The Analysis of Adaptive Data Collection Methods for Machine Learning

By

Kevin Jamieson

A dissertation submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN – MADISON

2015

Date of final oral examination: May 7, 2015

The dissertation is approved by the following members of the Final Oral Committee:
Robert Nowak, Electrical and Computer Engineering, UW - Madison
Ben Recht, Electrical Engineering and Computer Sciences, UC Berkeley
Rebecca Willett, Electrical and Computer Engineering, UW - Madison
Stephen J. Wright, Computer Sciences, UW - Madison
Xiaojin (Jerry) Zhu, Computer Sciences, UW - Madison
Jordan S. Ellenberg, Mathematics, UW - Madison

Abstract

Over the last decade the machine learning community has watched the size and complexity of datasets grow at an exponential rate, with some describing the phenomenon as *big data*. There are two main bottlenecks for the performance of machine learning methods: computational resources and the amount of labelled data, often provided by a human expert. Advances in distributed computing and the advent of cloud computing platforms has turned computational resources into a commodity and the price has predictably dropped precipitously. But the human response time has remained constant: the time it will take a human to answer a question tomorrow is the same amount of time it takes today, but tomorrow it will cost more due to rising wages world-wide. This thesis proposes a simple solution: require fewer labels by asking better questions.

One way to ask better questions is to make the data collection procedure adaptive so that the question that is asked next depends on all the information gathered up to the current time. Popular examples of adaptive data collection procedures include the *20 questions game* or simply the *binary search* algorithm. We will investigate several examples of adaptive data collection methods and for each we will be interested in answering questions like, how many queries are sufficient for a particular algorithm to achieve a desired prediction error? How many queries must *any* algorithm necessarily ask to achieve a desired prediction error? What are the fundamental quantities that characterize the difficulty of a particular problem?

This thesis focuses on scenarios where the answers to queries are provided by a human. Humans are much more comfortable offering qualitative statements in practice like “this

color is more blue than purple" rather than offering the quantitative RGB values of a particular hue, and the algorithms in this thesis take these kinds of considerations into account. Part I of this thesis considers the identification of a ranking or set of rankings over a set of items by sequentially and adaptively querying an oracle for pairwise comparisons like "do you prefer A or B?" We characterize the difficulty of these problems using geometrical arguments and show that adaptively selecting pairwise comparisons can drastically reduce the number of questions that must be asked relative to the standard non-adaptive method.

In part II we consider a multi-armed bandits framework that allows us to dive deep into subtle effects of adaptive data collection when the data is corrupted in some way, perhaps by some stochastic adversary. In these idealized settings we identify fundamental quantities that almost completely characterize the difficulty of these problems and propose algorithms that are nearly optimal with respect to these fundamental quantities. Part III builds off of the advances of Part II and applies the techniques to scenarios that involve pairwise comparisons like those used in Part I. Namely, Part III considers combinatorial and continuous optimization with only pairwise comparison feedback.

Acknowledgements

There are many people over the years that have helped me reach this point. Some have given me life-changing opportunities, some have given me encouragement and support, and others have simply had short conversations that stuck with me for years. It is impossible to thank them all, but I would like to acknowledge a special few who have contributed so much.

It is unlikely that I would have attended graduate school if it were not for Austin Miller and Maya Gupta. Austin helped land me an internship at an aerospace company where, under his mentorship, I first learned about Kalman filters and why statistics was, in fact, insanely cool. I remember the day Austin convinced me to consider graduate school over lunch, I suppose I owe him some sushi. Maya was my undergraduate research advisor at the University of Washington and is responsible for teaching me how to do academic research and how to write a paper. Her guidance while performing research, applying to graduate school, and navigating graduate school was invaluable. Robert Miyamoto and David Krout were also paramount to my research at the Applied Physics Lab at the University of Washington. Rui Castro acted as my advisor at Columbia University during my masters and was the first person to introduce me to active learning. He taught me what it meant to be mathematically rigorous and how to prove a theorem for which I am forever grateful.

No one deserves more acknowledgement for my success other than my PhD advisor at the University of Wisconsin - Madison, Robert Nowak. From the first day we met in person over beer at the Memorial Terrace to discuss research directions, I knew I had

found my ideal advisor. Rob was hands off but always willing to meet at the whiteboard to get me out of a jam. He taught me the power of intuition and how to see the proof before writing a single line. He showed me how to tell a story both in my writing and presentations, his own style being an example that I strive towards to this day. Rob encouraged me to follow any direction I found interesting and above all, what made me happy, even if that meant moving 2,000 miles across the country for the last year of my thesis. And Rob's advising did not stop at academia. He was a business partner who encouraged me to create a beer recommendation app that we eventually sold to a startup. He was my personal trainer who would tear me away from work in the middle of the day to go on a bike ride with him. He convinced me to take up cross-country skiing and even got me into the ice hockey rink a few times. And finally, Rob is my friend. I cannot thank him enough.

Ben Recht also played a major role in my graduate career by advising me in all things optimization and was also invaluable in teaching me how to recognize an interesting problem. I thank Ben for encouraging me to deliberately work on the boundary of practical and theoretical research. Ben generously opened doors for me to the community and at the University of California Berkeley where I was introduced to collaborators and experiences that have significantly changed my view of research and my place in it. Other faculty also significantly contributed to my success at the University of Wisconsin. Tim Rogers from the psychology department showed me the satisfaction of interdisciplinary research and solving real problems. Sebastien Bubeck from Princeton University showed me the beautiful ideal of multi-armed bandits. Finally, I'd like to thank Jerry Zhu, Jordan Ellenberg, Rebecca Willett, and Steve Wright for all their advice and thoughtful feedback over the years, and also for agreeing to sit on my committee.

The work presented in this thesis would not have been possible without my brilliant collaborators Robert Nowak, Ben Recht, Sebastien Bubeck, Matt Malloy, Summet Katariya, Atul Deshpande, and Ameet Talwalkar. I also benefitted greatly from the discussions and feedback from my close colleagues and friends Laura Balzano, Aniruddha Bhargava, Gautam Dasarathy, Summet Katariya, Shirzad Malekpour, Matt Malloy, Nikhil Rao, Yana Shkel, Leslie Watkins and many more outside the lab to name. In particular I would like to thank Victor Bittorf, Srikrishna Sridhar, and Badri Bhaskar for all their help in teaching me how to write better code. While not discussed in this thesis, the NEXT project would not have been possible without the dedicated work of Ari Biswas, Chris Fernandez, Nick Glattard, and Lalit Jain. Finally, I must thank the ECE administrator Daryl Haessig who has pulled my feet out of the fire more times than I care to admit.

I would also like to thank my parents Linda and Frank for their unwavering support and encouragement through not only my graduate career but all the intermediate steps that led me here. I'd also like to thank my sisters Lauren and Jana who were always supportive. Finally I'd like to thank Sarah Rich for her support, advice and persistent ability to make me happy.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Some of the most advanced examples of adaptive data collection methods go unnoticed as regular human social interaction. As an example, consider the bar patron who enjoyed a beer at the bar last night, forgot the name of the favored beer, and returned to the bar to discover its name. The bartender agreed to help identify the favored beer among the 40 beers on tap by presenting the patron with a sequence of samples of the beers, two at a time, and asked the patron to identify which of the two beers was more similar to the favored beer. After just several questions of this type, the bartender had discovered the patron's beer.

The story of the bartender and patron is remarkable because the number of questions used to find the favored beer was much less than the 40 possible beers on tap. Clearly, if the bartender had asked the patron to try a randomly selected sequence of beers, then one would expect that the patron would have to try something very close to all 40 beers before his beer was found. The only way the bartender could find the patron's beer so quickly, assuming it was not just by blind luck, was if the bartender was exploiting some structure about the beer. Perhaps by realizing that the patron's answers were suggesting the favored beer was similar to a wheat beer, the bar tender could eliminate all queries involving stouts or dark beers on tap as possibilities for the favored beer, for instance.

The bartender is a perfect example of an adaptive data collection algorithm, and

the story raises many questions that go beyond this one example. What exactly is the structure being exploited by the bartender? Is the structure inherent to the beers themselves, or is it coupled with the patron's and bartender's model of how beers relate? How many questions would the bartender have had to ask if there were 100 beers on tap, rather than just 40? If it were not beer but wine, or music, or movies, how would the necessary number of questions to identify the patron's favored item change? That is, what characterizes the fundamental difficulty of this problem? If we can characterize how hard a problem is and why, can we design algorithms that can provably perform close to this fundamental speed limit?

In this thesis, we consider several examples of adaptive data collection like this one and try to answer these kinds of questions and focus on the fundamental quantities of the problems. The tools used to analyze these problems come from many disciplines. We will draw some motivation from psychometrics: what are the best ways to extract information from fallible humans? Statistical learning theory will allow us to confidently discard invalid hypotheses and confirm valid ones. Information theory will allow us to characterize the fundamental difficulty of problems. Convex analysis and optimization will allow us to make powerful statements about the rate at which our algorithms learn. And the multi-armed bandit framework will provide us with a powerful abstraction giving us the ability to generalize our results to many domains.

Adaptive data collection is an umbrella term for many sub-disciplines, all with with their own slightly different terminology. In this thesis, we will take the terms *adaptive data collection* and *adaptive learning* to be synonymous with each other. In the computer sciences, adaptive learning is often labelled under the name *active learning* whereas in electrical engineering and statistics, adaptive learning is sometimes labelled as *adaptive*

sampling or *adaptive sensing*. While one may argue that there are subtle differences between these terms based largely on historical context, for the purposes of this thesis we will treat all of these terms as synonymous.

1.1 The Query Complexity of Learning

Adaptive learning can be thought of as a game between two players: a player (taking the form of an algorithm) and an oracle (perhaps taking the form of a human or stochastic process). The game proceeds in rounds where at the beginning of the game the oracle selects some fixed, hidden hypothesis $h_* \in \mathcal{H}$ that is unknown to the player. Then at each round t the player chooses a query (or takes some action) $q_t \in \mathcal{Q}$, the oracle responds to q_t with a response $y_t \in \mathcal{Y}$, and then the game proceeds to the next round, where $\mathcal{H}, \mathcal{Q}, \mathcal{Y}$ are all possibly uncountable sets. The objective of the player is to identify $h_* \in \mathcal{H}$ (or perhaps an $h \in \mathcal{H}$ that is "close" to h_* is some well-defined sense) using as few queries as possible, perhaps in expectation or with high probability[1]. We define the *query complexity* of a problem to be the minimum number of queries that the player, using the best possible strategy, must make to the oracle in order to identify $h_* \in \mathcal{H}$ (or a sufficiently "close" h, perhaps with high probability). With this definition, one can talk about a lower bound on the query complexity of a problem which would say that no algorithm can identify h_* using fewer queries than the claimed lower bound. On the other hand, any algorithm that identifies $h_* \in \mathcal{H}$ with some number of queries is a valid upper bound on the query complexity. The goal of this thesis is to identify interesting problems and algorithms such that we can prove nearly matching lower and

[1]Stochasticity could be introduced to the process if the oracle responses are stochastic or if the algorithm itself is random.

upper bounds on the query complexity of a problem. Throughout this thesis we will alternate between "queries" and "samples" when one or the other is more appropriate, thus one should consider query complexity and sample complexity to be synonymous.

1.2 Learning with Comparative Judgments

Much of the work within in this thesis is motivated by applications that rely directly on human feedback. Therefore, it makes sense to consider the kinds of questions that are best suited for collecting data from humans. We saw above how comparative judgments, or pairwise comparisons, were used successfully by the bartender to identify the patron's favored beer. Another example of when pairwise comparisons are used is when the optometrist identifies suitable prescription lenses for a patient using just "better or worse?" feedback. Pairwise comparisons are convenient when the value of a stimuli (e.g. the goodness of fit of a given prescription lens) is difficult to quantify. For instance, one may find that rating the image of the left pane of Figure 1 on a scale of 1-5 in terms of how safe it appears to be much more difficult to do than if shown the left and right panes of Figure 1 together and asked, "which street view is more safe?". Such an approach using pairwise comparisons was recently explored in [1] to rank the images in the corpus from safest to most dangerous to compare crime statistics with perceived danger just by the appearance of the neighborhood at street level. The underlying premise is that human analysts can answer such a question much more robustly and consistently that they can apply more traditional labeling mechanisms, such as assigning a numerical safety rating to the image.

There is significant evidence to support the idea that pairwise comparisons can be

more informative than asking humans for quantitative scores. Indeed, across a variety of different pairwise discriminable stimuli (e.g. hues of color or tonal frequencies) human subjects have been found to only be able to reliably communicate around 3 bits of information about the perceived value of a stimulus over time despite perfectly answering queries of "which was it most similar to, A or B?" [2]. In addition, it has been shown that pairwise comparisons are more robustly recalled in the future compared to quantitative scores that may change over time if only because a lack of "anchors" in the space [3]. These studies suggest that robust, precise quantitative information can be gathered more efficiently through asking qualitative relative comparisons rather than asking for quantitative scores. Pairwise comparisons also have the benefit of not suffering from calibration over time or between participants: we may have the same preferences over movies but while I am more liberal with my scores using the full 1-5 scale, you might avoid giving very lower scores and just use stars in the 2-5 range.

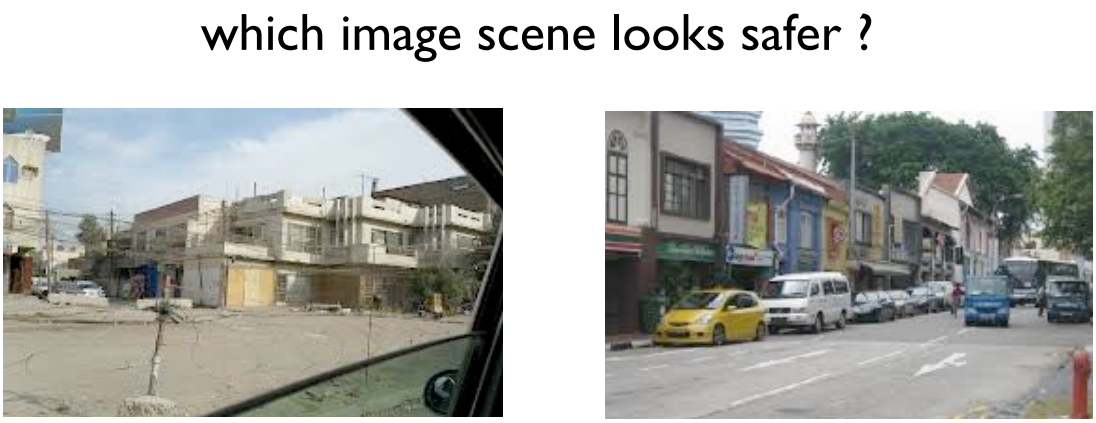


Figure 1.1: Asking humans "how safe is the scene on the left on a scale of one to five?" may feel much more difficult than simply asking "which scene is more safe, left or right?" Research also shows that comparative judgments are more robust and avoid calibration issues that arise in requesting scores from humans.

Finally, pairwise comparisons can admit a geometric interpretation in the domain

space that can be used to more easily determine a relationship between the internal beliefs of a human subject and their answered responses,, thereby making it easier to determine which queries may be most informative. To understand the importance of this last point, suppose I rate two movies a rating of 2 and 4 respectively. Does that mean that I like the second movie twice as much as the first? If I rate a third movie a 5, does that mean that the degree to which I prefer a score of 5 to 4 is less than my preference of a 4 to 2? Depending on how these questions are answered, one may need to encode this information into the algorithm's model which can lead to a possibly brittle and special purpose algorithm. However, these problems do not exist when requesting pairwise comparisons.

There are also some downsides to using pairwise comparisons. First, per query, a pairwise comparison admits at most 1-bit of information whereas other kinds of queries may provide more information (i.e. providing someone with $2^3 = 8$ options to choose from may provide up to 3 bits of information per query). The consequences of this issue is evident when trying to rank n items according to a human's preferences. If we can request a real-valued score for each item, (and for simplicity, assume the scores are unique) we can rank the items by requesting just n queries. However, if we request pairwise comparisons we must ask at least $n \log_2 n$ queries. To see this, there are $n! \approx n^n$ rankings which means that to describe a ranking, at least $n \log_2 n$ bits of information must be provided, but since a pairwise comparison provides at most one bit of information, at least this many pairwise comparison queries must be made [4]. On the other hand, the requested scores may be inaccurate leading to a less precise ranking than the one obtained using pairwise comparisons, so we see there is a tradeoff here. We will revisit this particular issue in Chapters 6 and 7. A second downside of pairwise comparisons is

the possibility for intransitivity of preferences: If I rate movies A,B,C scores with 3,4,5, respectively, then I may infer that $A \prec B$, $B \prec C$, and $A \prec C$ where $x \prec y$ is read as "x is preferred to y." However, if I ask for pairwise comparisons, it is possible to receive contradictions or intransitive information like $A \prec B$, $B \prec C$, and $C \prec A$. In this case the algorithm must have define a protocol for resolving these inconsistencies. While there exist approaches that operate in an agnostic and worst-case sense [5], in this work we model such contradictions as the result of "noise" in the sense that we model people as having transitive preferences but they occasionally will erroneously report inconsistent preferences by chance. We explore these issues further in Chapters 2, 6, and 7.

