dictated by the roll of the die according to the definition of the game

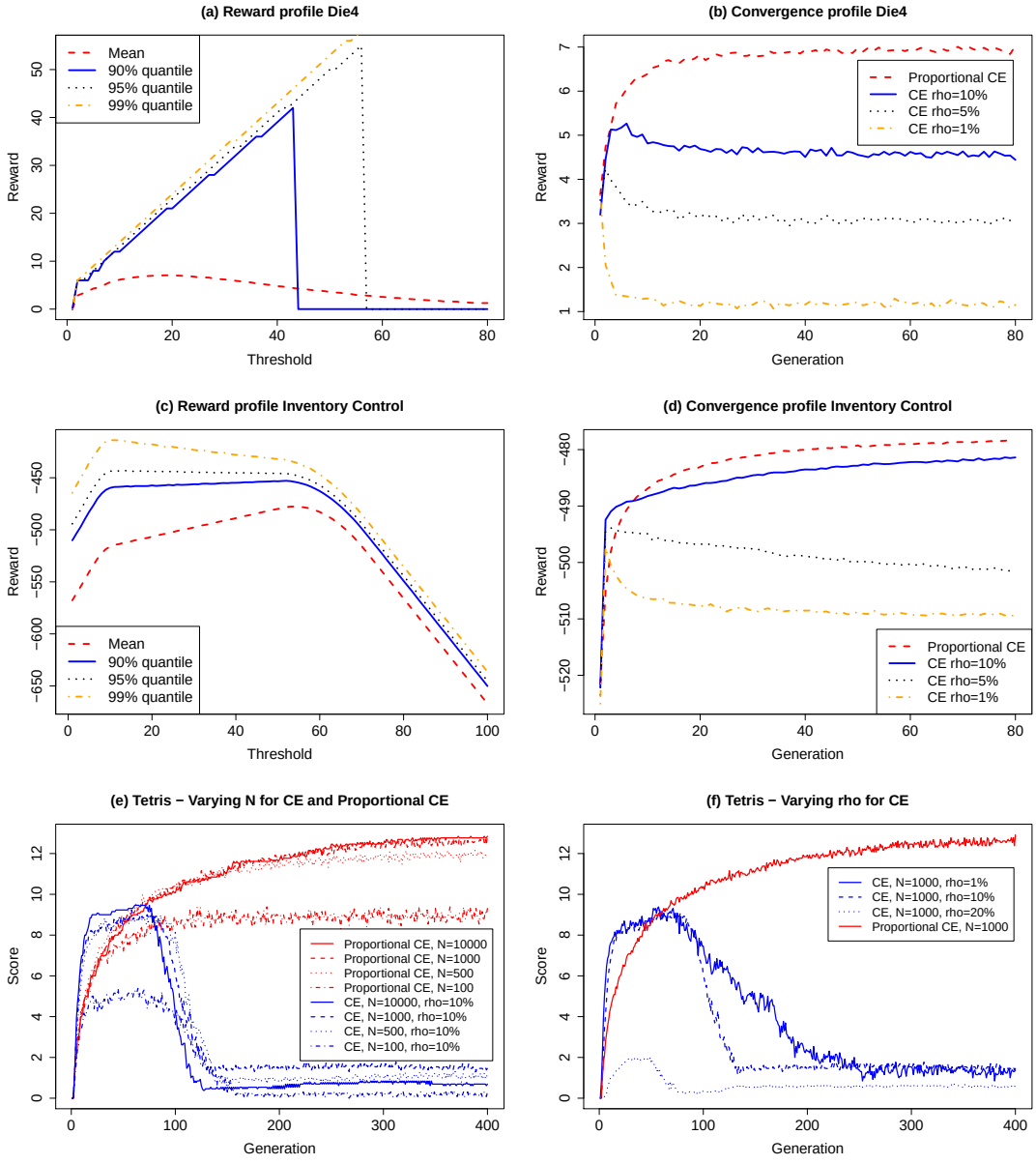

Figure 6.2: Die4, Inventory Control and Tetris experiments

while the transition for *stop* is to the same state (and the game stops). We follow Goschin et al. [2011] and define the policy space to be parameterized by a threshold x encoding a simple rule: "*stop* as soon as the sum of die values is at least x or *roll* otherwise". The (expected) optimal value is ≈ 7.2 and it is obtained by setting $x^* = 17$. In Fig. 6.2(a), we plot the curve for the mean scores for all the policies with thresholds in $\{2, ..., 80\}$. We also plot the curves corresponding to the $90, 95, 99\%$th quantiles.

Setup. We relax $x \in \mathbb{R}^+$ (even though the sums are discrete) so as to be able to apply CE easily. Both algorithms start with a normal distribution $N(\mu = 50, \sigma^2 = 100)$ over the set of thresholds, $N = 1000$ and are executed for 80 iterations. Each experiment is repeated 50 times and the average scores are reported.

Results. The results in Fig. 6.2(b) show Proportional CE converging to the optimal expected value. The distribution over the thresholds after 80 iterations is concentrated around the optimal threshold. On the other hand, CE with $\rho = 10\%, 5\%$ or 1% converged to sub-optimal values and actually finds solutions that are optimal according to the corresponding quantiles. The results are consistent with what the theory predicts for such a scenario where the input that yields a maximum expected value is different from inputs that determine optimal quantile values.

6.7.2 Inventory Control

Inventory Control is a standard benchmark problem from operations research. It was also used as an experimental domain in the first paper that utilized CE for policy search in RL Mannor et al. [2003]. We will describe the simplest version of the problem, which models a shop owner having to make decisions

about ordering one product.

Model. The state space consists of possible stock values at the beginning of each day: $s_t \in \mathbb{R}$ (with $t > 0$ denoting the day), and negative stock possible due to under-ordering. For each state, the action space $a_t \in [0, s_{\max} - s_t]$ is the amount of stock the owner can order at the beginning of day t (with s_{max} being the maximum stock). The transition function is determined by i.i.d. requests from clients $d_t \sim \mathcal{P}$ from a fixed, but unknown probability distribution with the next state being determined by $s_{t+1} = s_t + a_t - d_t$. The costs for "holding"(h) too much stock, "backlogging" (b) due to insufficient ordering and the price for one unit of stock c are fixed and known. We will use the same reward function as Mannor et al. [2003] $r_t(s_t, a_t, d_t) = -h \max\{0, s_t\} - b \max\{0, -s_t\} - ca_t$.

We will also use the same policy space as Mannor et al. [2003]: each policy is determined by a threshold x that sets the stock order as a function of the current stock, meaning at every time t, $a_t = \max\{x - s_t, 0\}$. Searching in this policy space is thus equivalent to finding the best threshold x.

Setup. For the experiments, we instantiated an inventory control problem with the following characteristics: $h = 5, b = 6, c = 10, s_{\max} = 100$ and with $\mathcal{P}$ being a mixture of two normal distributions with equal weights ($N(\mu = 5, \sigma^2 = 5)$ and $N(\mu = 50, \sigma^2 = 20)$) in an attempt to model a mix of small and large requests. Similarly to Die4, both algorithms start with a Normal distribution $N(\mu = 50, \sigma^2 = 100)$ over the set of thresholds, $N = 1000$ and are executed for 80 iterations. Each experiment is repeated 50 times. In Fig. 6.2(c), we plotted the curves for the means and the same set of quantiles as for Die4. The optimal expected value is around -477 and it is obtained by setting $x = 53$ while the maximum 99% quantile corresponds to an expected value of -514

and is obtained by setting $x = 10$.