1.3 Pure Exploration for Multi-armed Bandits

Multi-armed bandits is a conceptual framework for sequential decision processes that reduces many complex problems from different domains down to a simple game closely resembling the two-player game introduced in Section 1.1. While in this thesis we are concerned with pure exploration games, there is a large body of literature in the multi-armed bandits field that balances exploration and exploitation, so it is prudent to take a moment to clarify the difference between the two.

In *stochastic* multi-armed bandit problems there are n "arms" representing the actions the player can take at each round, i.e. $\mathcal{Q} = [n]$ where $[n] = \{1, \ldots, n\}$. If at round t an arm $I_t = i \in [n]$ is selected, or "pulled", by the player for the jth time, a random variable $X_{i,j}$ is drawn from an unknown distribution with $\mathbb{E}[X_{i,j}] = \mu_i \in [0, 1]$. In the regret framework one wishes to balance exploration with exploitation and the player's goal is to minimize the cumulative regret of playing suboptimal arms: $\sum_{i=1}^{n} \max_{i \in [n]} \mu_i - \mu_{I_t}$, either

in expectation or with high probability. In the *pure exploration* framework with which we are interested in, the objective is to identify $\arg\max_{i\in[n]} \mu_i$ (assuming it is unique) with high probability in as few pulls, or queries, as possible. Therefore, a player's strategy for the pure exploration multi-armed bandit game is composed of deciding which arm to pull given all the observed pulls up to the current time, and recommending an arm believed to be optimal. In some formulations, as in Chapter 4, the algorithm (player) must also define a stopping time at which time the player declares that he has found the best arm with sufficiently high probability.

One can also define a *non-stochastic* multi-armed bandit game for both the regret and pure exploration frameworks. This scenario enforces less than in the stochastic case so fewer guarantees can be made, but it has the advantage of being applicable in more domains. The benign conditions on the responses from the arms are technical and require some motivation so we defer their introduction until Chapter 5.

We also consider the marriage of pairwise comparisons with multi-armed bandits.The *dueling bandits* framework, as it is known, was introduced by Yue et. al. [118] where at each round t a *pair* of arms $(i, j) \in [n]^2$ are chosen by the player and a Bernoulli random variable is observed whose mean $p_{i,j}$ is interpreted as the probability that arm i "beats" arm j in a duel. As alluded to in Section 1.2, it is possible that $p_{i,j} > 1/2$, $p_{j,k} > 1/2$ and $p_{i,k} < 1/2$ resulting in a cycle or an intransitive set of relations, making it difficult to define a "best" arm in general. Several definitions have been proposed throughout history for the "best" arm including the Condorcet, Borda, and Copeland winners. In this work we focus on the Borda winner because it always exists and also exhibits subtle structure that can be exploited by adaptive data collection methods.

1.4 Organization of the Dissertation

This thesis is organized into three parts. Each part presents a theme that is more or less self-contained, but draws context from those parts that precede it. Likewise, each chapter within each part is a variation on that theme and can be read on its own, but the reader may enjoy the context provided by the preceding chapters. Nevertheless, for the reader that chooses to read out of order, the text notes where it may be advisable to consult previous content.

In Chapter 2 we study the problem of identifying a ranking among a set of total orderings induced by known structure about the objects where queries take the form "which comes first in the ordering, A or B?" Chapter 3 is concerned with a related problem of identifying how n objects relate to each other using just queries of the form "is object C more similar to A or B?" In Chapter 4 we shift our attention to multi-armed bandits where given n stochastic sources that we can sample, we attempt to identify the source with the highest mean using as few total samples as possible. Chapter 5 studies the same problem as the previous chapter, but now the sources are no longer assumed to be stochastic, leading to more practical applications at the cost of weaker guarantees. In Chapter 6 we revisit the use of pairwise comparisons in a multi-armed bandit setting. Chapter 7 then considers the use pairwise comparison for derivative free optimization of a convex function.

A bibliographical remarks section is found at the end of each chapter describing the author's publications that contributed to the chapter as well as references to follow up work in the literature. For the committee's convenience, the authors's relevant publications contributing to this thesis are listed below:

- Kevin G Jamieson and Robert D Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2240–2248, 2011

- Kevin G Jamieson and Robert D Nowak. Active ranking in practice: General ranking functions with sample complexity bounds. In *NIPS Workshop*, 2011

- Kevin G Jamieson and Robert D Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 1077–1084. IEEE, 2011

- Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014

- Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–6. IEEE, 2014

- Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sebastien Bubeck. On finding the largest mean among many. *Signals, Systems and Computers (ASILOMAR)*, 2013

- Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *arXiv preprint arXiv:1502.07943*, 2015

- Kevin Jamieson, Sumeet Katariya, Atul Deshpande, and Robert Nowak. Sparse dueling bandits. In *AISTATS*, 2015

- Kevin G. Jamieson, Robert D Nowak, and Ben Recht. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2012

Part I

Concept Learning with Comparative Judgments

Chapter 2

Active Ranking

This chapter addresses the first adaptive learning problem we encountered in this thesis: the story of the bartender and patron. The patron was thinking of a beer, and the bartender attempted to identify it by asking as few questions as possible that are of the form "is it more similar to beer A or B?" where A and B are beers are from a finite set. In what follows, we modify the problem setup a little bit. We no longer assume that the patron's "ideal" beer is among the finite set of beers, and the bartender is now trying to rank the entire finite set of beers with respect to the patron's preferences just by asking the patron about his preferences in the form of pairwise comparisons: "do you prefer A or B?" It is known that n unstructured, but comparable items can be ranked using just $n \log_2(n)$ pairwise comparisons using an algorithm like binary sort. However, we know from the story of the bartender and the patron that there is sometimes substantial structure among the objects/beers that can be taken advantage of to reduce the number of questions. This chapter focuses on strategies of discovering a ranking over n objects using only $O(\log(n))$ comparisons by exploiting known structure.

2.1 Introduction

A *ranking* over a set of n objects $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ is a mapping $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that prescribes an order

$$\sigma(\mathcal{X}) := x_{\sigma(1)} \prec x_{\sigma(2)} \prec \cdots \prec x_{\sigma(n-1)} \prec x_{\sigma(n)} \tag{2.1}$$

where $x_i \prec x_j$ means x_i precedes x_j in the ranking. A ranking uniquely determines the collection of pairwise comparisons between all pairs of objects. The primary objective here is to bound the number of pairwise comparisons needed to correctly determine the ranking when the objects (and hence rankings) satisfy certain known structural constraints. Specifically, we suppose that the objects may be embedded into a low-dimensional Euclidean space such that the ranking is consistent with distances in the space. We wish to exploit such structure in order to discover the ranking using a very small number of pairwise comparisons.

We begin by assuming that every pairwise comparison is consistent with an unknown ranking. Each pairwise comparison can be viewed as a query: is x_i before x_j? Each query provides 1 bit of information about the underlying ranking. Since the number of rankings is $n!$, in general, specifying a ranking requires $\Theta(n \log n)$ bits of information. This implies that at least this many pairwise comparisons are required without additional assumptions about the ranking. In fact, this lower bound can be achieved with a standard adaptive sorting algorithm like binary sort [15]. In large-scale problems and n is very large or when humans are queried for pairwise comparisons, obtaining this many pairwise comparisons may be impractical and therefore we consider situations in which the space of rankings is structured and thereby less complex.

A natural way to induce a structure on the space of rankings is to suppose that the objects can be embedded into a d-dimensional Euclidean space so that the distances between objects are consistent with the ranking. This may be a reasonable assumption in many applications, and for instance the audio dataset used in our experiments is believed to have a 2 or 3 dimensional embedding [16]. We further discuss motivations for this assumption in Section 2.1.2. It is not difficult to show (see Section 2.2) that the number of full rankings that could arise from n objects embedded in $\mathbb{R}^d$ grows like n^{2d}, and so specifying a ranking from this class requires only $O(d \log n)$ bits. The main results of the paper show that under this assumption, a randomly selected ranking can be determined using $O(d \log n)$ pairwise comparisons selected in an adaptive and sequential fashion, but almost all $\binom{n}{2}$ pairwise rankings are needed if they are picked randomly rather than selectively. In other words, *actively selecting the most informative queries has a tremendous impact on the complexity of learning the correct ranking.*

2.1.1 Problem statement

Let σ denote the ranking to be learned. The objective is to learn the ranking by querying the reference for pairwise comparisons of the form

$$q_{i,j} := \{x_i \prec x_j\}. \tag{2.2}$$

The response or label of $q_{i,j}$ is binary and denoted as $y_{i,j} := \mathbf{1}\{q_{i,j}\}$ where $\mathbf{1}$ is the indicator function taking a value of 1 if its argument is true and 0 otherwise; ties are not allowed. The main results quantify the minimum number of queries or labels required to determine the reference's ranking, and they are based on two key assumptions.

A1 Embedding: The set of n objects are embedded in $\mathbb{R}^d$ (in general position) and we will also use $x_1, \ldots, x_n$ to refer to their (known) locations in $\mathbb{R}^d$. Every ranking σ can be specified by a *reference* point $r_\sigma \in \mathbb{R}^d$, as follows. The Euclidean distances between the reference and objects are consistent with the ranking in the following sense: if the σ ranks $x_i \prec x_j$, then $\|x_i - r_\sigma\| < \|x_j - r_\sigma\|$. Let $\Sigma_{n,d}$ denote the set of all possible rankings of the n objects that satisfy this embedding condition.

The interpretation of this assumption is that we know how the objects are related (in the embedding), which limits the space of possible rankings. The ranking to be learned, specified by the reference (e.g., preferences of the bar patron), is unknown. Many have studied the problem of finding an embedding of objects from data [17, 18, 19]. While related, this is not the focus here, but it could certainly play a supporting role in our methodology (e.g., the embedding could be determined from known similarities between the n objects, as is done in our experiments with the audio dataset). We assume the embedding is given and our interest is minimizing the number of queries needed to learn the ranking, and for this we require a second assumption.

A2 Consistency: Every pairwise comparison is consistent with the ranking to be learned. That is, if the reference ranks $x_i \prec x_j$, then x_i must precede x_j in the (full) ranking.

As we will discuss later in Section 2.2.2, these two assumptions alone are not enough to rule out pathological arrangements of objects in the embedding for which at least $\Omega(n)$ queries must be made to recover the ranking. However, because such situations are not representative of what is typically encountered, we analyze the problem in the framework of the average-case analysis [20].

Definition 2.1. *With each ranking $\sigma \in \Sigma_{n,d}$ we associate a probability π_σ such that*

$\sum_{\sigma \in \Sigma_{n,d}} \pi_\sigma = 1$. *Let* π *denote these probabilities and write* $\sigma \sim \pi$ *for shorthand. The uniform distribution corresponds to* $\pi_\sigma = |\Sigma_{n,d}|^{-1}$ *for all* $\sigma \in \Sigma_{n,d}$*, and we write* $\sigma \sim \mathcal{U}$ *for this special case.*

Definition 2.2. *If* $M_n(\sigma)$ *denotes the number of pairwise comparisons requested by an algorithm to identify the ranking* σ*, then the* average query complexity *with respect to* π *is denoted by* $\mathbb{E}_\pi[M_n]$.

We focus on the special case of $\pi = \mathcal{U}$, the uniform distribution, to make the analysis more transparent and intuitive. However in the statement and proof of our main result we show how to extend the results to general distributions π that satisfy certain mild conditions. All results henceforth, unless otherwise noted, will be given in terms of (uniform) average query complexity and we will say such results hold "on average."

Our main results can be summarized as follows. If the queries are chosen deterministically or randomly in advance of collecting the corresponding pairwise comparisons, then we show that almost all $\binom{n}{2}$ pairwise comparisons queries are needed to identify a ranking under the assumptions above. However, if the queries are selected in an adaptive and sequential fashion according to the algorithm in Figure 2.1, then we show that the number of pairwise rankings required to identify a ranking is no more than a constant multiple of $d \log n$, on average. The algorithm requests a query if and only if the corresponding pairwise ranking is ambiguous (see Section 2.3.2), meaning that it cannot be determined from previously collected pairwise comparisons and the locations of the objects in $\mathbb{R}^d$. The efficiency of the algorithm is due to the fact that *most* of the queries are unambiguous when considered in a sequential fashion. For this very same reason, picking queries in a non-adaptive or random fashion is very inefficient. It is also noteworthy that the algorithm is also computationally efficient with an overall complexity

no greater than $O(n \text{ poly}(d) \text{ poly}(\log n))$ (see Appendix A.1). In Section 2.4 we present a robust version of the algorithm of Figure 2.1 that is tolerant to a fraction of errors in the pairwise comparison queries. In the case of *persistent errors* (see Section 2.4) we show that we can find a probably approximately correct ranking by requesting just $O(d \log^2 n)$ pairwise comparisons. This allows us to handle situations in which either or both of the assumptions, **A1** and **A2**, are reasonable approximations to the situation at hand, but do not hold strictly (which is the case in our experiments with the audio dataset).

Proving the main results involves an uncommon marriage of ideas from the ranking and statistical learning literatures. Geometrical interpretations of our problem derive from the seminal works of [21] in ranking and [22] in learning. From this perspective our problem bears a strong resemblance to the halfspace learning problem, with two crucial distinctions. In the ranking problem, the underlying halfspaces are not in general position and have strong dependencies with each other. These dependencies invalidate many of the typical analyses of such problems [23, 24]. One popular method of analysis in exact learning involves the use of something called the *extended teaching dimension* [25]. However, because of the possible pathological situations alluded to earlier, it is easy to show that the extended teaching dimension must be at least $\Omega(n)$ making that sort of worst-case analysis uninteresting. These differences present unique challenges to learning.

2.1.2 Motivation and related work

The problem of learning a ranking from few pairwise comparisons is motivated by what we perceive as a significant gap in the theory of ranking and permutation learning. Most work in ranking with structural constraints assumes a passive approach to learning;

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\mathcal{X} = \{x_1, \ldots, x_n\}$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 2.1: Sequential algorithm for selecting queries. See Figure 2.2 and Section 2.3.2 for the definition of an ambiguous query.

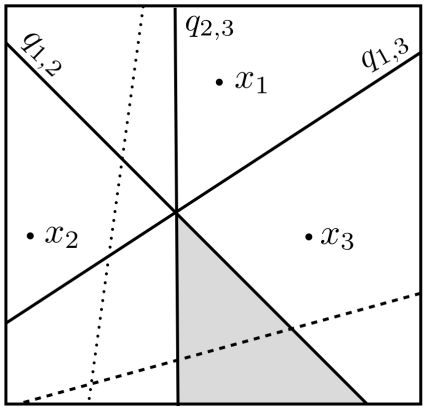


Figure 2.2: Objects x_1, x_2, x_3 and queries. The r_σ lies in the shaded region (consistent with the labels of $q_{1,2}, q_{1,3}, q_{2,3}$). The dotted (dashed) lines represent new queries whose labels are (are not) ambiguous given those labels.

pairwise comparisons or partial rankings are collected in a random or non-adaptive fashion and then aggregated to obtain a full ranking (cf. [26, 27, 28, 29]). However, this may be quite inefficient in terms of the number of pairwise comparisons or partial rankings needed to learn the (full) ranking. This inefficiency was recently noted in the related area of social choice theory [30]. Furthermore, empirical evidence suggests that adaptively selecting pairwise comparisons based on certain heuristics can reduce the number needed to learn the ranking [31, 32, 33]. In many applications it is expensive and time-consuming to obtain pairwise comparisons. For example, psychologists and market researchers collect pairwise comparisons to gauge human preferences over a set of objects, for scientific understanding or product placement. The scope of these experiments is often very limited simply due to the time and expense required to collect the data [3]. This suggests the consideration of more selective and judicious approaches to gathering inputs for ranking. We are interested in taking advantage of underlying structure in the

set of objects in order to choose more informative pairwise comparison queries. From a learning perspective, our work provides provable guarantees for active learning for a problem domain that has primarily been dominated by passive learning results.

We assume that the objects can be embedded in $\mathbb{R}^d$ and that the distances between objects and the reference are consistent with the ranking (Assumption **A1**). The problem of learning a general function $f : \mathbb{R}^d \to \mathbb{R}$ using just pairwise comparisons that correctly ranks the objects embedded in $\mathbb{R}^d$ has previously been studied in the passive setting [26, 27, 28, 29]. The main contributions of this paper are theoretical bounds for the specific case when $f(x) = ||x - r_\sigma||$ where $r_\sigma \in \mathbb{R}^d$ is the reference point. This is a standard model used in multidimensional unfolding and psychometrics [21, 34] and one can show that this model also contains the familiar functions $f(x) = r_\sigma^T x$ for all $r_\sigma \in \mathbb{R}^d$. We are unaware of any existing query-complexity bounds for this problem. We do not assume a generative model is responsible for the relationship between rankings to embeddings, but one could. For example, the objects might have an embedding (in a feature space) and the ranking is generated by distances in this space. Or alternatively, structural constraints on the space of rankings could be used to generate a consistent embedding. Assumption **A1**, while arguably quite natural/reasonable in many situations, significantly constrains the set of possible rankings.

2.2 Geometry of rankings from pairwise comparisons

The embedding assumption **A1** gives rise to geometrical interpretations of the ranking problem, which are developed in this section. The pairwise comparison $q_{i,j}$ can be viewed as the membership query: is x_i ranked before x_j in the (full) ranking σ? The geometrical

interpretation is that $q_{i,j}$ requests whether the reference r_σ is closer to object x_i or object x_j in $\mathbb{R}^d$. Consider the line connecting x_i and x_j in $\mathbb{R}^d$. The hyperplane that bisects this line and is orthogonal to it defines two halfspaces: one containing points closer to x_i and the other the points closer to x_j. Thus, $q_{i,j}$ is a membership query about which halfspace r_σ is in, and there is an equivalence between each query, each pair of objects, and the corresponding bisecting hyperplane. The set of all possible pairwise comparison queries can be represented as $\binom{n}{2}$ distinct halfspaces in $\mathbb{R}^d$. The intersections of these halfspaces partition $\mathbb{R}^d$ into a number of cells, and each one corresponds to a unique ranking of $\mathcal{X}$. Arbitrary rankings are not possible due to the embedding assumption **A1**, and recall that the set of rankings possible under **A1** is denoted by $\Sigma_{n,d}$. The cardinality of $\Sigma_{n,d}$ is equal to the number of cells in the partition. We will refer to these cells as d-cells (to indicate they are subsets in d-dimensional space) since at times we will also refer to lower dimensional cells; e.g., $(d-1)$-cells.

2.2.1 Counting the number of possible rankings

The following lemma determines the cardinality of the set of rankings, $\Sigma_{n,d}$, under assumption **A1**.

Lemma 2.3. *[21] Assume **A1-2**. Let $Q(n,d)$ denote the number of d-cells defined by the hyperplane arrangement of pairwise comparisons between these objects (i.e. $Q(n,d) = |\Sigma_{n,d}|$). $Q(n,d)$ satisfies the recursion*

$$Q(n,d) = Q(n-1,d) + (n-1)Q(n-1,d-1)\ ,\ \textit{where } Q(1,d) = 1 \textit{ and } Q(n,0) = 1. \tag{2.3}$$

In the hyperplane arrangement induced by the n objects in d dimensions, each hyperplane is intersected by every other and is partitioned into $Q(n-1, d-1)$ subsets or $(d-1)$-cells. The recursion, above, arises by considering the addition of one object at a time. Using this lemma in a straightforward fashion, we prove the following corollary in Appendix A.2.

Corollary 2.4. *Assume **A1-2**. There exist positive real numbers k_1 and k_2 such that*

$$k_1 \frac{n^{2d}}{2^d d!} < Q(n,d) < k_2 \frac{n^{2d}}{2^d d!}$$

for $n > d+1$. If $n \leq d+1$ then $Q(n,d) = n!$. For fixed d and n sufficiently large, $k_1 = 1/2$ and $k_2 = 2$ suffice.

2.2.2 Lower bounds on query complexity

Since the cardinality of the set of possible rankings is $|\Sigma_{n,d}| = Q(n,d)$, we have a simple lower bound on the number of queries needed to determine the ranking.

Theorem 2.5. *Assume **A1-2**. To reconstruct an arbitrary ranking $\sigma \in \Sigma_{n,d}$ any algorithm will require at least $\log_2 |\Sigma_{n,d}| = \Theta(2d \log_2 n)$ pairwise comparisons.*

Proof. By Corollary 2.4 $|\Sigma_{n,d}| = \Theta(n^{2d})$, and so at least $2d \log n$ bits are needed to specify a ranking. Each pairwise comparison provides at most one bit. □

If each query provides a full bit of information about the ranking, then we achieve this lower bound. For example, in the one-dimensional case ($d = 1$) the objects can be ordered and binary search can be used to select pairwise comparison queries, achieving the

lower bound. This is generally impossible in higher dimensions. Even in two dimensions there are placements of the objects (still in general position) that produce d-cells in the partition induced by queries that have $n-1$ faces (i.e., bounded by $n-1$ hyperplanes) as shown in Appendix A.3. It follows that the worst case situation may require at least $n-1$ queries in dimensions $d \geq 2$. In light of this, we conclude that worst case bounds may be overly pessimistic indications of the typical situation, and so we instead consider the average case performance introduced in Section 2.1.1.

2.2.3 Inefficiency of random queries

The geometrical representation of the ranking problem reveals that randomly choosing pairwise comparison queries is inefficient relative to the lower bound above. To see this, suppose m queries were chosen uniformly at random from the possible $\binom{n}{2}$. The answers to m queries narrows the set of possible rankings to a d-cell in $\mathbb{R}^d$. This d-cell may consist of one or more of the d-cells in the partition induced by all queries. If it contains more than one of the partition cells, then the underlying ranking is ambiguous.

Theorem 2.6. *Assume* ***A1-2****. Let* $N = \binom{n}{2}$*. Suppose* m *pairwise comparison are chosen uniformly at random without replacement from the possible* $\binom{n}{2}$*. Then for all positive integers* $N \geq m \geq d$ *the probability that the* m *queries yield a unique ranking is* $\binom{m}{d}/\binom{N}{d} \leq (\frac{em}{N})^d$.

Proof. No fewer than d hyperplanes bound each d-cell in the partition of $\mathbb{R}^d$ induced by all possible queries. The probability of selecting d specific queries in a random draw of

m is equal to

$$\binom{N-d}{m-d} \Big/ \binom{N}{m} = \binom{m}{d} \Big/ \binom{N}{d} \leq \frac{m^d}{d!} \frac{d^d}{N^d} \leq \left(\frac{m}{N}\right)^d \frac{d^d}{d!} \leq \left(\frac{em}{N}\right)^d. \qquad \square$$

Note that $\binom{m}{d}/\binom{N}{d} < 1/2$ unless $m = \Omega(n^2)$. Therefore, if the queries are randomly chosen, then we will need to ask almost all queries to guarantee that the inferred ranking is probably correct.

2.3 Analysis of sequential algorithm for query selection

Now consider the basic sequential process of the algorithm in Figure 2.1. Suppose we have ranked $k-1$ of the n objects. Call these objects 1 through $k-1$. This places the reference r_σ within a d-cell (defined by the labels of the comparison queries between objects $1, \ldots, k-1$). Call this d-cell C_{k-1}. Now suppose we pick another object at random and call it object k. A comparison query between object k and one of objects $1, \ldots, k-1$ can only be informative (i.e., ambiguous) if the associated hyperplane intersects this d-cell C_{k-1} (see Figure 2.2). If k is significantly larger than d, then it turns out that the cell C_{k-1} is probably quite small and the probability that one of the queries intersects C_{k-1} is very small; in fact the probability is on the order of $1/k^2$.

2.3.1 Hyperplane-point duality

Consider a hyperplane $\mathbf{h} = (h_0, h_1, \ldots, h_d)$ with $(d+1)$ parameters in $\mathbb{R}^d$ and a point $\mathbf{p} = (p_1, \ldots, p_d) \in \mathbb{R}^d$ that does not lie on the hyperplane. Checking which halfspace $\mathbf{p}$ falls in, i.e., $h_1 p_1 + h_2 p_2 + \cdots + h_d p_d + h_0 \gtrless 0$, has a dual interpretation: $\mathbf{h}$ is a point in $\mathbb{R}^{d+1}$ and $\mathbf{p}$ is a hyperplane in $\mathbb{R}^{d+1}$ passing through the origin (i.e., with d free parameters).

Recall that each possible ranking can be represented by a reference point $r_\sigma \in \mathbb{R}^d$. Our problem is to determine the ranking, or equivalently the vector of responses to the $\binom{n}{2}$ queries represented by hyperplanes in $\mathbb{R}^d$. Using the above observation, we see that our problem is equivalent to finding a labeling over $\binom{n}{2}$ points in $\mathbb{R}^{d+1}$ with as few queries as possible. We will refer to this alternative representation as the dual and the former as the primal.

2.3.2 Characterization of an ambiguous query

The characterization of an ambiguous query has interpretations in both the primal and dual spaces. We will now describe the interpretation in the dual which will be critical to our analysis of the sequential algorithm of Figure 2.1.

Definition 2.7. *[22] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to S.

Lemma 2.8. *[22, Lemma 1] The label of x is ambiguous with respect to S if and only if S^+ and S^- are homogeneously linearly separable by a $(d-1)$-dimensional subspace*

containing x.

Let us consider the implications of this lemma to our scenario. Assume that we have labels for all the pairwise comparisons of $k-1$ objects. Next consider a new object called object k. In the dual, the pairwise comparison between object k and object i, for some $i \in \{1, \ldots, k-1\}$, is ambiguous if and only if there exists a hyperplane that still separates the original points and also passes through this new point. In the primal, this separating hyperplane corresponds to a point lying on the hyperplane defined by the associated pairwise comparison.

2.3.3 The probability that a query is ambiguous

An essential component of the sequential algorithm of Figure 2.1 is the initial random order of the objects; every sequence in which it could consider objects is equally probable. This allows us to state a nontrivial fact about the partial rankings of the first k objects observed in this sequence.

Lemma 2.9. *Assume **A1-2** and $\sigma \sim \mathcal{U}$. Consider the subset $S \subset \mathcal{X}$ with $|S| = k$ that is randomly selected from $\mathcal{X}$ such that all $\binom{n}{k}$ subsets are equally probable. If $\Sigma_{k,d}$ denotes the set of possible rankings of these k objects then every $\sigma \in \Sigma_{k,d}$ is equally probable.*

Proof. Let a k-partition denote the partition of $\mathbb{R}^d$ into $Q(k,d)$ d-cells induced by k objects for $1 \leq k \leq n$. In the n-partition, each d-cell is weighted uniformly and is equal to $1/Q(n,d)$. If we uniformly at random select k objects from the possible n and consider the k-partition, each d-cell in the k-partition will contain one or more d-cells of the n-partition. If we select one of these d-cells from the k-partition, on average there will be $Q(n,d)/Q(k,d)$ d-cells from the n-partition contained in this cell. Therefore the

probability mass in each d-cell of the k-partition is equal to the number of cells from the n-partition in this cell multiplied by the probability of each of those cells from the n-partition: $Q(n,d)/Q(k,d) \times 1/Q(n,d) = 1/Q(k,d)$, and $|\Sigma_{k,d}| = Q(k,d)$. □

As described above, for $1 \leq i \leq k$ some of the pairwise comparisons $q_{i,k+1}$ may be ambiguous. The algorithm chooses a random sequence of the n objects in its initialization and does not use the labels of $q_{1,k+1}, \ldots, q_{j-1,k+1}, q_{j+1,k+1}, \ldots, q_{k,k+1}$ to make a determination of whether or not $q_{j,k+1}$ is ambiguous. It follows that the events of requesting the label of $q_{i,k+1}$ for $i = 1, 2, \ldots, k$ are independent and identically distributed (conditionally on the results of queries from previous steps). Therefore it makes sense to talk about the probability of requesting any one of them.

Lemma 2.10. *Assume* ***A1-2*** *and* $\sigma \sim \mathcal{U}$. *Let* $A(k,d,\mathcal{U})$ *denote the probability of the event that the pairwise comparison* $q_{i,k+1}$ *is ambiguous for* $i = 1, 2, \ldots, k$. *Then there exists a positive, real number constant* a *independent of* k *such that for* $k \geq 2d$, $A(k,d,\mathcal{U}) \leq a\frac{2d}{k^2}$.

Proof. By Lemma 2.8, a point in the dual (pairwise comparison) is ambiguous if and only if there exists a separating hyperplane that passes through this point. This implies that the hyperplane representation of the pairwise comparison in the primal intersects the cell containing r_σ (see Figure 2.2 for an illustration of this concept). Consider the partition of $\mathbb{R}^d$ generated by the hyperplanes corresponding to pairwise comparisons between objects $1, \ldots, k$. Let $P(k,d)$ denote the number of d-cells in this partition that are intersected by a hyperplane corresponding to one of the queries $q_{i,k+1}$, $i \in \{1, \ldots, k\}$. Then it is not difficult to show that $P(k,d)$ is bounded above by a constant independent of n and k times $\frac{k^{2(d-1)}}{2^{d-1}(d-1)!}$ (see Appendix A.4). By Lemma 2.9, every d-cell in the partition

induced by the k objects corresponds to an equally probable ranking of those objects. Therefore, the probability that a query is ambiguous is the number of cells intersected by the corresponding hyperplane divided by the total number of d-cells, and therefore $A(k, d, \mathcal{U}) = \frac{P(k,d)}{Q(k,d)}$. The result follows immediately from the bounds on $P(k, d)$ and Corollary 2.4. □

Because the individual events of requesting each query are conditionally independent, the total number of queries requested by the algorithm is just $M_n = \sum_{k=1}^{n-1} \sum_{i=1}^{k} \mathbf{1}\{\text{Request } q_{i,k+1}\}$. Using the results above, it straightforward to prove our main result.

Theorem 2.11. *Assume **A1-2** and $\sigma \sim \mathcal{U}$. Let the random variable M_n denote the number of pairwise comparisons that are requested in the algorithm of Figure 2.1, then*

$$\mathbb{E}_{\mathcal{U}}[M_n] \leq \lceil 2da \rceil \log_2 n.$$

Furthermore, if $\sigma \sim \pi$ and $\max_{\sigma \in \Sigma_{n,d}} \pi_\sigma \leq c|\Sigma_{n,d}|^{-1}$ for some $c > 0$, then $\mathbb{E}_\pi[M_n] \leq c\, \mathbb{E}_{\mathcal{U}}[M_n]$.