Results In Fig. 6.2(d), we plot the algorithms' convergence curves. It can be observed that Proportional CE converges to a value close to the expected optimal value while CE($\rho = 1\%$) for example converges, as expected, to the value corresponding to the optimal 99% quantile. As in the case of Die4, the solutions distributions over inputs that the algorithms converge to are centered around the input values that are predicted by the theoretical results.

6.7.3 Tetris

The video game Tetris is well known for being a difficult benchmark for policy search in RL and one where CE performed very well in the past Szita and Lörincz [2006]. In our experiments we followed closely the setup from Szita and Szepesvári [2010a] that defines a simpler version of Tetris (*Stochastic SZTetris*) that only allows the S and Z tetraminoes with the goal of maintaining the difficulty of the game and make it more efficient to simulate. We used the code base from Szita and Szepesvári [2010b] and extended it to parameterize the domain. We chose the feature representation defined in Bertsekas and Ioffe [1996]. We refer the reader to Szita and Szepesvári [2010a] for an excellent presentation of the challenges of SZTetris.

In an attempt to do simulations more efficiently (so that we could run parameter search in reasonable time), we decreased the height of the SZTetris board to from 20 to 5 (the problem is far from trivial even in this modified setup). Moreover we decided to give a bonus of 10 points for clearing two lines (as compared to the default value of 2) with the goal of "infusing risk" in the game. We ran a parameter search for a reasonable value of N (convergence results can be seen in Fig. 6.2(e) where ρ is fixed to 10%) and for a good ρ value

for CE (Fig. 6.2(f) with fixed $N = 1000$). Every experiment is repeated 15 times and the results are averaged. The first observation is that while Proportional CE converges slower than CE, it will converge to better solutions than the maximum performance of CE throughout its execution. The second observation is that after its performance plateaus, CE is degrading no matter how we set the initial parameters. We made significant efforts to find a setting of the algorithm where the divergence phenomenon doesn't happen (including setting various smoothing parameters, initial variance etc.) but we were unable to eliminate it. To verify that this was not a direct cause of our setup, we ran the original code base with the original algorithm and SZTetris parameters and found the same phenomenon occurring around generation 2000 (for reasonable tractability reasons, Szita and Szepesvári [2010a] only ran the algorithm up to generation 50).

While the experiments above offer a less clear-cut perspective with respect to the main goal of the chapter, we believe they are interesting enough to report. Even in the region where CE converges, its performance is worse than what Proportional CE can achieve. We note that this seems to be a direct cause of the increased bonus for clearing two lines. In experiments with the original scores for clearing lines, the average best performances of the two algorithms are essentially the same (even though CE still degrades). This suggests that the phenomenon of optimizing for different objectives affects this "risky" version of Tetris.

6.7.4 Blackjack

In this section, we discuss two variants of blackjack and describe how differences in mechanics can lead to changes in policies when optimizing for quantiles or expectation. The first variant of the game reduces the game to its most

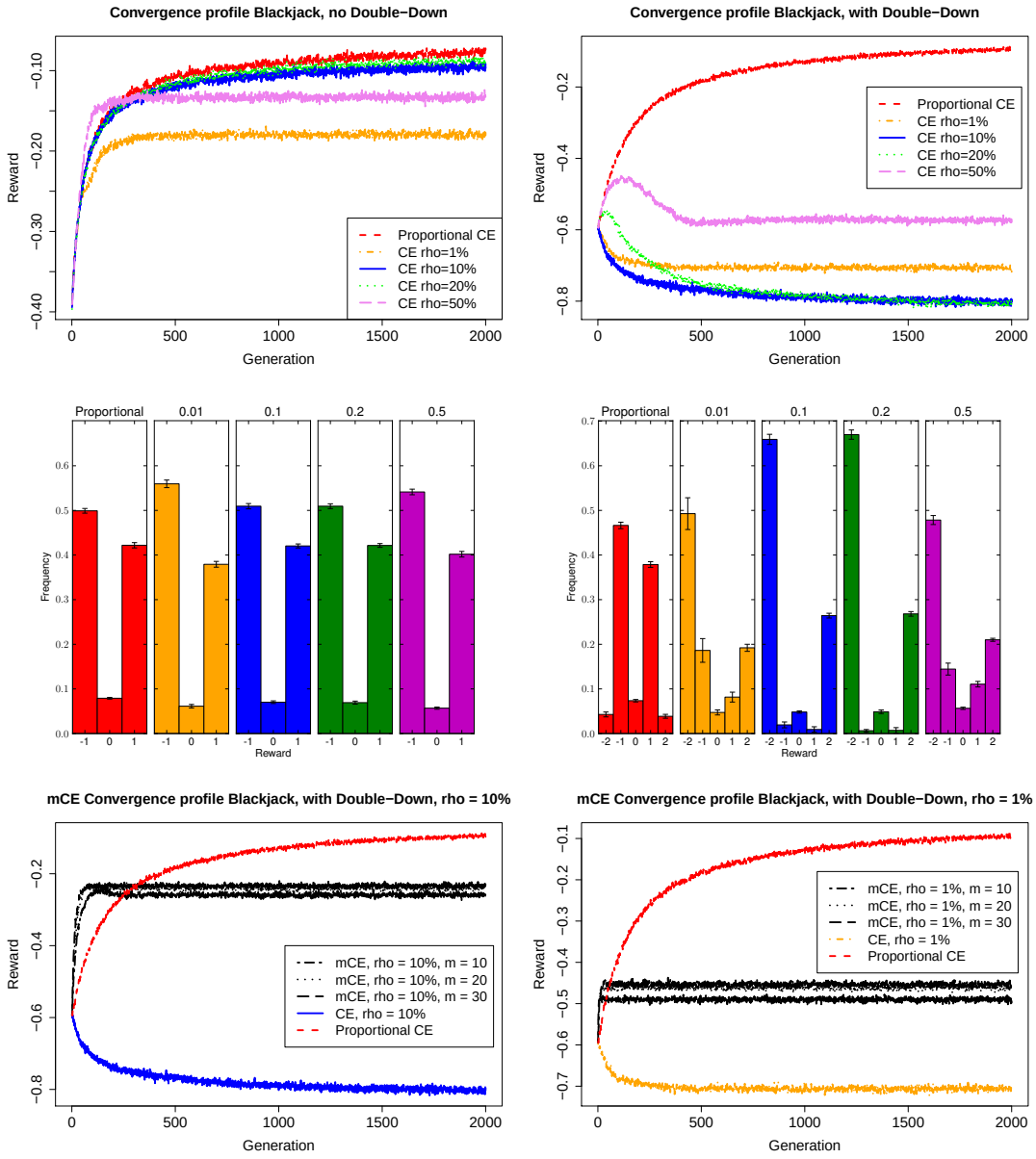

Figure 6.3: Blackjack experiments. Subfigures top (left, right): (a), (b), middle (left, right): (c), (d), down (left, right): (e), (f)

important dynamics, as described in Sutton and Barto [1998]. We also adopt the policy representation of Sutton and Barto [1998]. The state is represented by the dealer's showing card, the sum of the player's hand, and whether or not the player holds a usable ace. On hand values less than 12, the player automatically hits, because there is no chance of busting. Therefore, the game can be represented with $n = 200$ states with 2 actions (the distribution over which is binomial).

The experiment is run for 2,000 generations, with $N = 10000$. Each experiment is repeated 10 times. Fig. 6.3(a) shows the average reward per generation over each of the 10 executions of CE with various selection methods. As can be seen, policy improvement occurs most rapidly with $\rho = 50\%$, but levels off quite rapidly. It is then surpassed by CS Proportional, which produces the highest quality policies for the rest of the experiment. The distribution of rewards according to strategy is depicted in Fig. 6.3(c), with error bars displaying the standard deviation of the average of the 10 final populations in each experiment. While Proportional CE produces the best policy, the difference between Proportional CE and CE is minimal.

In the second variant tested, the option to *double* is introduced. This action causes the player to double the wager (after which payoffs can be only $-2, 0$, or 2), hit, and then stick. All other details are identical to the first setting, and the dealer is not able to apply this action. The performance of the various CE variants is rendered in Fig. 6.3(b). While in the original variant, CE improved all policies over time, only the proportional strategy resulted in consistent improvement over time when doubling was allowed. Both CE with $\rho = 20\%, 50\%$ initially improved, but later degraded, with $\rho = 50\%$ being essentially equal to

chance performance by the end of the experiment, and all other policies produced by non-proportional selection being worse than chance. As can be seen in the distributions over rewards in Fig. 6.3(d), Proportional CE exercises the double action less than 10% of the time, and has a PDF markedly different from the other strategies. In particular, CE with $\rho = 10\%, 20\%$ both performed the worst, and doubled the most (almost 95% of the time), and lost almost $1/3$ of all bets where doubling was used, resulting in very poor performance.

mCE. We also ran experiments to verify the behavior of mCE in the context of the second variant of Blackjack. As rendered in Fig. 6.3(e) and (f) (for two values of ρ), even with as high as $m = 30$ samples per individual, the quality of the mCE algorithm doesn't match Proportional CE. The performance improves (as compared to CE) as more samples per input are added (up to a point due to the finite size of the population), but it is not clear how to set m and N such that mCE performs at least as good as Proportional CE.

6.8 Summary

The goal of the chapter was to discuss the impact of naturally occurring evaluation noise on the performance of a well known optimization algorithm: the cross-entropy method. We proved that sometimes CE optimizes for a different criterion than the maximum expected value of a function, namely a quantile metric. We proposed an algorithm that has the same structure but optimizes for the correct objective. We also described a variety of naturally occurring optimization problems which determine CE to behave sub-optimally in a way consistent with the theoretical results.

While the new algorithm we introduced, Proportional Cross-Entropy, is

more robust with respect to the optimization criterion, we only proved that it optimizes correctly in a simple, discrete, stochastic optimization setting. Moreover, we have only showed that this is true for a particular family of distributions. Significant more work is needed to extend the theoretical result to other families of distributions and more complex stochastic optimization models.

The key theoretical result is thus a negative one: that when the noise distributions are such that different objectives (like optimizing for expectation and for certain quantiles) don't align, CE will *not* optimize for expectation. We note that this is part of the reason we have extended the noise model we used in the previous chapters to more general noise distributions since for a Bernoulli noise distribution, ordering by expectation is always consistent with the ordering determined by any quantile.

Chapter 7
Conclusions and Future Work

As a general direction of research, I believe the Stochastic Dilemma problem is an essential component in most general stochastic optimization settings. I thus think it is important to both (1) develop algorithmic strategies that are as close as possible to optimal in the Stochastic Dilemma model and (2) apply them to more complex optimization settings to verify whether they provide any benefit in more complex scenarios.

The theoretical and empirical results from this dissertation support the thesis statement: **Noise in function evaluations makes even the simplest optimization problems difficult. Developing correct and efficient techniques for solving stochastic dilemmas has a significant impact on improving the solving of more complex stochastic optimization problems. I introduced novel algorithms for solving such stochastic dilemmas and proved how the analysis and algorithmic ideas can be re-used to solve more general problems.**

In Chapter 2 I introduced the key algorithmic tools used throughout the dissertation and analyzed their performance in the simple Stochastic Dilemma model. The main contribution from this chapter is the definition and the analysis of the Beat-By-K algorithm and the proof about its dominance over non-adaptive strategies like Majority Vote. The second contribution is to gather under the same umbrella similar algorithmic ideas for solving a Stochastic Dilemma problem when the parameter p is unknown. The simple model from

Chapter 2 is generalized in Chapter 3 with the goal of solving realistic, categorical data annotation problems. The empirical results include experiments with crowdsourcing platforms like Amazon Mechanical Turk or Galaxy Zoo.

An interesting open problem from Chapter 2 that is left for future work is to decrease the constant factor gap between the lower and the upper bounds for adaptive strategies for solving a Stochastic Dilemma when the parameter p is known. Another open problem (described in detail in Section 2.4.4) is closing the logarithmic gap between the lower and upper bounds in the setting for which p is unknown. Moreover, as discussed in Section 3.5, the existent algorithms for the unknown p case tend to be very conservative in practice. Concretely, the empirical failure probability for algorithms like Hoeffding Rejection tends to be significantly smaller than the parameter δ. This fact shows that there is room for improvement regarding the definition of the dynamic confidence intervals for the algorithms from Section 2.5.2.

Among the assumptions behind applying Beat-By-K for solving data annotation problems is that we only need to solve binary labeling problems. In practice though, it is often the case that the number of labels is larger than two. Extending Beat-By-K and the other strategies for such settings is left for future work.

While the experiments from Chapter 3 concern only data annotation problems, it is natural to apply algorithms like Majority Vote, Beat-By-K or Hoeffding Rejection to other types of applications for which reducing label noise is relevant. Based on some promising initial results, it appears that Beat-By-K can also act as an alternative baseline algorithm to Majority Vote in supervised machine learning, when the data is noisy.

In Chapter 4 I connected the Stochastic Dilemma problem with the PAC-Bandits model [Even-Dar et al., 2002] and proved how the Greedy Rejection algorithm (which can be viewed as a version of Beat-By-K), is asymptotically optimal in an infinite bandit setting. The result is interesting as it offers an alternative perspective to solving PAC-Bandit problems. The algorithmic ideas from Chapters 2 and 4 are extended and applied to non-trivial stochastic optimization problems in Chapter 5 for designing planning algorithms for computer games.

Similarly to the data annotation setting, Hoeffding Rejection-like bandit algorithms (IHR, GHR, ILHR, GLHR from chapter 5) tend to be conservative in terms of the number of samples required to reach a certain accuracy for finding an (approximately) optimal solution. For applications in which samples are expensive, decreasing the sample complexity of these strategies, while still giving formal guarantees of their performance, is of significant practical interest and is left for future work.

A generalized version of the Hoeffding inequality is the set of Bernstein inequalities [Mnih et al., 2008b], which are concentration of measure results that take into consideration the variance of the sampling distribution. Since in the first four chapters of the thesis, I focused on classes of problems involving the "worst case" scenario for Bernoulli(p) distributions (that is when the parameter p is "close" to 0.5 and the variance is maximized), the two types of bounds lead to identical results. But other classes of problems, for which the parameter p is close to 0 or 1, Bernstein bounds lead to much tighter results as compared to Hoeffding bounds. Such classes of problems are natural for example in ads optimization, where the click-through rate for various ads is in the vast majority of cases very close to 0 (as relatively few people actually click

on advertisements on web pages). It is thus of interest to extend the theoretical results from Chapters 2 and 4 to use these more general bounds.

Another current direction of research is to establish connections with the results from the online multi-armed bandits literature [Auer et al., 2002], which is the setting that is mostly studied in machine learning.