Proof. Let B_{k+1} denote the total number of pairwise comparisons requested of the $(k+1)$st object; i.e., number of ambiguous queries in the set $q_{i,k+1}$, $i = 1, \ldots, k$. Because the individual events of requesting these are conditionally independent (see Section 2.3.3), it follows that each B_{k+1} is an independent binomial random variable with parameters $A(k, d, \mathcal{U})$ and k. The total number of queries requested by the algorithm is

$$M_n = \sum_{k=1}^{n-1} \sum_{i=1}^{k} \mathbf{1}\{\text{Request } q_{i,k+1}\} = \sum_{k=1}^{n-1} B_{k+1} \ . \tag{2.4}$$

Because Lemma 2.10 is only relevant for sufficiently large k, we assume that none of the

pairwise comparisons are ambiguous when $k \leq 2da$. Recall from Section A.1 that binary sort is implemented so for these first $\lceil 2da \rceil$ objects, at most $\lceil 2da \rceil \log_2(\lceil 2da \rceil)$ queries are requested. For $k > 2da$ the number of requested queries to the kth object is upper bounded by the number of ambiguous queries of the kth object. Then using the known mean and variance formulas for the binomial distribution

$$\begin{aligned}
\mathbb{E}_{\mathcal{U}}[M_n] &= \sum_{k=1}^{n-1} \mathbb{E}_{\mathcal{U}}[B_{k+1}] \\
&\leq \sum_{k=2}^{\lceil 2da \rceil} B_{k+1} + \sum_{k=\lceil 2da \rceil + 1}^{n-1} \frac{2da}{k} \\
&\leq \lceil 2da \rceil \log_2 \lceil 2da \rceil + 2da \log\left(n/\lceil 2da \rceil\right) \\
&\leq \lceil 2da \rceil \log_2 n
\end{aligned}$$

We now consider the case for a general distribution π. Enumerate the rankings of $\Sigma_{n,d}$. Let N_i denote the (random) number of requested queries needed by the algorithm to reconstruct the ith ranking. Note that the randomness of N_i is only due to the randomization of the algorithm. Let π_i denote the probability it assigns to the ith ranking as in Definition 2.1. Then

$$\mathbb{E}_{\pi}[M_n] \;=\; \sum_{i=1}^{Q(n,d)} \pi_i \, \mathbb{E}[N_i]. \tag{2.5}$$

Assume that the distribution over rankings is bounded above such that no ranking is overwhelmingly probable. Specifically, assume that the probability of any one ranking is upper bounded by $c/Q(n,d)$ for some constant $c > 1$ that is independent of n. Under this bounded distribution assumption, $\mathbb{E}_{\pi}[M_n]$ is maximized by placing probability $c/Q(n,d)$

on the $k := Q(n,d)/c$ cells for which $\mathbb{E}[N_i]$ is largest (we will assume k is an integer, but it is straightforward to extend the following argument to the general case). Since the mass on these cells is equal, without loss of generality we may assume that $\mathbb{E}[N_i] = \mu$, a common value on each, and we have $\mathbb{E}_\pi[M_n] = \mu$. For the remaining $Q(n,d) - k$ cells we know that $\mathbb{E}[N_i] \geq d$, since each cell is bounded by at least d hyperplanes/queries. Under these conditions, we can relate $\mathbb{E}_\pi[M_n]$ to $\mathbb{E}_\mathcal{U}[M_n]$ as follows. First observe that

$$\mathbb{E}_\mathcal{U}[M_n] \;=\; \frac{1}{Q(n,d)} \sum_{i=1}^{Q(n,d)} \mathbb{E}[N_i] \;\geq\; \frac{k}{Q(n,d)}\mu + d\,\frac{Q(n,d)-k}{Q(n,d)}\;,$$

which implies

$$\mathbb{E}_\pi[M_n] \;=\; \mu \;\leq\; \tfrac{Q(n,d)}{k}\left(\mathbb{E}_\mathcal{U}[M_n] - d\tfrac{Q(n,d)-k}{Q(n,d)}\right) \;=\; c\left(\mathbb{E}_\mathcal{U}[M_n] - d\tfrac{Q(n,d)-k}{Q(n,d)}\right) \;\leq\; c\,\mathbb{E}_\mathcal{U}[M_n]\;.$$

In words, the non-uniformity constant $c > 1$ scales the expected number of queries. Under **A1-2**, for large n we have $\mathbb{E}_\pi[M_n] = O(c\,d\log n)$. □

2.4 Robust sequential algorithm for query selection

We now extend the algorithm of Figure 2.1 to situations in which the response to each query is only probably correct. If the correct label of a query $q_{i,j}$ is $y_{i,j}$, we denote the possibly incorrect response by $Y_{i,j}$. Let the probability that $Y_{i,j} = y_{i,j}$ be equal to $1-p$, $p < 1/2$. The robust algorithm operates in the same fashion as the algorithm in Figure 2.1, with the exception that when an ambiguous query is encountered several (equivalent) queries are made and a decision is based on the majority vote. We will now judge performance based on two metrics: (i) how many queries are requested and (ii)

how accurate the estimated ranking is with respect to the true ranking before it was corrupted. For any two rankings σ, $\widehat{\sigma}$ we adopt the popular Kendell-Tau distance [35]

$$d_\tau(\sigma, \widehat{\sigma}) = \sum_{(i,j):\sigma(i)<\sigma(j)} \mathbf{1}\{\widehat{\sigma}(j) < \widehat{\sigma}(i)\} \tag{2.6}$$

where $\mathbf{1}$ is the indicator function. Clearly, $d_\tau(\sigma, \widehat{\sigma}) = d_\tau(\widehat{\sigma}, \sigma)$ and $0 \leq d_\tau(\sigma, \widehat{\sigma}) \leq \binom{n}{2}$. For any ranking $\sigma \in \Sigma_{n,d}$ we wish to find an estimate $\widehat{\sigma} \in \Sigma_{n,d}$ that is close in terms of $d_\tau(\sigma, \widehat{\sigma})$ without requesting too many pairwise comparisons. For convenience, we will some times report results in terms of the proportion ϵ of incorrect pairwise orderings such that $d_\tau(\sigma, \widehat{\sigma}) \leq \epsilon\binom{n}{2}$. Using the equivalence of the Kendell-Tau and Spearman's footrule distances (see [36]), if $d_\tau(\sigma, \widehat{\sigma}) \leq \epsilon\binom{n}{2}$ then each object in $\widehat{\sigma}$ is, on average, no more than $O(\epsilon n)$ positions away from its position in σ. Thus, the Kendell-Tau distance is an intuitive measure of closeness between two rankings.

First consider the case in which each query can be repeated to obtain multiple independent responses (votes) for each comparison query. This *random errors* model arises, for example, in social choice theory where the "reference" is a group of people, each casting a vote.

Theorem 2.12. *Assume* ***A1-2*** *and* $\sigma \sim \mathcal{U}$ *but that each response to the query* $q_{i,j}$ *is a realization of an i.i.d. Bernoulli random variable* $Y_{i,j}$ *with* $P(Y_{i,j} \neq y_{i,j}) \leq p < 1/2$ *for all distinct* $i, j \in \{1, \ldots, n\}$. *If all ambiguous queries are decided by the majority vote of* R *independent responses to each such query, then with probability greater than* $1 - 2n\log_2(n)\exp(-\frac{1}{2}(1-2p)^2 R)$ *this procedure correctly identifies the correct ranking (i.e.* $\epsilon = 0$*) and requests no more than* $O(Rd\log n)$ *queries on average.*

Proof. Suppose $q_{i,j}$ is ambiguous. Let $\hat{\alpha}$ be the frequency of $Y_{i,j} = 1$ after R trials. Let

$\mathbb{E}[\hat{\alpha}] = \alpha$. The majority vote decision is correct if $|\alpha - \hat{\alpha}| \leq 1/2 - p$. By Chernoff's bound, $\mathbb{P}(|\alpha - \hat{\alpha}| \geq 1/2 - p) \leq 2\exp(-2(1/2 - p)^2 R)$. The result follows from the union bound over the total number of queries considered: $n \log_2 n$. □

We can deduce from the above theorem that to exactly recover the true ranking under the stated conditions with probability $1 - \delta$, one need only request $O\Big(d(1 - 2p)^{-2} \log^2(n/\delta)\Big)$ pairwise comparisons, on average.

In other situations, if we ask the same query multiple times we may get the same, possibly incorrect, response each time. This *persistent errors* model is natural, for example, if the reference is a *single* human. Under this model, if two rankings differ by only a single pairwise comparison, then they cannot be distinguished with probability greater than $1 - p$. So, in general, exact recovery of the ranking cannot be guaranteed with high probability. The best we can hope for is to exactly recover a *partial ranking* of the objects (i.e. the ranking over a subset of the objects) or a ranking that is merely probably approximately correct in terms of the Kendell-Tau distance of (2.6). We will first consider the task of exact recovery of a partial ranking of objects and then turn our attention to the recovery of an approximate ranking. Henceforth, we will assume the errors are persistent.

2.4.1 Robust sequential algorithm for persistent errors

The robust query selection algorithm for persistent errors is presented in Figure 2.3. The key ingredient in the persistent errors setting is the design of a voting set for each ambiguous query encountered. Suppose the query $q_{i,j}$ is ambiguous in the algorithm of Figure 2.1. In principle, a voting set could be constructed using objects ranked between

i and j. If object k is between i and j, then note that $y_{i,j} = y_{i,k} = y_{k,j}$. In practice, we cannot identify the subset of objects ranked between i and j exactly, but we can find a set that contains them. For an ambiguous query $q_{i,j}$ define

$$T_{i,j} := \{k \in \{1 \ldots, n\} : \ q_{i,k},\, q_{k,j}, \text{ or both are ambiguous}\}. \tag{2.7}$$

Then $T_{i,j}$ contains all objects ranked between i and j (if k is ranked between i and j, and $q_{i,k}$ and $q_{k,j}$ are unambiguous, then so is $q_{i,j}$, a contradiction). Furthermore, if the first $j-1$ objects ranked in the algorithm were selected uniformly at random (or initialized in a random order in the algorithm) Lemma 2.9 implies that each object in $T_{i,j}$ is ranked between i and j with probability at least $1/3$ due to the uniform distribution over the rankings $\Sigma_{n,d}$ (see proof of Theorem 2.13 for an explanation). $T_{i,j}$ will be our voting set. If we follow the sequential procedure of the algorithm of Figure 2.3, the first query encountered, call it $q_{1,2}$, will be ambiguous and $T_{1,2}$ will contain all the other $n-2$ objects. However, at some point for some query $q_{i,j}$ it will become probable that the objects i and j are closely ranked. In that case, $T_{i,j}$ may be rather small, and so it is not always possible to find a sufficiently large voting set to accurately determine $y_{i,j}$. Therefore, we must specify a size-threshold $R \geq 0$. If the size of $T_{i,j}$ is at least R, then we draw R indices from $T_{i,j}$ uniformly at random without replacement, call this set $\{t_l\}_{l=1}^{R}$, and decide the label for $q_{i,j}$ by voting over the responses to $\{q_{i,k}, q_{k,j} : k \in \{t_l\}_{l=1}^{R}\}$; otherwise we pass over object j and move on to the next object in the list. Given that $|T_{i,j}| \geq R$

Robust Query Selection Algorithm

input: n objects in $\mathbb{R}^d$, $R \geq 0$
initialize: objects $\mathcal{X} = \{x_1, \ldots, x_n\}$ in uniformly random order, $\mathcal{X}' = \mathcal{X}$

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      $T_{i,j} := \{k \in \{1 \ldots, n\} : \ q_{i,k}, q_{k,j}$, or both are ambiguous$\}$
      if $|T_{i,j}| \geq R$
        $\{t_l\}_{l=1}^{R} \overset{i.i.d.}{\sim}$ uniform$(T_{i,j})$.
        request $Y_{i,k}$, $Y_{k,j}$ for all $k \in \{t_l\}_{l=1}^{R}$
        decide label of $q_{i,j}$ with (2.8)
      else
        $\mathcal{X}' \leftarrow \mathcal{X}' \setminus x_j \ , \quad j \leftarrow j+1$
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking over objects in $\mathcal{X}'$

Figure 2.3: Robust sequential algorithm for selecting queries of Section 2.4.1. See Figure 2.2 and Section 2.3.2 for the definition of an ambiguous query.

the label of $q_{i,j}$ is determined by:

$$0 \underset{\prec i}{\overset{i \prec j}{\gtrless}} \sum_{k \in \{t_l\}_{l=1}^{R}} \mathbf{1}\{Y_{i,k} = 1 \wedge Y_{k,j} = 1\} - \mathbf{1}\{Y_{i,k} = 0 \wedge Y_{k,j} = 0\}. \tag{2.8}$$

In the next section we will analyze this algorithm and show that it enjoys a very favorable query complexity while also admitting a probably approximately correct ranking.

2.4.2 Analysis of the robust sequential algorithm

Consider the robust algorithm in Figure 2.3. At the end of the process, some objects that were passed over may then be unambiguously ranked (based on queries made after they were passed over) or they can be ranked without voting (and without guarantees). As

mentioned in Section 2.4.1, if the first $j-1$ objects ranked in the algorithm of Figure 2.3 were chosen uniformly at random from the full set (i.e., none of the first $j-1$ objects were passed over) then there is at least a one in three chance each object in $T_{i,j}$ for some ambiguous query $q_{i,j}$ is ranked between i and j.

Theorem 2.13. *Assume* ***A1-2****, $\sigma \sim \mathcal{U}$, and $P(Y_{i,j} \neq y_{i,j}) = p$. For every set $T_{i,j}$ constructed in the algorithm of Figure 2.3, assume that an object selected uniformly at random from $T_{i,j}$ is ranked between x_i and x_j with probability at least $1/3$. Then for any size-threshold $R \geq 1$, with probability greater than $1 - 2n\log_2(n)\exp\left(-\frac{2}{9}(1-2p)^2 R\right)$ the algorithm correctly ranks at least $n/(2R+1)$ objects and requests no more than $O(Rd\log n)$ queries on average.*

Proof. Suppose $q_{i,j}$ is ambiguous. Let $S_{i,j}$ denote the subset of $\mathcal{X}$ such that $x_k \in S_{i,j}$ if it is ranked between objects x_i and x_j (i.e. $S_{i,j} = \{x_k \in \mathcal{X} : x_i \prec x_k \prec x_j$ or $x_j \prec x_k \prec x_i\}$). Note that $y_{i,j} = y_{i,k} = y_{k,j}$ if and only if $x_k \in S_{i,j}$. If we define $E^k_{i,j} = \mathbf{1}\{Y_{i,k} = 1 \wedge Y_{k,j} = 1\} - \mathbf{1}\{Y_{i,k} = 0 \wedge Y_{k,j} = 0\}$, where $\mathbf{1}$ is the indicator function, then for any subset $T \subset \mathcal{X}$ such that $S_{i,j} \subset T$, the sign of the sum $\sum_{x_k \in T} E^k_{i,j}$ is a predictor of $y_{i,j}$. In fact, with respect to just the random errors, $\mathbb{E}\left[\left|\sum_{x_k \in T} E^k_{i,j}\right|\right] = |S_{i,j}|(1-2p)$. To see this, without loss of generality let $y_{i,j} = 1$, then for $x_k \in S_{i,j}$

$$\begin{aligned}
\mathbb{E}[E^k_{i,j}] &= \mathbb{E}\left[\mathbf{1}\{Y_{i,k} = 1 \wedge Y_{k,j} = 1\} - \mathbf{1}\{Y_{i,k} = 0 \wedge Y_{k,j} = 0\}\right] \\
&= \mathbb{P}(Y_{i,k} = 1 \wedge Y_{k,j} = 1) - \mathbb{P}(Y_{i,k} = 0 \wedge Y_{k,j} = 0) \\
&= (1-p)^2 - p^2 \\
&= 1 - 2p.
\end{aligned}$$

If $x_k \notin S_{i,j}$ then it can be shown by a similar calculation that $\mathbb{E}[E^k_{i,j}] = 0$.

To identify $S_{i,j}$ we use the fact that if $x_k \in S_{i,j}$ then $q_{i,k}$, $q_{j,k}$, or both are also ambiguous simply because otherwise $q_{i,j}$ would not have been ambiguous in the first place (Figure 2.4 may be a useful aid to see this). While the converse is false, Lemma 2.9 says that each of the six possible rankings of $\{x_i, x_j, x_k\}$ are equally probable if they were uniformly at random chosen (thus partly justifying this explicit assumption in the theorem statement). It follows that if we define the subset $T_{i,j} \in \mathcal{X}$ to be those objects x_k with the property that $q_{i,k}$, $q_{k,j}$, or both are ambiguous then the probability that $x_k \in S_{i,j}$ is at least $1/3$ if $x_k \subset T_{i,j}$. You can convince yourself of this using Figure 2.4. Moreover, $\mathbb{E}\left[\left|\sum_{k\in T_{i,j}} E^k_{i,j}\right|\right] \geq |T_{i,j}|(1-2p)/3$ which implies the sign of the sum $\sum_{x_k \in T_{i,j}} E^k_{i,j}$ is a reliable predictor of $q_{i,j}$; just how reliable depends only on the size of $T_{i,j}$.