Finally, in Chapter 6, we discuss the properties of the Cross-Entropy Method when applied to solving stochastic optimization problems. I formally show, in an infinite population, multi-armed bandit model, that CE optimizes for quantiles and propose a similar alternative algorithm that optimizes for expectation. The theoretical results are supported by empirical experiments in several non-trivial benchmark domains from operations research and computer science.

One interesting open question remaining from Chapter 6 is to extend the theoretical results from an infinite to a finite population model. It seems intuitive that for a large enough number of samples, CE (and similar algorithms) will have similar properties as in the infinite population setting. Moreover, studying quantile or risk-based optimization in a multi-armed bandit setting is an interesting problem in itself.

Chapter 8

Appendices

Appendix A
Technical Tools

A.1 Hoeffding Inequality

We will first discuss the Hoeffding inequality for Bernoulli random variables. The theorems are given without proof (the proofs are standard and can be found in the referenced articles or in a variety of other sources).

First we will introduce the notation. Let x be a Bernoulli(q) random variable with $q \in (0,1)$ and let $x_i \sim$ Bernoulli$(q), i \in [m]$ be m i.i.d. samples from it and let $\hat{x} = \frac{\sum_{i=1}^{m} x_i}{m}$. The Hoeffding inequality states that for large m, $\hat{x}$ will become concentrated around p with an exponentially small tail (see for example Kearns and Vazirani [1994], Appendix 9.3). Formally:

Theorem A.1.1. *For a fixed parameter $\alpha \in (0,1)$, the following results hold:*

(i) $P(\hat{x} > q + \alpha) \leq e^{-2\alpha^2 m}$

(ii) $P(\hat{x} < q - \alpha) \leq e^{-2\alpha^2 m}$

(iii) $P(\hat{x} \in (q - \alpha, q + \alpha)) \geq 1 - 2e^{-2\alpha^2 m}$

A.2 Random Walks

The goal of this section is to introduce the key random walks technical tools that we use throughout the thesis. We begin by stating a classical result from

random walk theory concerning ruin problems. We will use the notation from Feller [1968] (chapter XIV) and assume a particle is starting at a positive position z on the integer line and at every step it has probability q of shifting its position (with step size 1) to the left and $p = 1 - q$ of shifting its position to the right. We define q_z to be the probability of the particle to eventually be absorbed at 0 and by p_z the probability that the particle is absorbed at $a > z$ (with a also an integer). We also define D_z to be the expected duration of the process until absorption. Then:

Theorem A.2.1 (chapter XIV, sections 2, 3 and 4, Feller [1968])**.** *(i)* $q_z + p_z = 1$ *(in other words the process will stop with probability* 1*).*

(ii) $q_z = \frac{(\frac{q}{p})^a - (\frac{q}{p})^z}{(\frac{q}{p})^a - 1}$.

(iii) $D_z = \frac{z}{q-p} - \frac{a}{q-p}\frac{1-(\frac{q}{p})^z}{1-(\frac{q}{p})^a}$.

We use the result in a slightly different form. Let $X^{(0)} \sim \text{Bernoulli}(p)$ and $X^{(1)} \sim \text{Bernoulli}(1-p), p \in (0, 0.5)$. Let's perform a change in variables and note $Y^{(0)} = 2X^{(0)} - 1$ with $\mathbb{E}[Y^{(0)}] = 2p - 1$ and respectively $Y^{(1)} = 2X^{(1)} - 1$ with $\mathbb{E}[Y^{(1)}] = 1 - 2p$.

If we denote a series $T_0^{(0)} = 0$ and $T_m^{(0)} = \sum_{i=1}^{m} Y_i^{(0)}$ and symmetrically define $T_m^{(1)}$, then $T_m^{(0)}$ is a simple random walk on the integers with unit steps and bias $2p - 1$ (i.e. a negatively biased random walk) while $T_m^{(1)}$ is a positively biased random walk with bias $1 - 2p$. Let's consider two absorbing barriers on opposite sides of the origin with absolute values: $k_0, k_1 \in \mathbb{N}^*$ and let $\beta = \frac{1-p}{p}$. Then the following results hold:

Corollary A.2.2. *If $\mathcal{P}$ is the distribution over steps for a positively biased random walk with bias $1 - 2p$ and $\mathcal{N}$ is the distribution over steps for a negatively biased random walk with bias $2p - 1$ for $p \in (0, 0.5)$, then:*

(i)

$$P_{\mathcal{P}}(T_\tau^{(1)} = k_1) = \frac{\beta^{k_1}(\beta^{k_0} - 1)}{\beta^{k_0+k_1} - 1} \text{ and } P_{\mathcal{P}}(T_\tau^{(1)} = k_0) = \frac{\beta^{k_1} - 1}{\beta^{k_0+k_1} - 1} \quad \text{(A.1)}$$

(ii) If $\tau^{(1)}(k_0, k_1)$ is the expected time until absorption for the positively biased random walk (i.e. the expected value of m such that either $T_m^{(1)} = -k_0$ or $T_m^{(1)} = k_1$) then $\tau^{(1)}(k_0, k_1)$ is finite with probability 1 and:

$$\tau^{(1)}(k_0, k_1) = \frac{\mathbb{E}_P(T_\tau^{(1)})}{1 - 2p} = \frac{k_1\beta^{k_1}(\beta^{k_0} - 1) - k_0(\beta^{k_1} - 1)}{(1 - 2p)(\beta^{k_0+k_1} - 1)} \quad \text{(A.2)}$$

(iii)

$$P_{\mathcal{N}}(T_\tau^{(0)} = k_1) = \frac{\beta^{k_0} - 1}{\beta^{k_0+k_1} - 1} \text{ and } P_{\mathcal{N}}(T_\tau^{(0)} = k_0) = \frac{\beta^{k_0}(\beta^{k_1} - 1)}{\beta^{k_0+k_1} - 1} \quad \text{(A.3)}$$

(iv) In the same manner as (ii):

$$\tau^{(0)}(k_0, k_1) = \frac{k_0\beta^{k_0}(\beta^{k_1} - 1) - k_1(\beta^{k_0} - 1)}{(1 - 2p)(\beta^{k_0+k_1} - 1)} \quad \text{(A.4)}$$

We note that this corollary is also well known in the literature, and we only state it for completeness of presentation and to be able to refer to its parts easily in the rest of the proofs. The proof of the corollary is a simple application of theorem A.2.1 and we will only prove part (i) with the rest following similarly:

Proof. (Part (i)) Let's shift the initial position of the random walk to k_0. Then the original problem of hitting the barriers at $-k_0$ and k_1 is equivalent to hitting barriers at 0 or $k_0 + k_1$ starting at k_0. Then we can apply theorem A.2.1(ii), with $z = k_0$, $a = k_0 + k_1$, $q = p$ and p in A.2.1 being $1 - p$. And we get that $q_z = P_{\mathcal{P}}(T_\tau^{(1)} = k_0) = \frac{(\frac{p}{1-p})^{k_0+k_1} - (\frac{p}{1-p})^{k_0}}{(\frac{p}{1-p})^{k_0+k_1} - 1} = \frac{(\frac{1-p}{p})^{k_1} - 1}{(\frac{1-p}{p})^{k_0+k_1} - 1}$ after simple manipulations, as desired. And we can apply A.2.1(i) to get that $P_{\mathcal{P}}(T_\tau^{(1)} = k_1) = 1 - P_{\mathcal{P}}(T_\tau^{(1)} = k_0)$. □

A.3 Asymptotics With Multiple Complexity Parameters

The goal of this section is to define the notation for situations where we use asymptotic notation with multiple variables. While the definitions for one variable are widely defined and used and the extension for multiple variables is natural, having the formal definitions in clear helps with the understanding of the lower and upper bounds in the thesis that deal with several complexity parameters.

First let's define the usual asymptotic notation for the scenario of functions of only one variable n (see for example Cormen et al. [2001]):

Definition 1. *We define the sets of functions (with C, M constants in all definitions):*

(i) $O(g(n)) = \{f(n) : \exists C, M > 0 \text{ such that } f(n) \leq Cg(n), \forall n \geq M\}$.

(ii) $\Omega(g(n)) = \{f(n) : \exists C, M > 0 \text{ such that } Cg(n) \leq f(n), \forall n \geq M\}$.

(iii) $o(g(n)) = \{f(n) : \forall C > 0, \exists M > 0 \text{ such that } f(n) < Cg(n), \forall n \geq M\}$.

(iv) $\omega(g(n)) = \{f(n) : \forall C > 0, \exists M > 0 \text{ such that } Cg(n) < f(n), \forall n \geq M\}$.

and we abuse notation and write in each case that $f(n) = O(g(n))$ when what is meant is that $f(n) \in O(g(n))$.

Now let's extend the definitions to the scenario where functions depend on multiple variables $n_1, n_2, \ldots, n_k$ for some $k \geq 1$:

Definition 2. *We define the sets of functions (with C, M constants in all definitions):*

(i) $O(g(n_1, \ldots, n_k)) = \{f(n_1, \ldots, n_k) : \exists C, M > 0 \text{ such that } f(n_1, \ldots, n_k) \leq Cg(n_1, \ldots, n_k), \forall n_1, \ldots, n_k \text{ such that } n_i \geq M, \forall i \in [k]\}$.

(i) $\Omega(g(n_1,\ldots,n_k)) = \{f(n_1,\ldots,n_k) : \exists C, M > 0 \textit{ such that } Cg(n_1,\ldots,n_k) \leq f(n_1,\ldots,n_k), \forall n_1,\ldots,n_k \textit{ such that } n_i \geq M, \forall i \in [k]\}$.

(i) $o(g(n_1,\ldots,n_k)) = \{f(n_1,\ldots,n_k) : \forall C > 0, \exists M > 0 \textit{ such that } f(n_1,\ldots,n_k) < Cg(n_1,\ldots,n_k), \forall n_1,\ldots,n_k \textit{ such that } n_i \geq M, \forall i \in [k]\}$.

(i) $\omega(g(n_1,\ldots,n_k)) = \{f(n_1,\ldots,n_k) : \forall C > 0, \exists M > 0 \textit{ such that } Cg(n_1,\ldots,n_k) < f(n_1,\ldots,n_k), \forall n_1,\ldots,n_k \textit{ such that } n_i \geq M, \forall i \in [k]\}$.

The idea of the definition is that a function $f(n_1,\ldots,n_k)$ is part of a set ($O(g(n_1,\ldots,n_k))$ for example) if there is a threshold M such that if all parameters are larger than M, the desired relation holds.

We will use the following theorem several times in the thesis. The result is straightforward, but we give it here for the multiple variable case for completeness of presentation.

Theorem A.3.1. *For a set of complexity parameters $n_1,\ldots,n_k$, for some $k \geq 1$, and two functions $f, g : \mathbb{N}^k \to \mathbb{R}^{\geq 0}$, $f = o(g) \implies f \neq \Omega(g)$.*

Proof. Let's assume $f = \Omega(g)$. Then $\exists C_1, M_1$ s.t. $C_1 g \leq f, \forall n_i \geq M_1$. Since $f = o(g)$, it is also the case that for an arbitrary $C_2, \exists M_{C_2}$ such that $f < C_2 g, \forall n_i \geq M$.

Let's pick $C_2 = \frac{C_1}{2}$ and let $M' = \max\{M_1, M_{C_2=\frac{C_1}{2}}\}$. Then for an arbitrary set of n_i', each with the property that $n_i' \geq M'$, from the two relations we get that $C_1 g(n_1',\ldots,n_k') \leq f(n_1',\ldots,n_k') < \frac{C_1}{2} g(n_1',\ldots,n_k')$ and thus $C_1 < \frac{C_1}{2}$ which is absurd. Hence we got a contradiction with the initial assumption that $f = \Omega(g)$. □

Appendix B
Proofs Chapter 5

B.1 Proof of Theorem 5.4.1

The proof is similar to the proof of theorem 4.5.1.

Proof. We will use contradiction and assume there exists an (ϵ, δ, r_0)-correct algorithm ALG that solves any IB$(\epsilon, \delta, r_0, \mathcal{P})$ problem with expected sample complexity $o(\frac{1}{\epsilon^2}(\frac{1}{\rho} + \log\frac{1}{\delta}))$. The goal is to show that ALG would imply a correct algorithm for the PAC-Bandit problem with expected sample complexity $o(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta}))$, which would contradict the known lower bound in the PAC-Bandit setting.

Let $\mathcal{P}$ be a categorical probability distribution with 2 values in its support: $0.5 - \epsilon$ (a suboptimal arm) and $0.5 + \epsilon$ (an optimal arm) with probability mass $1 - x$ on the first value and x on the second. Let's choose an arbitrary $r_0 \in (0.5, 0.5 + \epsilon]$ (so that we follow the constraint that ρ is bounded away from zero). Then $\rho = x$. Now, define a PAC-Bandit problem as follows: assume we are given n arms, $n - 1$ of which have expected reward of $0.5 - \epsilon$ and one of which has expected reward of $0.5 + \epsilon$. To be precise, it is worth mentioning that we allow the algorithms in the PAC-Bandit setting to resample arms and ignore any previous pulls taken for those arms (this actually makes the PAC-Bandit problem harder, so the lower bound still has to hold).

Let $x = \rho = \frac{1}{n}$. When we use ALG for the PAC-Bandit problem, each

time the algorithm samples a new arm from the environment, it selects an arm uniformly at random, with replacement, from the n arms. Applying *ALG*, it will get the good arm with probability at least $1-\delta$ with an expected number of samples $o(\frac{1}{\epsilon^2}(\frac{1}{\rho}+\log\frac{1}{\delta})) = o(\frac{1}{\epsilon^2}(n+\log\frac{1}{\delta}))$, which contradicts the lower bound from Theorem 13 in Mannor et al. [2004]). □

B.2 Proof of Theorem 5.4.2

Proof. **Sample Complexity**. We will first show that there is a constant lower bound on the probability of the algorithm stopping, which will in turn help us show the expected sample complexity is finite. Let A_i be the event of accepting the i'th sampled arm ($i \in 1, 2, ...$), conditioned on rejecting the first $i-1$ arms. Let N be a random variable that stands for the number of arms sampled until the algorithm returns an arm, and SC be a random variable that stands for the sample complexity. Note that

$$P(A_i) = P(\text{accept arm } a_i | a_i \text{ is 'good'})P(a_i \text{ is 'good'}) + \tag{B.1}$$

$$P(\text{accept arm } a_i | a_i \text{ is 'bad'})P(a_i \text{ is 'bad'}) \tag{B.2}$$

$$\geq P(\text{accept arm } a_i | a_i \text{ is 'good'})P(a_i \text{ is 'good'}) \tag{B.3}$$

$$\geq (1 - \frac{\delta}{2i^2})\rho \tag{B.4}$$

$$\geq \frac{\rho}{2}, \forall i \in 1, 2, \ldots \tag{B.5}$$

(where the third inequality holds due to an application of the Hoeffding inequality). Thus $\mathbb{E}[N] \leq \frac{2}{\rho}$ (by the properties of the geometric distribution, where A_i stands for "success").