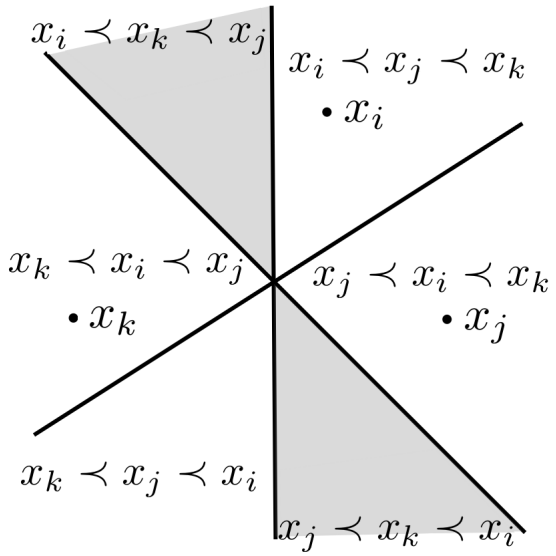


Figure 2.4: Let $q_{i,j}$ be ambiguous. Object k will be informative to the majority vote of $y_{i,j}$ if the reference lies in the shaded region. There are six possible rankings and if $q_{i,k}$, $q_{k,j}$, or both are ambiguous then the probability that the reference is in the shaded region is at least $1/3$

Fix $R > 0$. Suppose $q_{i,j}$ is ambiguous and assume without loss of generality that $y_{i,j} =$

1. Given that $\mathbb{E}\left[\sum_{k\in T_{i,j}} E_{i,j}^k\right] \geq |T_{i,j}|(1-2p)/3$ from above, it follows from Hoeffding's inequality that the probability that $\sum_{k\in T_{i,j}} E_{i,j}^k \leq 0$ is less than $\exp\left(-\frac{2}{9}(1-2p)^2|T_{i,j}|\right)$. If only a subset of $T_{i,j}$ of size R is used in the sum then $|T_{i,j}|$ is replaced by R in the exponent. This test is only performed when $|T_{i,j}| > R$ and clearly no more times than the number of queries considered to rank n objects in the full ranking: $n\log_2 n$. Thus, all decisions using this test are correct with probability at least $1 - 2n\log_2(n)\exp\left(-\frac{2}{9}(1-2p)^2R\right)$. Only a subset of the n objects will be ranked and of those, $2R+1$ times more queries will be requested than in the error-free case (two queries per object in $T_{i,j}$). Thus the robust algorithm will request no more than $O(Rd\log n)$ queries on average.

To determine the number of objects that are in the partial ranking, let $\mathcal{X}' \subset \mathcal{X}$ denote the subset of objects that are ranked in the output partial ranking. Each $x_k \in \mathcal{X}'$ is associated with an index in the true full ranking and is denoted by $\sigma(x_k)$. That is, if $\sigma(x_k) = 5$ then it is ranked fifth in the full ranking but in the partial ranking could be ranked first, second, third, fourth, or fifth. Now imagine the real line with tick marks only at the integers $1,\ldots,n$. For each $x_k \in \mathcal{X}'$ place an R-ball around each x_k on these tick marks such that if $\sigma(x_k) = 5$ and $R = 3$ then $2,\ldots,8$ are covered by the ball around $\sigma(x_k)$ and 1 and $9,\ldots,n$ are not. Then the union of the balls centered at the objects in $\mathcal{X}'$ cover $1,\ldots,n$. If this were not true then there would be an object $x_j \notin \mathcal{X}'$ with $|S_{i,j}| > R$ for all $x_i \in \mathcal{X}'$. But $S_{i,j} \subset T_{i,j}$ implies $|T_{i,j}| > R$ which implies $j \in \mathcal{X}'$, a contradiction. Because at least $n/(2R+1)$ R-balls are required to cover $1,\ldots,n$, at least this many objects are contained in $\mathcal{X}'$. □

Note that before the algorithm skips over an object for the first time, all objects that are ranked at such an intermediate stage are a subset chosen uniformly at random from

the full set of objects, due to the initial randomization. Therefore, if $T_{i,j}$ is a voting set in this stage, an object selected uniformly at random from $T_{i,j}$ is ranked between x_i and x_j with probability at least 1/3, per Lemma 2.9. After one or more objects are passed over, however, the distribution is no longer necessarily uniform due to this action, and so the assumption of the theorem above may not hold. The procedure of the algorithm is still reasonable, but it is difficult to give guarantees on performance without the assumption. Nevertheless, this discussion leads us to wonder how many objects the algorithm will rank before it skips over its first object.

Lemma 2.14. *Consider a ranking of n objects and suppose objects are drawn sequentially, chosen uniformly at random without replacement. If M is the largest integer such that M objects are drawn before any object is within R positions of another one in the ranking, then $M \geq \sqrt{\frac{n/R}{6\log(2)}}$ with probability at least $\frac{1}{6\log(2)}\left(e^{-(\sqrt{6\log(2)R/n}+1)^2/2} - 2^{-n/(3R)}\right)$. As $n/R \to \infty$, $P(M \geq \sqrt{\frac{n/R}{6\log(2)}}) \to \frac{1}{6\sqrt{e}\log(2)}$.*

Proof. Assume $M \leq \frac{n}{3R}$. If p_m denotes the probability that the $(m+1)$st object is within R positions of one of the first m objects, given that none of the first m objects are within R positions of each other, then $\frac{Rm}{n} < p_m \leq \frac{2Rm}{n-m}$ and

$$P(M = m) \geq \prod_{l=1}^{m-1}\left(1 - \frac{2Rl}{n-l}\right)\frac{Rm}{n}.$$

Taking the log we find

$$\begin{aligned}
\log P(M=m) &\geq \log\frac{Rm}{n} + \sum_{l=1}^{m-1}\log\left(1-\frac{2Rl}{n-l}\right) \\
&\geq \log\frac{Rm}{n} + (m-1)\log\left(\frac{1}{(m-1)}\sum_{l=1}^{m-1}\left(1-\frac{2Rl}{n-l}\right)\right) \\
&\geq \log\frac{Rm}{n} + (m-1)\log\left(1-\frac{Rm}{n-m+1}\right) \\
&\geq \log\frac{Rm}{n} + (m-1)\log\left(1-\frac{3Rm}{2n}\right) \\
&\geq \log\frac{Rm}{n} + (m-1)\left(-\frac{3\log(2)Rm}{n}\right)
\end{aligned}$$

where the second line follows from Jensen's inequality, the fourth line follows from the fact that $m \leq \frac{n}{3R}$, and the last line follows from the fact that $(1-x) \geq \exp(-2\log(2)x)$ for $x \leq 1/2$. We conclude that $P(M=m) \geq \frac{R}{n}m\exp\{-3\log(2)\frac{R}{n}m^2\}$. Now if $a = \sqrt{\frac{n/R}{6\log(2)}}$ we have

$$\begin{aligned}
P(M \geq a) &\geq \sum_{m=\lceil a\rceil}^{n/(3R)-1}\frac{R}{n}m\exp\{-3\log(2)\frac{R}{n}m^2\} \\
&\geq \int_{a+1}^{n/(3R)}\frac{R}{n}x\exp\{-3\log(2)\frac{R}{n}x^2\}dx \\
&= \frac{1}{6\log(2)}\left(e^{-(\sqrt{6\log(2)R/n}+1)^2/2} - e^{-\log(2)n/(3R)}\right)
\end{aligned}$$

where the second line follows from the fact that $xe^{-\alpha x^2/2}$ is monotonically decreasing for $x \geq \sqrt{1/\alpha}$. Note, $P(M \geq \sqrt{\frac{n/R}{6\log(2)}})$ is greater than $\frac{1}{100}$ for $n/R \geq 7$, and $\frac{1}{10}$ for $n/R \geq 40$. Moreover, as $n/R \to \infty$, $P(M \geq \sqrt{\frac{n/R}{6\log(2)}}) \to \frac{1}{6\sqrt{e}\log(2)}$. □

Lemma 2.14 characterizes how many objects the robust algorithm will rank before

it passes over its first object because if there are at least R objects between every pair of the first M objects, then $T_{i,j} \geq R$ for all distinct $i, j \in \{1, \ldots, M\}$ and none of the first M objects will be passed over. We can conclude from Lemma 2.14 and Theorem 2.13 that with constant probability (with respect to the initial ordering of the objects and the randomness of the voting), the algorithm of Figure 2.3 exactly recovers a partial ranking of at least $\Omega(\sqrt{(1-2p)^2 n / \log n})$ objects by requesting just $O\left(d(1-2p)^{-2} \log^2 n\right)$ pairwise comparisons, on average, with respect to all the rankings in $\Sigma_{n,d}$. If we repeat the algorithm with different initializations of the objects each time, we can boost this constant probability to an arbitrarily high probability (recall that the responses to queries will not change over the repetitions). Note, however, that the correctness of the partial ranking does not indicate how approximately correct the remaining rankings will be. If the algorithm of Figure 2.3 ranks m objects before skipping over its first, then the next lemma quantifies how accurate an estimated ranking is in terms of Kendel-Tau distance, given that it is some ranking in $\Sigma_{n,d}$ that is consistent with the probably correct partial ranking of the first m objects (the output ranking of the algorithm may contain more than m objects but we make no guarantees about these additional objects).

Lemma 2.15. *Assume **A1-2** and $\sigma \sim \mathcal{U}$. Suppose we select $1 \leq m < n$ objects uniformly at random from the n and correctly rank them amongst themselves. If $\widehat{\sigma}$ is any ranking in $\Sigma_{n,d}$ that is consistent with all the known pairwise comparisons between the m objects, then $\mathbb{E}[d_\tau(\sigma, \widehat{\sigma})] = O(d/m^2)\binom{n}{2}$, where the expectation is with respect to the random selection of objects and the distribution of the rankings $\mathcal{U}$.*

Proof. Enumerate the objects such that the first m are the objects ranked amongst

themselves. Let y be the pairwise comparison label vector for σ and $\hat{y}$ be the corresponding vector for $\widehat{\sigma}$. Then

$$\begin{aligned}
\mathbb{E}[d_\tau(\sigma,\widehat{\sigma})] &= \sum_{k=2}^{m}\sum_{l=1}^{k-1}\mathbf{1}\{y_{l,k}\neq\hat{y}_{l,k}\} + \sum_{k=m+1}^{n}\sum_{l=1}^{k-1}\mathbf{1}\{y_{l,k}\neq\hat{y}_{l,k}\} \\
&= \sum_{k=m+1}^{n}\sum_{l=1}^{k-1}\mathbf{1}\{y_{l,k}\neq\hat{y}_{l,k}\} \\
&\leq \sum_{k=m+1}^{n}\sum_{l=1}^{k-1}P\{\text{Request } q_{l,k}|\text{labels to } q_{s\leq m,t\leq m}\} \\
&\leq \sum_{k=m+1}^{n}\sum_{l=1}^{k-1}\frac{2ad}{m^2} \\
&\leq \frac{2ad}{m^2}\frac{(n-m)(n+m+1)}{2} \\
&\leq ad\left(\frac{(n+1)^2}{m^2}-1\right).
\end{aligned}$$

where the third line assumes that every pairwise comparison that is ambiguous (that is, cannot be imputed using the knowledge gained from the first m objects) is incorrect. The fourth line follows from the application of Lemma 2.9 and Lemma 2.10. □

Combining Lemmas 2.14 and 2.15 in a straightforward way, we have the following theorem.

Theorem 2.16. *Assume* ***A1-2****, $\sigma\sim\mathcal{U}$, and $P(Y_{i,j}\neq y_{i,j})=p$. If $R=\Theta((1-2p)^{-2}\log n)$ and $\widehat{\sigma}$ is any ranking in $\Sigma_{n,d}$ that is consistent with all known pairwise comparisons between the subset of objects ranked in the output of the algorithm of Figure 2.3, then with constant probability $\mathbb{E}[d_\tau(\sigma,\widehat{\sigma})]=O(d(1-2p)^{-2}\log(n)/n)\binom{n}{2}$ and no more than $O(d(1-2p)^{-2}\log^2(n))$ pairwise comparisons are requested, on average.*

If we repeat the algorithm with different initializations of the objects until a sufficient

number of objects are ranked before an object is passed over, we can boost this constant probability to an arbitrarily high probability. However, in practice, we recommend running the algorithm just once to completion since we do not believe passing over an object early on greatly affects performance.

2.5 Empirical results

In this section we present empirical results for both the error-free algorithm of Figure 2.1 and the robust algorithm of Figure 2.3. For the error-free algorithm, $n = 100$ points, representing the objects to be ranked, were uniformly at random simulated from the unit hypercube $[0, 1]^d$ for $d = 1, 10, 20, \ldots, 100$. The reference was simulated from the same distribution. For each value of d the experiment was repeated 25 times using a new simulation of points and the reference. Because responses are error-free, exact identification of the ranking is guaranteed. The number of requested queries is plotted in Figure 2.5 with the lower bound of Theorem 2.5 for reference. The number of requested queries never exceeds twice the lower bound which agrees with the result of Theorem 2.11.

The robust algorithm of Figure 2.3 was evaluated using a symmetric similarity matrix dataset available at [37] whose (i, j)th entry, denoted $s_{i,j}$, represents the human-judged similarity between audio signals i and j for all $i \neq j \in \{1, \ldots, 100\}$. If we consider the kth row of this matrix, we can rank the other signals with respect to their similarity to the kth signal; we define $q_{i,j}^{(k)} := \{s_{k,i} > s_{k,j}\}$ and $y_{i,j}^{(k)} := \mathbf{1}\{q_{i,j}^{(k)}\}$. Since the similarities were derived from human subjects, the derived labels may be erroneous. Moreover, there is no possibility of repeating queries here and so the errors are persistent. The analysis of this dataset in [16] suggests that the relationship between signals can be well approximated

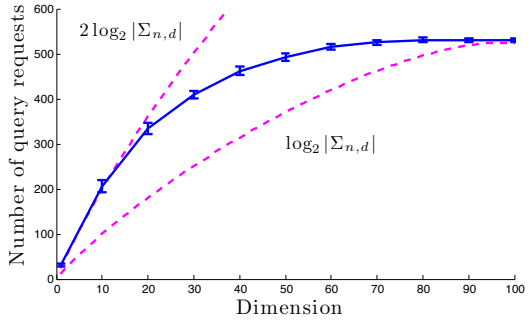


Figure 2.5: Mean and standard deviation of requested queries (solid) in the error-free case for $n = 100$; $\log_2|\Sigma_{n,d}|$ is a lower bound (dashed).

Table 2.1: Statistics for the algorithm robust to persistent errors of Section 2.4 with respect to all $\binom{n}{2}$ pairwise comparisons. Recall y is the noisy response vector, $\tilde{y}$ is the embedding's solution, and $\hat{y}$ is the output of the robust algorithm.

Dimension		2	3
% of queries requested	mean	14.5	18.5
	std	5.3	6
Average error	$d(y,\tilde{y})$	0.23	0.21
	$d(y,\hat{y})$	0.31	0.29

by an embedding in 2 or 3 dimensions. We used *non-metric multidimensional scaling* [19] to find an embedding of the signals: $x_1,\ldots,x_{100} \in \mathbb{R}^d$ for $d = 2$ and 3. For each object x_k, we use the embedding to derive pairwise comparison labels between all other objects as follows: $\tilde{y}_{i,j}^{(k)} := \mathbf{1}\{||x_k - x_i|| < ||x_k - x_j||\}$, which can be considered as the best approximation to the labels $y_{i,j}^{(k)}$ (defined above) in this embedding. The output of the robust sequential algorithm, which uses only a small fraction of the similarities, is denoted by $\hat{y}_{i,j}^{(k)}$. We set $R = 15$ using Theorem 2.16 as a rough guide. Using the popular Kendell-Tau distance $d(y^{(k)},\hat{y}^{(k)}) = \binom{n}{2}^{-1}\sum_{i<j}\mathbf{1}\{y_{i,j}^{(k)} \neq \hat{y}_{i,j}^{(k)}\}$ [35] for each object k, we denote the average of this metric over all objects by $d(y,\hat{y})$ and report this statistic and the number of queries requested in Table 2.1. Because the average error of $\hat{y}$ is only 0.07 higher than that of $\tilde{y}$, this suggests that the algorithm is doing almost as well as we could hope. Also, note that $2R\,2d\log n/\binom{n}{2}$ is equal to 11.4% and 17.1% for $d = 2$ and 3, respectively, which agrees well with the experimental values.

2.6 Discussion

This chapter considered a natural model for constraining the set of total orderings over a set of objects. By a counting argument we proved a lower bound on the query complexity of this problem and presented an algorithm that matches it up to constants. In addition, we considered the possibility that answers to pairwise comparisons were"noisy" or reversed with some probability less than one half and proposed a robust version of our algorithm to account for this uncertainty.