The expected sample complexity is

$$\mathbb{E}[SC] = \mathbb{E}[\sum_{i=1}^{N} \frac{4}{\epsilon^2} \log \frac{2i^2}{\delta}] \quad \leq \frac{4}{\epsilon^2} \mathbb{E}[N \log \frac{2N^2}{\delta}] \leq \frac{8}{\epsilon^2} (\mathbb{E}[N \log N] + \frac{1}{\rho} \log \frac{2}{\delta}) \tag{B.6}$$

(with the right hand side of the first equality determined by the Hoeffding inequality). The expectation for sample complexity is taken with respect to both sampling the arms from $\mathcal{P}$ and noise in the pulls themselves.

In a similar manner to the proof of 4.4.5 it can be shown that $\mathbb{E}[N \log N] \leq \mathbb{E}[N] \log \mathbb{E}[N] \leq \frac{2}{\rho} \log \frac{2}{\rho}$ and the desired result follows.

Correctness. The algorithm will stop (with probability 1, since its expected sample complexity is finite) and recommend either a 'good' arm (with reward $r \geq r_0 - \epsilon$) or a 'bad' one (reward $r < r_0 - \epsilon$). The failure probability is $P(\text{failure}) \leq P(\bigcup_{i \geq 1} \{\text{incorrect recommendation at step } i\}) \leq \sum_{i \geq 1} \frac{\delta}{2i^2} \leq \delta$ (the second inequality follows via the Hoeffding inequality given the number of samples $n_0(i)$ for each arm). □

B.3 Proof of Theorem 5.4.3

Proof. **Sample Complexity**. We keep the same notation as in the proof of Theorem 5.4.2. We use r_{a_i} to represent the expected value associated with arm a_i, $\hat{r}_{a_i,j}$ the empirical average of a_i's rewards after its jth pull, and $CI(j) = \sqrt{\frac{2 \log(2j^2/\delta_0)}{j}}$ the confidence interval for the empirical average at step j. Let

$n_{\max}(i) = \frac{4}{\epsilon^2} \log \frac{1}{\epsilon\delta_0}$ be the maximum number of pulls for arm a_i. Now:

$$P(A_i) = \rho(1 - P(\text{reject arm } a_i | \text{arm } a_i \text{ is good})) \tag{B.7}$$

$$= \rho(1 - P(\cup_{j=1}^{n_{\max}(i)} \{\hat{r}_{a_i,j} \notin [r_{a_i} - CI(j), r_{a_i} + CI(j)]\})) \tag{B.8}$$

$$\geq \rho(1 - \sum_{j=1}^{n_{\max}(i)} \frac{\delta_0}{2j^2}) \tag{B.9}$$

$$\geq \rho(1 - \delta_0) \tag{B.10}$$

$$\geq \frac{\rho}{2} \tag{B.11}$$

So, as in Theorem 5.4.2, $\mathbb{E}[N] \leq \frac{2}{\rho}$. Since the sampling of each arm stops after at most $n_{max}(i)$ steps, $\mathbb{E}[SC] \leq \mathbb{E}[\sum_{i=1}^{N} \frac{4}{\epsilon^2} \log \frac{2i^2}{\epsilon\delta}]$ and then the sample complexity bound follows similarly to the proof of Theorem 5.4.2.

Correctness. The algorithm stops with probability 1 in finite time, and

$$P(\text{failure}) \leq \sum_{i \geq 1} P(\{\text{incorrect recommendation at step } i\}) \tag{B.12}$$

$$= \sum_{i \geq 1} P(\bigcup_{j=1}^{n_{\max}(i)} \{\hat{r}_{a_i,j} \notin [r_{a_i} - CI(j), r_{a_i} + CI(j)]\}) \tag{B.13}$$

$$\leq \sum_{i \geq 1} \sum_{j \geq 1} \frac{\delta}{4i^2 j^2} \tag{B.14}$$

$$\leq \delta. \tag{B.15}$$

□

B.4 Proof of Theorem 5.4.4

Proof. Since this is a high probability statement, we can assume for the rest of the proof that we are in a situation where the algorithm commits no errors (which happens w.p. at least $1 - \delta$ as it can be shown that, for the entire experiment, the algorithm fails w.p. at most δ). Let's define $SC(a)$ to be the same

complexity of accepting or rejecting an arm a. Let's fix (for now) the total number of sampled arms to $N = n$, and fix an arm a, with $\Delta_a < 0$ (that we label as a 'bad' arm).

Then, using the Hoeffding inequality, $SC(a) = \frac{4}{\max(\epsilon^2, \Delta_a^2)} \log \frac{2i^2}{\max(\epsilon, \Delta_a)\delta}$ (where i is the index of the arm among all n arms). Let's assume $\mathcal{P}$ is continuous (the discrete case is similar) and let's define $f(\Delta_a)$ to be the pdf of Δ_a. Then, since $r_a \sim \mathcal{P}$:

$$\mathbb{E}[SC(a)|a \text{ 'bad'}] = \int_{\Delta_a<0} \frac{4}{\max(\epsilon^2, \Delta_a^2)} \log \frac{2i^2}{\max(\epsilon, \Delta_a)\delta} f(\Delta_a) d\Delta_a \tag{B.16}$$

$$\leq \log \frac{2n^2}{\epsilon\delta} \int_{\Delta_a<0} \frac{4}{\max(\epsilon^2, \Delta_a^2)} f(\Delta_a) d\Delta_a \tag{B.17}$$

$$\leq \frac{4}{\Delta_-^2} \log \frac{2n^2}{\epsilon\delta} \tag{B.18}$$

So:

$$\mathbb{E}[\text{SC from 'bad' arms}|N = n] = \sum_{k=1}^{n} \mathbb{E}[\text{SC from } k \text{ 'bad' arms}|N = n] \tag{B.19}$$

$$P(k \text{ arms are bad}) \tag{B.20}$$

$$\leq \frac{4}{\Delta_-^2} \log \frac{2n^2}{\epsilon\delta} \sum_{k=1}^{n} kP(k \text{ arms are bad}) \tag{B.21}$$

$$= \frac{4(1-\rho)n}{\Delta_-^2} \log \frac{2n^2}{\epsilon\delta} \tag{B.22}$$

Similarly, it can be shown that:

$$\mathbb{E}[\text{samples from all 'good' arms}|N = n] \leq \frac{4\rho n}{\epsilon^2} \log \frac{2n^2}{\epsilon\delta} \tag{B.23}$$

Then, for any N:

$$EB = \mathbb{E}[\text{samples from all 'bad' arms}] \tag{B.24}$$

$$\leq \sum_{n=1}^{\infty} \frac{4(1-\rho)n}{\Delta_-^2} \log \frac{2n^2}{\epsilon\delta} P(N=n) \tag{B.25}$$

$$\leq \frac{4(1-\rho)}{\Delta_-^2} \log \frac{2}{\epsilon\delta} \mathbb{E}[N] + \frac{8(1-\rho)}{\Delta_-^2} \mathbb{E}[N \log(N)] \tag{B.26}$$

$$\leq \frac{16(1-\rho)}{\rho\Delta_-^2} \log \frac{4}{\epsilon\rho\delta} \tag{B.27}$$

(where the last inequality follows from the bounds for $\mathbb{E}[N]$ and $\mathbb{E}[N\log(N)]$ from Theorem 5.4.2). So, $EB = O(\frac{1}{\rho\Delta_-^2}\log(\frac{1}{\epsilon\rho\delta}))$. Using a similar argument, one can show that $EG = \mathbb{E}[\text{samples from all 'good' arms}] = O(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon\rho\delta}))$. Thus, $\mathbb{E}[SC] = EB + EG = O((\frac{1}{\epsilon^2} + \frac{1}{\rho\Delta_-^2})\log\frac{1}{\epsilon\rho\delta})$. □

B.5 Proof of Theorem 5.4.5

Before we give the actual proof we will restate for completeness (and to unify notation) Corollary 2.3 from Kallenberg [1999].

Theorem B.5.1. *[Kallenberg [1999]] Let $(Z_1, Z_2, ...)$ a finite or infinite, stationary sequence of random variables with values in $\mathbb{R}_+ = [0,\infty]$ and let $T_j = \sum_{i\leq j} Z_i$ and $\beta = E[Z_1]$. Then there exists a random variable σ, uniform over $(0,1)$ (and independent of Z_i) such that:*

$$P_{\sigma,Z_i,i\geq 1}[\sup_{j>0} \frac{T_j}{j} \leq \frac{\beta}{\sigma}] = 1.$$

Now we can prove theorem 5.4.5.

Proof. We use the notation from Theorem 5.4.2 and we will only discuss the sample complexity (the correctness follows similarly to the other algorithms). As mentioned, the main challenge, given the aggressiveness of the rejection

procedure, is to get a positive lower bound on the probability of accepting a good arm. Let B be the event of accepting an arm if the expected reward associated with that arm is r_0 (the bound follows immediately for all arms with $r \geq r_0$, since the probability of acceptance will be at least as large as for r_0). Define $X = \text{Bernoulli}(r_0)$.

Define $Y = \frac{X - r_0 + \epsilon/2}{1 - r_0 + \epsilon/2}$ as an affine transformation of X. Then, let $\alpha = E[Y] = \frac{\epsilon/2}{1 - r_0 + \epsilon/2} \geq \frac{\epsilon}{2}$ (since $r_0 > \epsilon$). Since $Y = 1$ with probability r_0 and $Y = \frac{-r_0 + \epsilon/2}{1 - r_0 + \epsilon/2} < 0$ with probability $1 - r_0$, we can interpret the series $\{T_j\}$ (with $T_j = \sum_{i=1}^{j} Y_i$, with Y_i being i.i.d. samples of Y, and implicitly X_i being i.i.d. samples of X) as a random walk.

We will now make two simplifying assumptions (and then describe at the end of the proof how to remove them). We assume: (1) $r_0 - \frac{\epsilon}{2} \geq \frac{1}{2}$ and (2) $\frac{-r_0 + \epsilon/2}{1 - r_0 + \epsilon/2} \in \mathbb{Z}^-$ (the set of negative integers). Then, $\{T_j\}$ is a positively biased random walk on the integers with maximum step value 1. In this case, we can apply a classic ballot-style result that says that $P(T_j > 0, \forall j = 1, 2, ...) = \max(\mathbb{E}[Y], 0) = \alpha$, for example, Theorem 3 from Addario-Berry and Reed [2008], which is based on a result by Takacs [1967]. But:

$$\alpha = P(T_j > 0, \forall j = 1, 2, ...) \leq P(T_j \geq 0, \forall j = 1, 2, ...) \tag{B.28}$$

$$= P(\frac{\sum_{i=1}^{j} X_i}{j} \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, \ldots) \tag{B.29}$$

$$= P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ...) \tag{B.30}$$

$$\leq P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ..., n_{\max}(i)) \tag{B.31}$$

(where $\hat{X}_j$ is the empirical average after j samples and corresponds to $\hat{r}_{a_i,j}$ from the description of the algorithm). Thus, $P(B) \geq \alpha \geq \frac{\epsilon}{2}$.

So, as in Theorem 5.4.2, $P(A_i) \geq P(B)\rho \geq \frac{\epsilon\rho}{2}$. This fact implies $\mathbb{E}[N] \leq \frac{2}{\epsilon\rho}$ and the proof for the expected sample omplexity bound follows similarly to

that of Theorem 5.4.3 with an extra $\frac{2}{\epsilon}$ factor in the bound that comes via the upper bound on $\mathbb{E}[N]$.

To complete the proof, one needs to show that a 'bad' arm (with expected value smaller than $r_0 - \epsilon$) will be rejected after at most $n_{\max}(i)$ samples. This claim follows via the same application of the Hoeffding inequality as for the other algorithms (the probability that a 'bad' arm is accepted after $n_{\max}(i)$ samples is smaller than δ_0, which is enough to bound the probability of error for the entire execution of the algorithm).

To remove Assumptions 1 and 2, we will apply ballot-style theorems for random variables with real values. The result will then follow by applying Corollary 2.3 from Kallenberg [1999].

The goal of the last part of the proof is to complete the proof for GHR for the general case of random walks on the real numbers.

Let $Z = 1 - Y$. But $(Z_1, Z_2, ...)$ is a stationary sequence of variables with $\mathbb{E}[Z_1] = \mathbb{E}[Z] = 1 - \mathbb{E}[Y] = 1 - \alpha$. The support of Z is $\{0, \frac{1}{1-r_0+\frac{\epsilon}{2}}\} \subset \mathbb{R}_+$.

Let $R_j = \sum_{i \leq j} Z_i$. Then, the conditions for theorem B.5.1 hold and so there exists a Uniform$(0,1)$ random variable σ such that $P[\sup_{j>0} \frac{R_j}{j} \leq \frac{1-\alpha}{\sigma}] = 1$.

Let's note $V = \sup_{j>0} \frac{R_j}{j}$ and $W = \frac{1-\alpha}{\sigma}$ two transformed random variables of Z_i and σ respectively. We know that $P[V \leq W] = 1$ (a relation known under the name of absolute stochastic dominance or statewise stochastic dominance). It is known that this relation implies the usual notion of (first order) stochastic dominance (which states that a random variable Y stochastically dominates a random variable X if for all elements x in the support of X and Y, $P(Y > x) \geq P(X > x)$ or equivalently $P(Y \leq x) \leq P(X \leq x)$).

So since W stochastically dominates V in an absolute sense, it also dominates it in a first-order sense. We will pick $1 \in \mathbb{R}_+$ and it follows that $P(W \leq$

$1) \leq P(V \leq 1)$.

But $P(W \leq 1) = P_\sigma(\frac{1-\alpha}{\sigma} \leq 1) = P_\sigma(\sigma \geq 1-\alpha) = 1-(1-\alpha) = \alpha$ (where the third equality follows immediately from the properties of the uniform distribution).