However, there are obstacles to overcome before something like the schemes proposed in this Chapter can be realized in practice. First, the algorithm is quite brittle in that if it makes a mistake early on, the mistake can cascade through the algorithm resulting in unpredictable behavior. The most likely way the algorithm could falter is by abiding by the model too strictly and not accounting for possible model mismatch. After all, the geometrical model is trying to model a possibly unknowable reality of someone's perception, so while it may be a reasonable model, it should be taken with a grain of salt and an algorithm should be robust to small perturbations of this model. The second obstacle to overcome is one of computation. By making "hard" decisions, i.e. deciding the direction of a pairwise comparison was absolutely one way or the other without any uncertainty, the task of identifying which queries were ambiguous or not and boiled down to a simple linear program. However, methods that make"soft" decisions and update those beliefs as more information becomes available tend to be much more robust [38]. Unfortunately, these statistical advantages come at a substantially higher computational cost making them infeasible for all but the most simple cases. While this chapter provided a theoretical foundation for active ranking, the question of how best to

realize it in practice remains open.

2.7 Bibliographical Remarks

The content of this chapter was based on the author's following publications:

- Kevin G Jamieson and Robert D Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2240–2248, 2011,

- Kevin G Jamieson and Robert D Nowak. Active ranking in practice: General ranking functions with sample complexity bounds. In *NIPS Workshop*, 2011.

Two lines of related research were performed around the time of the publication of this work.

The first related work considers a set of n objects and an arbitrary set of bits $S = \{y_{i,j}\}_{1 \le i < j \le n}$ that each represent the pairwise preference $y_{i,j} = \mathbf{1}\{i \prec j\}$. It is not assumed that there exists a ranking consistent with all $\binom{n}{2}$ pairwise preferences in S and one can define the loss of a total ordering π as $\ell(\pi, S) = \sum_{y_{i,j}=0} \mathbf{1}\{i \prec_\pi j\}$. It is shown in [5, 39] that using an adaptive sampling procedure one can find a ranking π such that $\ell(\pi, S) - \min_{\pi'} \ell(\pi', S) \le \epsilon$ using no more than $n \log(n) \text{poly}(\epsilon^{-1})$ pairwise comparisons with high probability, whereas $\Omega(n^2 \text{poly}(\epsilon^{-1}))$ are required if pairwise comparisons are chosen non-adaptively.

The work presented in this chapter is very relevant to nearest neighbor search or top-k nearest neighbor search when only pairwise comparisons are available. This is precisely the setting studied in [40] who introduce a complexity measure called the

combinatorial disorder coefficient which, in the context of this chapter, roughly measures how far the embedding of the objects differs from a one dimensional subspace. They show that the number of pairwise comparisons to identify a nearest neighbor using pairwise comparisons is polynomial in $D\log(n)$ where D is the combinatorial disorder coefficient and n is the number of objects. While this is reminiscent of the results presented here„ the combinatorial disorder coefficient cannot be directly mapped to this setting and the tools used there are significantly different.

Chapter 3

Active Non-metric Multidimensional Scaling

The main mathematical question of active ranking introduced in Chapter 2 was essentially the following: given $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$ and one additional point x_{n+1} whose location was not known, find the ranking $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that

$$||x_{\sigma(1)} - x_{n+1}||_2 < ||x_{\sigma(2)} - x_{n+1}||_2 < \cdots < ||x_{\sigma(n)} - x_{n+1}||_2$$

using as few binary queries or comparisons of the form $\{||x_i - x_{n+1}||_2 < ||x_j - x_{n+1}||_2\}$ as possible. In some sense, we are "adding" x_{n+1} to the embedding of n points since the possible location of x_{n+1} is seems highly constrained given that its location obeys the discovered ranking. Indeed, one could even consider starting with no points and adding points one at a time, at each time requiring the new point to rank all other points, and visa versa. If we knew that there existed an Euclidean embedding of n points in d dimensions that was consistent with all possible triplet queries of the form $\{||x_i - x_k||_2 < ||x_j - x_k||_2\}$ that we could ask, it is natural to wonder how many queries would it take to find *an* embedding of the n points in $\mathbb{R}^d$ that agrees with all the answers to all possible triplet queries. Active ranking hints that to add the $(k+1)$st object to the

embedding, it may only take just $O(d\log(k))$ queries, suggesting that finding a consistent ordinal embedding may only require $O(dn\log(n))$ queries. This chapter explores this hypothesis in-depth.

3.1 Introduction

We study the problem of learning a low-dimensional embedding from ordinal data. Consider a set of n points $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$. The locations of the points are unknown to us, but assume we are given the set of constraints of the form "object x_k is closer to (or further from) x_i than x_j" for all distinct $i, j, k \in \{1, \ldots, n\}$. The goal is to identify an embedding into $\mathbb{R}^d$ consistent with these constraints. This is a classic problem that has been addressed using a technique known as non-metric multidimensional scaling (non-metric MDS).

Here we consider a new variation on this problem. Constraints of the form above are often collected from human subjects. Each constraint is associated with a binary variable, the answer to the *comparison query* "Is object x_k closer to x_i than x_j?" People are better at providing this sort of information as opposed to giving more fine-grained numerical judgments or distances [2]. There are on the order of n^3 constraints. Collecting ordinal data of this type from people is time-consuming and costly, and quickly becomes prohibitive as n grows. So it is of interest to consider whether it is necessary to obtain the complete set of data. Since the points are assumed to exist in $\mathbb{R}^d$, it is reasonable to conjecture that if the embedding dimension is low, i.e., $d \ll n$, then there may be a high degree of redundancy in these constraints. If this conjecture is correct, then it should be possible to identify a *consistent embedding* (i.e., consistent will **all** the constraints) from

a small subset of the constraints.

In this work we lower bound the minimum number of constraints needed to determine a consistent embedding by $dn \log n$, far fewer than the total number. We conjecture that this lower bound is tight and propose a sequential procedure for adaptively selecting comparison queries. A comparison query is made if and only if the answer to the query (i.e., the corresponding constraint) is ambiguous given the answers to all previously selected queries. Ambiguity can be tested by solving an optimization problem that is, in general, non-convex but is observed to be well-behaved in practice (see Section 3.3.4). Analysis of the procedure and numerical experiments support the conjecture that on the order of $dn \log n$ queries/constraints determine the embedding. Furthermore, we show that if queries are selected uniformly at random, then almost all the queries must be requested in order to determine an embedding consistent with all the constraints.

3.1.1 Related work

Non-metric multidimensional scaling (MDS) was designed to provide researchers with a graphical or spatial representation of the human-perceived relationships between a set of arbitrary objects [41]. In addition to the pairwise comparisons of the form $||x_i - x_j|| < ||x_j - x_k||$ for all triples (i, j, k), non-metric MDS also forces constraints of the type

$$||x_i - x_j|| < ||x_k - x_l|| \tag{3.1}$$

for all quadruples $(i, j, k, l) \in \{1, \ldots, n\}$. These additional queries make the total number of queries grow like n^4 which is often prohibitively large for even small values of n.

Consequently these additional constraints are often omitted in practice [42, 43]. Also note that because our query-ambiguity test alluded to above is a special case of non-metric MDS, it follows that non-metric MDS is also non-convex and these additional queries can make the already difficult optimization even harder. However, these issues are not the only problems; they can also, at times, be difficult to answer accurately because a query "is the distance between objects i and j less than the distance between objects k and l?" requires a comparison of the absolute scales of the dissimilarities instead of simply asking which object is closer to another. This difficulty is our primary reason for considering constraints using just three objects.

While pairwise comparisons using triples of objects are very natural and easy to answer, some research suggests that people can find answering these kind of queries extremely tedious and boring. Presumably, this could lead to erroneous answers after extended sessions of querying a user [3]. Some researchers have suggested that perhaps only a sparse subset of these inequalities are actually required, greatly reducing the load on the human subject [42, 44]. Early work using just a random subset of these kinds of queries by Johnson supports this hypothesis [45]. While researchers in the past have proposed algorithms to find an embedding given a fixed number of answers to queries, we are unaware of any research that attempts to characterize the number of queries that must *necessarily* be made to uniquely determine an embedding. We provide partial answers to this question and propose an algorithm that we conjecture to be optimal in the sense that it asks within a constant factor of the minimum number of necessary queries to uniquely determine an embedding consistent with all the constraints.

Prior to this point, we have assumed that the constraints we are querying for are consistent with an embedding of a known dimension. However, [43] assumes that labels

to queries are the result of a consensus from a number of individuals, or a crowd. This perspective allows one to consider the problem from a probabilistic point of view so that one can speak of requesting the comparison that would provide the greatest potential information gain. While [43] presents some results for empirical datasets, few guarantees were made about the quality of the embedding and no guidance was given to how many queries were "enough" or sufficient to achieve an embedding of satisfactory quality.

As discussed in the prelude to this chapter, this problem is very related to the active ranking problem [6]. Given a fixed embedding $x_1, \ldots, x_k$ of k objects in $\mathbb{R}^d$ (i.e. the locations of the k objects are known exactly) and just one additional object is placed in an unknown location in the same space, it is shown that the ranking of the objects relative to their proximity to this new object can be discovered with just $\Theta(d \log k)$ queries on average, depending on the particular placement of the new object. Note that the embeddings we consider in this paper are determined up to an equivalence class by the correct ranking of the objects with respect to each object. While active ranking appears to give us a valuable tool set, we will see later that it cannot get us all the way to a sufficiency result. However, the active ranking analysis immediately yields a lower bound on the query complexity of finding an embedding, providing us with a sufficiency result.

3.2 The geometry of an embedding

Consider an embedding of n points in $\mathbb{R}^d$. For any triple (i, j, k), we have that either $||x_i - x_k|| < ||x_k - x_j||$ or its opposite are true (we assume ties are not allowed). We wish to learn an embedding $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that satisfies all of these constraints. If we

concatenate each point $x_i \in \mathbb{R}^d$ into a column vector $x = (x_1^T, \ldots, x_n^T)^T$, we see that an embedding of n points in d dimensions can be represented by a single vector living in $\mathbb{R}^{nd}$. If for every triple (i, j, k) we define the region $r_{i,j,k} = \{x \in \mathbb{R}^{nd} : ||x_i - x_k|| < ||x_k - x_j||\}$, then the query: "is object x_k closer to x_i or x_j" is equivalent to asking if $x \in r_{i,j,k}$. We call this pairwise comparison query a membership query. All possible intersections of these regions (and their complements) partition $\mathbb{R}^{nd}$ into many nd-dimensional regions which we will call nd-cells to distinguish them from the regions of the form of $r_{i,j,k}$. Because every point in an nd-cell agrees with all of the same constraints, we call any two embeddings in the same nd-cell *equivalent*. From this perspective, we see that we are trying to locate a point in one of these nd-cells bounded by surfaces passing through $\mathbb{R}^{nd}$ that are induced by the membership queries $r_{i,j,k}$. Before considering this problem directly, we would like to provide some intuition about the space of embeddings.

3.2.1 A lower bound on the query complexity

In this section we state a lower bound on the number of membership queries that are necessary to define an embedding of n objects in d dimensions such that all the constraints are satisfied. Our strategy is to add one object at a time to the embedding and lower bound how quickly the number of embeddings grows.

Recall that we assumed the existence of a fixed embedding of n points in d dimensions that generated the $n\binom{n-1}{2}$ constraints. Suppose we somehow had the exact locations of $k < n$ objects and we would simply like to add the $(k+1)$st point to the embedding. At the very least, we must determine the order of the distances from x_{k+1} to all the other x_i's for $i = 1, \ldots, k$. That is, we must determine some permutation σ of the k indices

such that we can write

$$||x_{k+1} - x_{\sigma(1)}|| < ||x_{k+1} - x_{\sigma(2)}|| < \cdots < ||x_{k+1} - x_{\sigma(k)}||\,. \tag{3.2}$$

Because the points $x_1, \ldots, x_k$ are embedded in d-dimensions, there are far fewer than $k!$ possibilities for σ. In fact, if all the points are in general position, the number of possibilities for σ is known exactly. Namely, Theorem 2.3 and Corollary 2.4 of Chapter 2 apply. We conclude from those results that if $Q(k, d)$ denotes the number of d-cells formed by the arrangement of hyperplanes induced by the $\binom{k}{2}$ pairs of objects, then $Q(k, d) = \Theta(\frac{n^{2d}}{2^d d!})$ where d is considered fixed. We are now ready to state a lower-bound on the query complexity of finding an embedding.

Theorem 3.1. *The number of membership queries $\{x \in r_{i,j,k}\}_{i,j,k \leq n}$ required to determine an embedding of n objects in d dimensions that satisfies all of the constraints is $\Omega(dn \log n)$.*

Proof. Using the help of an oracle, who will not only supply us the answers to membership queries but also additional side information, we will construct an embedding adding one object at a time. We will lower bound the number of bits of information that we will need to collect from the oracle and then use this as a lower bound for the number of queries necessary since a query provides at most one bit of information.

Recall that we assume the existence of a fixed embedding $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that generated the constraints. We begin by asking the oracle for the exact locations of the first two objects x_1 and x_2. Given the fixed positions of the first two objects, we find which d-cell (a single halfspace in this case) the third object resides in, tell the oracle, and then ask the oracle to provide the exact location of the third object. That is, before getting the exact location of an object from the oracle, we must tell the oracle

which d-cell the object is in. After k objects have been embedded this way, we find the d-cell that the $(k+1)$th object resides in, and then tell the oracle who returns the exact location of this object. Because the oracle is providing to us the exact locations of the objects, any queries that are inferred, due to them being unambiguous, are consistent with *any* embedding that satisfies all the $n\binom{n-1}{2}$ constraints; even those that have not been considered by this sequential procedure yet. This subtle point will be considered again in Section 3.3.2 when we do not have access to the exact locations of the objects. There are $Q(k,d)$ possible rankings of the k objects that we must discriminate between to tell which d-cell the $(k+1)$th object is in. Therefore, we must provide at least $\Omega(d \log k)$ bits of information to the oracle. The lower bound follows from summing the number of bits to add all of the objects to the embedding sequentially. □

Based on how the above lower bound was constructed, it is not clear how tight the bound is because the oracle provided the *exact* location of the current object: information which is clearly sufficient but unlikely to be absolutely necessary. However, as alluded to before in Section 3.1.1, theoretical and empirical evidence suggests that as the number of objects to be embedded grows, the amount of "wiggle" possible in each point of the embedding decreases rapidly to zero [46]. Intuitively, as the number of constraints grows with the number of objects embedded, the embedding acts more and more as if it were constrained with metric constraints. We will revisit this idea in the Section 3.4.

3.2.2 Counting the number of embeddings

Given the lower bound of the last section, that showed that the log of the number of embeddings is $\Omega(dn \log(n))$, it is natural to wonder how tight it is. If we could *upper*

bound the number of embeddings and look at the log of this number, this would tell us how many *bits* it takes to encode an embedding of n objects in d dimensions. If this number matched the lower bound, it would still not be enough to tell us if we could *achieve* the lower bound because the possible membership queries we have at our disposal may not be informative enough. However, it would not rule out, and give some hope that a query complexity of $O(dn\log(n)$ is achievable.

Consider the membership query $x \in r_{i,j,k} = \{x \in \mathbb{R}^{nd} : ||x_i - x_k|| < ||x_k - x_j||\}$ for some (i, j, k) triple. By squaring both sides of the inequality in the definition of $r_{i,j,k}$ we find that

$$r_{i,j,k} = \{x \in \mathbb{R}^{nd} : x_i^T x_i - 2x_i^T x_k - x_j^T x_j + 2x_j^T x_k < 0\}. \tag{3.3}$$

Note that the boundary of $r_{i,j,k}$ is given by the degree-2 polynomial in nd dimensions defined by $x_i^T x_i - 2x_i^T x_k - x_j^T x_j + 2x_j^T x_k = 0$, and that there are $n\binom{n-1}{2}$ of them. Inspired by a technique used to count the number of unique sign patters of a matrix when the underlying matrix is low-rank [47], we use the same result used there to count the number of embeddings. We restate the main lemma from there verbatim that is originally thanks to Warren [48].

Lemma 3.2. *[48] Let $P_1, \ldots, P_m$ be real polynomials in r variables and let C be the complement of the variety defined by $\prod_i P_i$, i.e. the set of points in which all the m polynomials are non-zero: $C = \{z \in \mathbb{R}^r : \forall i P_i(z) \neq 0\}$. If all m polynomials are of degree*

at most q, then then number of connected components of C is at most

$$2(2q)^r \sum_{i=0}^{r} 2^i \binom{m}{i} \leq \left(\frac{4eqm}{r}\right)^r$$

where the inequality holds when $m > r > 2$.

In the lemma, each distinct connected components defines an nd-cell. While multiple nd-cells may correspond to the same equivalent embedding, counting the number of nd-cells is still an upper bound on the number of embeddings.

Corollary 3.3. *The number of equivalent embeddings of n objects in d-dimensions if $n \geq d$ is no greater than $\left(\frac{4n}{\sqrt{d}}\right)^{2dn}$.*

Proof. The result follows from a direct application of Lemma 3.2 with $q = 2$, $r = nd$, and $m = n\binom{n-1}{2} < n^3/2$. □

The above corollary implies that it only takes $O(dn \log(n))$ bits to describe each equivalent embedding of n objects in d dimensions. We stress that this does not imply that there exists an algorithm that can discover the embedding in just this many queries because the queries may not provide enough information (e.g. less than a constant fraction of a bit). But the result does not rule out the existence of such a result.

3.2.3 The inefficiency of randomly selected queries

Before we discuss different adaptive methods of selecting queries, it is natural to wonder if such complicated schemes are really necessary; is it sufficient to simply select queries uniformly at random to find a solution? In this section we show that if membership queries are selected in a random fashion, $\Omega(n^3)$ queries must be requested to uniquely

determine an nd-cell and thus, an embedding. In fact, we actually show that to solve a problem using extra side information would require this many queries and because that information could have always been ignored, to solve the problem without the side information is *at least* as hard. Our strategy is to add a single object to the embedding one at a time and show that if there are k objects already embedded, it requires $\Omega(k^2)$ queries to add the $(k+1)$th object.

We assume that queries are selected independently such that after selecting a subset of the queries, they are exchangeable in the sense that we can reorder them any way we like and it does not affect which nd-cell they define. Enumerate the objects so that they are labeled $1, \ldots, n$. Then, order the randomly selected queries such that for any query defined over the triple (i, j, k) in the list, all queries that are ordered before it use objects whose indices are less than or equal to $\texttt{max}\{i, j, k\}$. In other words, we would like to reorder the selected queries such that it appears as if we are adding one object at a time like we constructed the lower bound of above. Again, suppose we somehow had the exact locations of $k < n$ objects and we would simply like to add the $(k+1)$th point to the embedding. At the very least, we must determine the order of the distances from x_{k+1} to all the other x_i's for $i = 1, \ldots, k$ as in (3.2). Recall from Section 3.2.1 that if the k objects are fixed, each possible ranking over the k objects has a one-to-one correspondence with a d-cell that is bounded by hyperplanes corresponding to the queries. If m queries were chosen uniformly at random from the possible $\binom{k}{2}$, the answers to m queries narrows the set of possible rankings to a d-cell in $\mathbb{R}^d$. This d-cell may consist of one or more of the d-cells in the partition induced by all $\binom{k}{2}$ hyperplanes. If it contains more than one of the partition cells, then the underlying ranking is ambiguous.

Lemma 3.4. *Let* $N = \binom{k}{2}$. *Suppose* m *membership queries* $\{x \in r_{i,k+1,j}\}_{i,j \leq k}$ *are*

chosen uniformly at random without replacement from the possible $\binom{k}{2}$*. Then for all positive integers* $N \geq m \geq d$ *the probability that the* m *queries yield a unique ranking is* $\binom{m}{d}/\binom{N}{d} \leq (\frac{em}{N})^d$.

Proof. No fewer than d hyperplanes bound each d-cell in the partition of $\mathbb{R}^d$ induced by all possible queries. The probability of selecting d specific queries in a random draw of m is equal to

$$\binom{N-d}{m-d} \Bigg/ \binom{N}{m} = \binom{m}{d} \Bigg/ \binom{N}{d} \leq \frac{m^d}{d!}\frac{d^d}{N^d} \leq \left(\frac{m}{N}\right)^d \frac{d^d}{d!} \leq \left(\frac{em}{N}\right)^d.$$

□

Note that $\binom{m}{d}/\binom{N}{d} < 1/2$ unless $m = \Omega(k^2)$. Therefore, if the queries are randomly chosen, then we will need to ask almost all queries to guarantee that the inferred ranking over the first k objects is probably correct. The proof of the next theorem is shown by repeated application of the above result using the same line of reasoning as the proof of Theorem 3.1.

Theorem 3.5. *Given the existence of an embedding of* n *objects in* d *dimensions, if* m *membership queries* $\{x \in r_{i,j,k}\}_{i,j,k \leq n}$ *are chosen uniformly at random without replacement from the possible* $n\binom{n-1}{2}$*, then to uniquely determine the* nd*-cell and thus an embedding that satisfies all of the constraints with probability greater than* $1/2$*,* $m = \Omega(n^3)$.

3.3 Query selection algorithms

In this section we propose query selection algorithms that attempt to satisfy all of the $n\binom{n-1}{2}$ constraints by only requesting a small subset of them. First, we review classical binary sort in Section 3.3.1 because it is implemented in all of the algorithms and its performance guarantees should be clearly stated. We then propose a sequential algorithm in Section 3.3.2 that adds one object at a time to the embedding and asks for queries only if they cannot be inferred using all the known constraints up to that time. Finally, we present a non-metric (or generalized) version of landmark MDS in Section 3.3.3 that was originally designed to reduce the amount of data collection for metric MDS [49]. Both algorithms assume the existence of a subroutine that, given any set of constraints that are consistent, will output whether there exists an embedding that is consistent with all of the constraints, or not. In addition, we assume that if such an embedding exists, we can request it from the subroutine. After presenting the algorithms that utilize this subroutine, we will consider its implementation in Section 3.3.4.

3.3.1 Binary Sort

Binary sort is a simple, adaptive algorithm that finds an arbitrary ranking over n objects using no more than $n \log_2 n$ pairwise comparisons. Because there are $n! = \Theta(n^n)$ possible rankings, this algorithm is optimal in terms of the number of requested pairwise comparisons if no additional structure about the objects is assumed. The algorithm works as follows: given a ranking of k objects, there are $(k+1)$ positions that the $(k+1)$th object can be put into; and because there is an ordering over the objects, binary search can be used to find the correct position in no more than $\log_2(k+1)$ queries. By induction,

no more than $n \log_2 n$ pairwise comparisons are needed to rank n objects.

Consider finding an embedding of n objects in d dimensions. An embedding is only unique up to the constraints $||x_i - x_j|| < ||x_j - x_k||$ for all triples (i, j, k). This is equivalent to having each object rank the other $n-1$ objects relative to their distance away from themselves, like in (3.2). By the above argument, to find n rankings of $(n-1)$ objects, no more than $n(n-1)\log_2(n-1)$ queries must be requested.

3.3.2 A sequential query selection algorithm

Here we introduce an algorithm to find an embedding of n objects in d dimensions that sequentially chooses membership queries in an adaptive way in hopes of drastically reducing the number of requested queries. But first, we consider a naïve approach to point out some potential pitfalls of a sequential algorithm.

Recall the sequential process used in the proof of the lower bound of Theorem 3.1. We added one object at a time by finding the d-cell the object was located in, and then requested the exact location of the object within that d-cell from the oracle. It is natural to wonder if this *exact* location is really necessary and if picking an *arbitrary* point in the d-cell would suffice. Unfortunately, as Borg illustrates in a non-pathological example of an embedding, this arbitrary placement of objects can potentially close off possibilities for the locations of future objects which would make it impossible to satisfy all the future constraints [41, Chapter 2]. Intuitively, if you are not careful with how you decide the coordinates of the objects, it is very easy to walk yourself into a corner with no escape. What we should take from this example is that we must allow for the objects to have maximum flexibility while obeying the constraints if we would like to guarantee that all

the constraints, in the end, are satisfied.

The underlying principle behind our proposed algorithm is very simple and has enjoyed great success in other active learning settings [50, 51]. The sequential algorithm for requesting queries begins by enumerating the objects and considers them one at a time. The algorithm will proceed through the queries using binary sort and request the membership query if only if it cannot be determined using the previously requested constraints. That is, if Q is the set of constraints corresponding to the membership queries we have requested up to the consideration of some new query $\{x \in r_{i,j,k}\}$, we will run our constraint-validating subroutine twice: once with $Q \cup \{x \in r_{i,j,k}\}$ and the second time with $Q \cup \{x \in r^c_{i,j,k}\}$. If the subroutine confirms that both set of constraints lead to valid embeddings, then the query in question $Q \cup \{x \in r_{i,j,k}\}$ is said to be *ambiguous*. Otherwise, if only one of the runs of the subroutine confirms a valid embedding, we can infer what the constraint must be and we do not need to request it from the user. This algorithm is presented in Figure 3.1. Note that despite what is written in the presentation of the algorithm in Figure 3.1, binary sort is implemented; it is presented this way for clarity.

Given the full set of $n\binom{n-1}{2}$ constraints from the algorithm, we can then run the subroutine to get the full embedding of the n objects in d dimensions.

3.3.3 Landmark non-metric MDS (LNM-MDS)

Here we introduce landmark non-metric MDS (LNM-MDS) which can be thought of as a non-metric or generalized version of landmark *metric* MDS [49]. The basic idea behind landmark-based versions of MDS is that instead of collecting data for all pairs

Sequential query selection algorithm

input: n objects in unknown positions in $\mathbb{R}^d$
initialize: $Q = \emptyset$, enumerate objects $x_1, \ldots, x_n$ in uniformly random order
for $k = \{2, \ldots, n\}$
  for $j = \{1, \ldots, k\}$
    for $i = \{1, \ldots, k\}$
      if $\{x \in r_{i,j,k}\}$ is *ambiguous* using only Q,
        ask if $\{x \in r_{i,j,k}\}$;
      else
        infer if $\{x \in r_{i,j,k}\}$ with Q and add it to Q
output: $n\binom{n-1}{2}$ constraints

Figure 3.1: Sequential algorithm for selecting queries. See Section 3.3.2 for the definition of an ambiguous query.

or triples of objects, a small number of objects are designated as landmarks. The objects are embedded using only distances (or comparisons, in this paper) relative to the landmarks. For example, the LNM-MDS proposed here only uses comparisons of the form $||x_i - l|| < ||x_j - l||$ or $||l - x_i|| < ||l' - x_i||$, where x_i and x_j are arbitrary objects, but l and l' must be members of a small set of landmarks. If $d \ll n$ and the number of landmarks is large enough, then the intuition is that using these landmarks may be sufficient to describe the same information as if all information was collected between all objects.

For any integer $L \geq 2$, LNM-MDS chooses L objects uniformly at random from the set of n and requests only the queries between the objects so that each landmark has a complete ranking over the other $n - 1$ objects and each non-landmark object has a ranking over the L landmarks. This algorithm is motivated by the idea that if the the dimension d is not too big, perhaps the relative proximities to just a small subset of the objects suffices to define the embedding. While these rankings define $L\binom{n-1}{2} + (n - L)\binom{L}{2}$

total pairwise comparisons, we will use binary sort to acquire these rankings which would mean only about $L(n-1)\log_2(n-1) + (n-L)L\log_2(L)$ will be requested. If the number of landmarks is small, this could be a significant savings in the number of requested queries compared to asking for all n rankings over $n-1$ objects, about $n^2 \log_2 n$. While LNM-MDS does not explicitly take advantage of the low-dimensional nature of the embedding, it may implicitly use it to its advantage because a few landmarks may suffice to define the embedding. One of the drawbacks of this algorithm is that to ensure that all the constraints are satisfied, one must check if all the other queries not asked are unambiguous. If landmarks are added one at a time, this could be very computationally demanding.

3.3.4 Constraint validation subroutine

This section describes the constraint validation subroutine that determines whether a query is ambiguous or not. This subroutine is, in essence, an algorithm for non-metric MDS that just uses constraints of triples of objects as input. As described in the beginning of Section 3.2, to check if a set of constraints is valid, we must check if there is non-empty intersection of the sets defined by the membership queries $r_{i,j,k}$.

In general, to find a point in $\mathbb{R}^p$ that lies in the intersection of sets is known as a feasibility program [52]. Unfortunately, it is easy to show that the sets defined by the membership queries, or equivalently the constraints of (3.3), produce non-convex sets which makes the feasibility program non-convex. This implies that what the constraint validation subroutine converges to could be a local minima (if it converges at all) which may erroneously indicate that a set of constraints do *not* correspond to a valid embedding

when they really do. Clearly, this could be disastrous to the algorithm because queries may be indicated as unambiguous when they really are ambiguous. Some algorithms for solving non-metric MDS deal with this non-convexity by allowing d to be variable (in contrast to fixed), possibly as large as n, but penalizing the optimization by adding the trace norm of the inner product matrix of the embedding. This encourages low-dimensional (or approximately low-dimensional) embeddings [42]. Because essentially arbitrary constraints can be obeyed if d is allowed to be n, this sort of approach would not constrain the set of solutions and would indicate that almost all the queries are ambiguous. What this means is that solving the non-convex program is unavoidable and the only thing that can be done is to repeat the optimization multiple times, each with a random initialization. If this is done enough times, we can be relatively confident that its results are trust-worthy. Fortunately, in practice, this optimization problem tends to converge to a solution rather easily if it exists. We will return to this issue in the presentation of our numerical results.

The earliest reference of an algorithm that attempts to do the job of the subroutine is credited to Johnson in 1973 who solves the feasibility problem by penalizing any violated inequalities with a quadratic loss function [45]. In the last few decades there have been enormous advances in optimization and we know that a linear loss function using Lagrange multipliers leads to much quicker convergence [52, 53]. To make the optimization problem converge in a reasonable amount of time for the problem sizes we are considering ($3 \leq n \leq 50$) many known techniques and tricks for non-convex optimization are necessary [53]. Matlab code of our implementation is available upon request.

3.4 Empirical Results

In this section we present empirical results regarding how many queries are requested to embed n objects into d dimensions. We compare standard binary sort of Section 3.3.1, the sequential algorithm of Section 3.3.2, and LNM-MDS of Section 3.3.3. In LNM-MDS, recall from Section 3.3.3 that to check if an existing solution given some number of landmarks satisfies all the constraints, we have to check if any of the queries not requested are ambiguous or not. Because we will be adding one landmark at a time, we will give the algorithm the benefit of the doubt in our simulations and end LNM-MDS as soon as it finds an embedding using the fewest number of landmarks with zero violations of all the constraints (even those that it may not know about yet). Clearly, this is a lower bound on its performance. On the other hand, when either binary sort or the sequential algorithm of Figure 3.1 finishes, it guarantees that all the constraints are satisfied (under the assumption that the subroutine always returns correct results).

Recall from Section 3.3.4 that the constraint-validating subroutine is solving a non-convex problem. It is possible that the subroutine will converge to a local minima, indicating that there does not exist an embedding consistent with the given constraints when, actually, there does. This behavior could potentially lead to the algorithm believeing that a query is unambiguous when, in reality, it is the opposite case and must be requested. For our simulations, we assumed that if the algorithm failed to converge to a consistent embedding after 3 attempts, then a consistent embedding did not exist. Fortunately, in our studies, with the number of restarts set to 3, in each run we observed no more than about a few of these mistakes out of the total of about 3000 considered for $n = 30$. However, this seemingly disastrous problem is actually not much of a problem

at all because in practice, the only queries fed to the constraint validation subroutine are those that were ambiguous when they were considered (we do not need to give the algorithm unambiguous constraints because by definition, their labels were determined by the constraints already in the subroutine, which were ambiguous.) This means that if a query is erroneously indicated as unambiguous, it is not added to the optimization problem and thus does not constrain the solution. Because we expect many queries to be redundant, it is even possible that we will infer the true label of this query with queries requested in the future.

For our experiments, we chose $d = \{1, 2, 3\}$ with $n = \{3, \ldots, 30\}$. Note that because binary sort is implemented in both our sequential algorithm and LNM-MDS, neither algorithm can do worse than binary sort, which requests about $n^2 \log_2 n$ queries, regardless of how large d is. All experiments were repeated just 5 times in the interest of time. In Figure 3.2 we have plotted the mean and standard deviation of the number of requested queries using error bars for the sequential algorithm of Figure 3.1 in blue, LNM-MDS in black, and binary sort in red. Clearly, LNM-MDS performs nearly as bad as binary sort (but can never perform as badly because binary sort is implemented for all the rankings in LNM-MDS). LNM-MDS was only run for $d = 1, 2$ because it was clear from just these results that LNM-MDS was not exploiting the fact that $d \ll n$. It is also clear that the sequential algorithm requests significantly fewer membership queries than either binary sort or LNM-MDS.