We have thus shown that $\alpha \leq P(V \leq 1)$. Then $\alpha \leq P(\sup_{j>0} \frac{R_j}{j} \leq 1) = P(\frac{R_j}{j} \leq 1, \forall j > 0) = P(\sum_{i \leq j}(1-Y_i) \leq j, \forall j > 0) = P(T_j \geq 0, \forall j > 0)$. But we know that $P(T_j \geq 0, \forall j > 0) \leq P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ..., n_{\max}(i))$. Thus the desired relation ($P(B) \geq \alpha$) holds even when Assumptions 1 and 2 are removed, and we consider random walks with steps taking real values. The rest of the proof remains unchanged. □

References

L. Addario-Berry and B. A. Reed. Ballot theorems, old and new. In *Horizons of Combinatorics*. Springer Berlin Heidelberg, 2008. 93, 106, 118, 125, 177

G. Alon, D. P. Kroese, T. Raviv, and R. Rubinstein. Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals Operations Research*, 134(1):137–151, 2005. 133

M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009. 9, 18

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*. 2010. 77

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *ML*, 2002. 4, 76, 164

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 2003. 2

R. Bardenet and B. Kégl. Surrogating the surrogate: accelerating gaussian-process-based global optimization with a mixture cross-entropy algorithm. In *International Conference on Machine learning*. 2010. 133

Jonathan Baxter. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12:149–198, 2000. 9

Shai Ben-David and Michael Lindenbaum. Learning distributions by their density levels. In *Computational Learning Theory: Second European Conference, EuroCOLT'95, Barcelona, Spain, March 13-15, 1995. Proceedings*, page 53. Springer. 9

Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical report, MIT, 1996. 139, 155

P. Boer, D.P. Kroese, S. Mannor, and R. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005. 133, 135, 136, 147

Sébastien Bubeck and Rémi Munos. Open loop optimistic planning. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*. 2010. 129

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *ALT*. 2009. 77

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. Online optimization in x-armed bandits. In *NIPS*. 2008. 76, 129

Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995. 9

Bob Carpenter. Multilevel bayesian models of categorical data annotation. available at http://lingpipe-blog.com/ lingpipe-white-papers. 2008. 44, 48, 50

H.S. Chang, M.C. Fu, J. Hu, and S.I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Springer-Verlag New York, Inc., 2007. 133, 134, 138

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001. 169

A. Costa, O.W. Jones, and D. Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573–580, 2007. 132, 133

Peng Dai, Mausam, and Daniel S. Weld. Artificial intelligence for artificial artificial intelligence. In *AAAI*. 2011. 50

DA Darling and Herbert Robbins. Inequalities for the sequence of sample means. *Proceedings of the National Academy of Sciences of the United States of America*, 57(6):1577, 1967a. 35, 36, 37

DA Darling and Herbert Robbins. Iterated logarithm inequalities. *Proceedings of the National Academy of Sciences of the United States of America*, 57(5):1188, 1967b. 35, 36

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 1979. 50

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML*. 2008. 126

Pinar Donmez and Jaime G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM*. 2008. 50

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*. 2009. 51

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *COLT*. 2002. xiii, 4, 6, 10, 17, 26, 27, 28, 32, 76, 77, 79, 82, 103, 107, 108, 109, 111, 118, 119, 163

William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968. 167

J. Michael Fitzpatrick and John J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 1988. 124, 130

S. Goschin, M.L. Littman, and D.H. Ackley. The effects of selection on noisy fitness optimization. In *Genetic and Evolutionary Computation Conference (GECCO)*. 2011. iv, 131, 153

Sergiu Goschin, Ari Weinstein, and Michael Littman. The cross-entropy method optimizes for quantiles. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1193–1201. 2013. iv

Sergiu Goschin, Ari Weinstein, Michael L Littman, and Erick Chastain. Planning in reward-rich domains via pac bandits. *JMLR, W&C Proceedings EWRL*, pages 25–42, 2012. iv

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolution Computation*, 9(2):159–195, 2001. 139

Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Technical Report 6869, INRIA, 2009. 131

Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML*. 2009. 27, 118, 130

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *HCOMP*. 2010. 50

JStella. Jstella project, atari 2600. 2008. URL `http://jstella.sourceforge.net/`. 126

Olav Kallenberg. Ballot theorems and sojourn laws for stationary processes. *The Annals of Probability*, 1999. 176, 178

S. Kalyanakrishnan and P. Stone. An empirical analysis of value function-based and policy search reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. 2009. 133

Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*. 2012. 51

Michael Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. The MIT Press, 1994. 8, 9, 166

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *ACM Aymposium on Theory of Computing*. 2008. 76, 129

M. Kobilarov. Cross-entropy randomized motion planning. In *Robotics: Science and Systems*. 2011. 133

Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324. 1998. 67

Omid Madani, Daniel J. Lizotte, and Russell Greiner. Budgeted learning, part 1: The multi-armed bandit case. Technical report, University of Alberta, 2003. 77, 124

S. Mannor, D. Peleg, and R. Rubinstein. The cross entropy method for classification. In *International Conference on Machine learning*. 2005. 132, 133

S. Mannor, R. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *International Conference on Machine Learning*. 2003. 132, 137, 153, 154

Shie Mannor, John N. Tsitsiklis, Kristin Bennett, and Nicol Cesa-Bianchi. The sample complexity of exploration in the multi-armed bandit problem. *JMLR*, 2004. xiii, 10, 18, 19, 86, 88, 103, 107, 108, 110, 172

L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134:201–214, 2005. 132, 133

Oded Maron and Andrew Moore. The racing algorithm: Model selection for lazy learners. In *Artificial Intelligence Review*. 1997. 7, 10, 27, 118, 121

Brad L. Miller and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995. 131

Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *Proceedings of the 25th international conference on Machine learning*, pages 672–679. ACM, 2008a. 27

Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *ICML*. 2008b. 123, 163

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11, 2010. 50

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952. 4

Herbert Robbins. Statistical methods related to the law of the iterated logarithm. *The Annals of Mathematical Statistics*, 41(5):1397–1409, 1970. 10, 39

R. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1996. 131, 136

R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability*, 1:127–190, 1999. 131

R. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, 2004. 133

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD KDD*, KDD '08. 2008. 9, 13, 44, 51

Hans Ulrich Simon. General bounds on the number of examples needed for learning probabilistic concepts. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 402–411. ACM, 1993. 9

Padhraic Smyth, Usama M. Fayyad, Michael C. Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. In *NIPS*. 1994. 44, 50, 51

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*. 2008. 50

F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. In *International Conference on Machine learning*. 2012. 133, 135, 139

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 158

I. Szita and A. Lörincz. Learning Tetris using the noisy cross-entropy method. *Neural Computation*, 18(12):2936–2941, 2006. 133, 136, 155

I. Szita and C. Szepesvári. SZ-Tetris as a benchmark for studying key problems of reinforcement learning. In *ICML 2010 Workshop on Machine Learning and Games*. 2010a. 133, 139, 155, 156

I. Szita and C. Szepesvári. sztetris-rl Library. `http://code.google.com/p/sztetris-rl/`, 2010b. 155

Lajos Takacs. *Combinatorial methods in the theory of stochastic processes*. Wiley New York, 1967. 177

Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario AI competition. In *IEEE CEC 2010*. 2010. 116

M.D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1998a. 132

Michael D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1998b. ISBN 026222058X. 142

Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. In *NIPS*. 2008. 76, 112

Fabian L. Wauthier and Michael I. Jordan. Bayesian bias mitigation for crowdsourcing. In *NIPS*. 2011. 50

Jacob Whitehill, Paul Ruvolo, Ting fan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*. 2009. 50

Shimon Whiteson, Brian Tanner, and Adam White. The reinforcement learning competitions. *AI Magazine*, 2010. 116

Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013. 6, 69, 70

Yan Yan, Romer Rosales, Glenn Fung, Mark Schmidt, Gerardo Hermosillo, Luca Bogoni, Linda Moy, and Jennifer Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. 2010. 50, 51

Liu Yang and Jaime Carbonell. Adaptive proactive learning with cost-reliability tradeoff. Technical report, CMU, 2009. 13