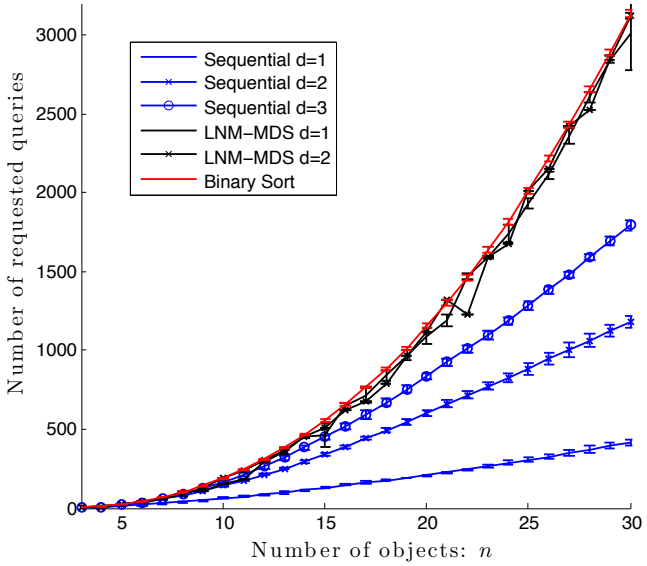


Figure 3.2: The mean number of requested membership queries to determine all the constraints of an embedding of n objects in d dimensions using the three algorithms described in Section 3.3. The standard deviation of the trials are presented using error bars.

Analysis of Empirical Results

From just Figure 3.2, for a fixed dimension d, it is unclear how the number of queries grows with n; is it more like $n^2 \log n$ or $n \log n$? It is our conjecture that it grows like the latter. In this section we will analyze the empirical data more closely and also point out some theoretical results that, together, we believe provide strong evidence to support our conjecture.

Consider how many queries are requested when adding just a single object to the embedding. Under the hypothesis that the number of queries for the sequential algorithm grows like $n \log n$ times some constant depending on the dimension, we should observe that the number of queries required to add just a single object should be no greater than order $\log n$. If the hypothesis is false and the number of queries actually grows faster

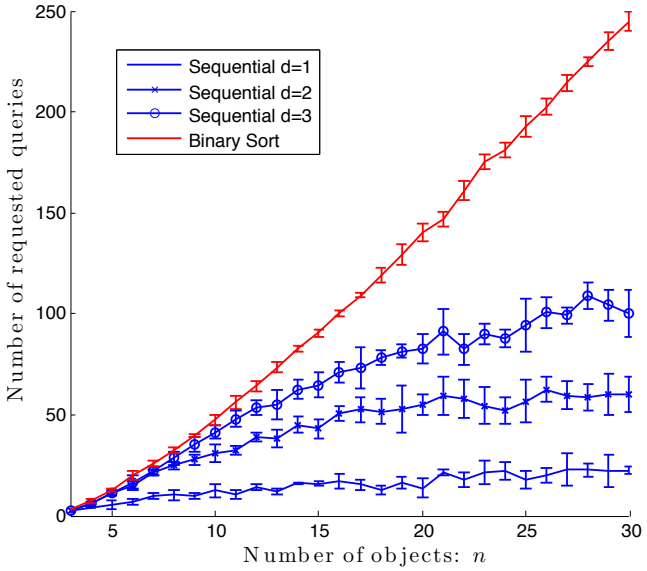


Figure 3.3: Given all the constraints between $(n-1)$ objects in d dimensions, the mean number of requested membership queries to determine the all the constraints of n objects in d dimensions. The standard deviation of the trials are presented using error bars.

than this, like $n^2 \log n$, the number of queries requested to add just a single object should grow like $n \log n$. Figure 3.3 presents the average number of queries required to add just the kth object for $k = 3, \ldots, 30$ and $d = 1, 2, 3$ for the sequential algorithm in blue and for binary sort in red. It is clear that this the quantity associated with the sequential algorithm grows sub-linearly and perhaps even reasonable to conjecture that it grows logarithmically. This behavior can be explained by some previous analyses of non-metric multidimensional scaling and the previous analysis of the ranking problem alluded to earlier.

If we consider an embedding of n objects in d dimensions that satisfies all of the constraints, we know that this embedding lives in some nd-cell and therefore has some amount of flexibility. In related studies, this amount of flexibility is observed to decrease rapidly to zero as n grows. For example, at least qualitatively, the amount of flexibility

in an embedding in 2 dimensions has been observed to be negligible for n as small as 10 or 15 using similar constraints to those discussed here [46, 54]. So as $k < n$ becomes very large, adding the $(k+1)$th object becomes more and more like adding an object to a *fixed* embedding of k objects. Recall that the embedding is constrained only so far as forcing each object to rank the other objects with respect to their relative proximity. To add the $(k+1)$th object to the embedding, we must discover how the $(k+1)$th object ranks the other k objects, and how the k objects insert the $(k+1)$th object into their ranking. In previous work, we showed that if the positions of the first k objects are fixed and known, and we have discovered how the $(k+1)$th object has ranked some subset of $j < k$ objects, it requires only about d/j pairwise comparisons, in expectation, to insert the $(j+1)$th object into the ranking [51, Lemma 4]. It follows that to discover how the $(k+1)$th object ranks all k objects, it requires only about $d \log k$ queries. This predicts part of the story, but we still must consider how many queries it requires to insert the $(k+1)$th object in to the rankings of the other $1, \ldots, k$ objects.

As k gets very large, the size of the d-cells corresponding to the possible ways the $(k+1)$th object can rank the first k objects (see Section 3.2) becomes very small, something like on the order k^{-2d}. What this means is that if we first locate the $(k+1)$th object in this tiny cell, with respect to the other objects, *it* looks fixed. This means that to these other objects, it looks as if they are simply adding a fixed object to their ranking which takes only about d/k queries. Using these informal approximations, we should expect that only about $d \log k + k \times d/k \approx d \log k$ queries will be requested to add the $(k+1)$th object. Repeated application of this argument and the observation that embeddings appear more and more fixed as $n \to \infty$, we conjecture with some level of confidence that the algorithm of Section 3.3.2 requests no more than $O(dn \log n)$ queries to uniquely

define an embedding of n objects in d dimensions.

3.5 Discussion

The previous section provided some support for the conjecture that the number of queries required to embed n objects in d dimensions grows no faster than $O(dn \log n)$. This would be consistent with the required number of bits to specify an embedding, as calculated in Section 3.2.2 when we upper bounded the number of equivalent embeddings. But, of course, this is just a conjecture. Future work will attempt to prove this conjecture.

While we have assumed throughout that the n objects embed into exactly d dimensions with no violations of the inequalities, this assumption should never be expected to be true in practice, especially when humans provide the query responses. While the sequential algorithm described here can easily be made robust to only probably-correct query responses by paying an additional $\log n$ multiplicative factor in the number of requested queries using the techniques developed in Chapter 2, this still does not resolve the problem that the model may be wrong. Any practical implementation of adaptive non-metric multidimensional scaling must be robust to a certain degree of mismatch between the perception of humans and the best d dimensional representation of the objects.

3.6 Bibliographical Remarks

The work presented in this chapter was largely based off of the author's publication

- Kevin G Jamieson and Robert D Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Communication, Control, and Computing*

(Allerton), 2011 49th Annual Allerton Conference on, pages 1077–1084. IEEE, 2011

however, the content of Section 3.2.2 is novel to this thesis.

Part II

Pure Exploration for Multi-armed Bandits

Chapter 4

Stochastic Best-arm Identification

In Part 1 of this thesis, it was shown that the query complexity of a problem can be dramatically reduced if the problem exhibits some low-dimensional structure that could be taken advantage of. It was also shown in Chapter 2 that the algorithm considered there could be made robust to random errors in the answers to queries, the result of flipping the binary answers with some known, fixed probability $p < 1/2$, by repeatedly sampling the answer to the same query for a number of trials dependent on the constant p. The result of which allows us to confidently state that the majority votes of the answers is correct with probability at least $1 - \delta$. By repeating this for N different encountered queries, one has that all of them are simultaneously correct with probability at least $1 - N\delta$. We see that the probability of failure increases linearly with N, the number of queries before the algorithm is terminated. It is natural to wonder if such a scaling in the probability of failure is unavoidable.

To study this subtle problem and others like it, we turn to the simple and unstructured setting of multi-armed bandits. This framework allows us to ignore the complexities of the low-dimensional structure and focus purely on the statistical problems. In this chapter we study a problem so easy to state and fundamental to sequential decision making that it is remarkable that it was not solved until recently: given n biased coins, what is the fewest number of total flips necessary to identify the coin with the highest

probability of heads with probability at least $1 - \delta$?

4.1 Introduction

This chapter introduces a new algorithm for the *best arm* problem in the stochastic multi-armed bandit (MAB) setting. Consider a MAB with n arms, each with unknown mean payoff $\mu_1, \ldots, \mu_n$ in $[0, 1]$. A sample of the ith arm is an independent realization of a sub-Gaussian random variable with mean μ_i. In the *fixed confidence setting*, the goal of the best arm problem is to devise a sampling procedure with a single input δ that, regardless of the values of $\mu_1, \ldots, \mu_n$, finds the arm with the largest mean with probability at least $1 - \delta$. More precisely, best arm procedures must satisfy $\sup_{\mu_1,\ldots,\mu_n} \mathbb{P}(\widehat{i} \neq i^*) \leq \delta$, where i^* is the best arm, $\widehat{i}$ an estimate of the best arm, and the supremum is taken over all set of means such that there exists a unique best arm. In this sense, best arm procedures must automatically adjust sampling to ensure success when the mean of the best and second best arms are arbitrarily close. Contrast this with the *fixed budget setting* where the total number of samples remains a constant and the confidence in which the best arm is identified within the given budget varies with the setting of the means. While the fixed budget and fixed confidence settings are related (see [55] for a discussion) this work focuses on the fixed confidence setting only.

4.1.1 Related Work

The best arm problem has a long history dating back to the '50s with the work of [56, 57]. In the fixed confidence setting, the last decade has seen a flurry of activity providing new upper and lower bounds. In 2002, the *successive elimination* procedure

of [58] was shown to find the best arm with order $\sum_{i \neq i^*} \Delta_i^{-2} \log(n\Delta_i^{-2})$ samples, where $\Delta_i = \mu_{i^*} - \mu_i$, coming within a logarithmic factor of the lower bound for any algorithm of $\sum_{i \neq i^*} \Delta_i^{-2}$, shown in 2004 in [59]. For reference, a lower bound of $n \max_{i \neq i_*} \Delta_i^{-2}$ can be shown for any non-adaptive method, exposing the gap between adaptive and non-adaptive methods for this problem [11]. A similar bound to the bound of [59] was also obtained using a procedure known as *LUCB1* that was originally designed for finding the m-best arms [60]. Recently, [11] proposed a procedure called *PRISM* which succeeds with $\sum_i \Delta_i^{-2} \log\log\left(\sum_j \Delta_j^{-2}\right)$ or $\sum_i \Delta_i^{-2} \log\left(\Delta_i^{-2}\right)$ samples depending on the parameterization of the algorithm, improving the result of [58] by at least a factor of $\log(n)$. The best sample complexity result for the fixed confidence setting comes from a procedure similar to PRISM, called *exponential-gap elimination* [61], which guarantees best arm identification with high probability using order $\sum_i \Delta_i^{-2} \log\log \Delta_i^{-2}$ samples, coming within a doubly logarithmic factor of the lower bound of [59]. While the authors of [61] conjecture that the $\log\log$ term cannot be avoided, it remained unclear as to whether the upper bound of [61] or the lower bound of [59] was loose.

The classic work of [62] answers this question. It shows that the doubly logarithmic factor is necessary, implying that order $\sum_i \Delta_i^{-2} \log\log \Delta_i^{-2}$ samples are necessary and sufficient in the sense that no procedure can satisfy $\sup_{\Delta_1,\ldots,\Delta_n} \mathbb{P}(\widehat{i} \neq i^*) \leq \delta$ and use fewer than $\sum_i \Delta_i^{-2} \log\log \Delta_i^{-2}$ samples in expectation for all $\Delta_1, \ldots, \Delta_n$. The doubly logarithmic factor is a consequence of the law of the iterated logarithm (LIL) [63]. The LIL states that if X_ℓ are i.i.d. sub-Gaussian random variables with $\mathbb{E}[X_\ell] = 0$, $\mathbb{E}[X_\ell^2] = \sigma^2$

and we define $S_t = \sum_{\ell=1}^{t} X_\ell$ then

$$\limsup_{t\to\infty} \frac{S_t}{\sqrt{2\sigma^2 t \log\log(t)}} = 1 \quad \text{and} \quad \liminf_{t\to\infty} \frac{S_t}{\sqrt{2\sigma^2 t \log\log(t)}} = -1$$

almost surely. Here is the basic intuition behind the lower bound. Consider the two-arm problem and let Δ be the difference between the means. In this case, it is reasonable to sample both arms equally and consider the sum of differences of the samples, which is a random walk with drift Δ. The deterministic drift crosses the LIL bound above when $t\,\Delta = \sqrt{2t \log\log t}$. Solving this equation for t yields $t \approx 2\Delta^{-2} \log\log \Delta^{-2}$. This intuition will be formalized in Section 4.2.

4.1.2 Motivation

The LIL also motivates a novel approach to the best arm problem. Specifically, the LIL suggests a natural scaling for confidence bounds on empirical means, and we follow this intuition to develop a new algorithm for the best-arm problem. The algorithm is an Upper Confidence Bound (UCB) procedure [64] based on a finite sample version of the LIL. The new algorithm, called lil'UCB, is described in Figure 4.1. By explicitly accounting for the log log factor in the confidence bound and using a novel stopping criterion, our analysis of lil'UCB avoids taking naive union bounds over time, as encountered in some UCB algorithms [60, 65], as well as the wasteful "doubling trick" often employed in algorithms that proceed in epochs, such as the PRISM and exponential-gap elimination procedures [11,58,61]. Also, in some analyses of best arm algorithms the upper confidence bounds of each arm are designed to hold with high probability for all arms uniformly, incurring a $\log(n)$ term in the confidence bound as a result of the necessary union bound

over the n arms [58, 60, 65]. However, our stopping time allows for a tighter analysis so that arms with larger gaps are allowed larger confidence bounds than those arms with smaller gaps where higher confidence is required. Like exponential-gap elimination, lil'UCB is order optimal in terms of sample complexity.

It is easy to show that without the stopping condition (and with the right δ) our algorithm achieves a cumulative regret of the same order as standard UCB. Thus for the expert it may be surprising that such an algorithm can achieve optimal sample complexity for the best arm identification problem given the lower bound of [66]. As it was empirically observed in the latter paper there seems to be a transient regime, before this lower bound applies, where the performance in terms of best arm identification is excellent. In some sense the results in the present paper can be viewed as a formal proof of this transient regime: if stopped at the right time performance of UCB for best arm identification is near-optimal (or even optimal for lil'UCB).

One of the main motivations for this work was to develop an algorithm that exhibits great practical performance in addition to optimal sample complexity. While the sample complexity of exponential-gap elimination is optimal up to constants, and PRISM up to small $\log\log$ factors, the empirical performance of these methods is rather disappointing, even when compared to non-sequential sampling. Both PRISM and exponential-gap elimination employ *median elimination* [58] as a subroutine. Median elimination is used to find an arm that is within $\epsilon > 0$ of the largest, and has sample complexity within a constant factor of optimal for this subproblem. However, the constant factors tend to be quite large, and repeated applications of median elimination within PRISM and exponential-gap elimination are extremely wasteful. On the contrary, lil'UCB does not invoke wasteful subroutines. As we will show, in addition to having the best theoretical

sample complexities bounds known to date, lil'UCB also exhibits superior performance in practice with respect to state-of-the-art algorithms.

4.2 Lower Bound

Before introducing the lil'UCB algorithm, we show that the log log factor in the sample complexity is necessary for best-arm identification. It suffices to consider a two armed bandit problem with a gap Δ. If a lower bound on the gap is unknown, then the log log factor is necessary, as shown by the following result.

Theorem 4.1. *Consider the best arm problem in the fixed confidence setting with $n = 2$, difference between the two means Δ, and expected number of samples $\mathbb{E}_\Delta[T]$. Any procedure with $\sup_{\Delta \neq 0} \mathbb{P}(\widehat{i} \neq i^*) \leq \delta$, $\delta \in (0, 1/2)$, then has*

$$\limsup_{\Delta \to 0} \frac{\mathbb{E}_\Delta[T]}{\Delta^{-2} \log \log \Delta^{-2}} \geq 2 - 4\delta.$$

Proof. The proof follows readily from Theorem 1 of [62] that considers the deviations of a biased random walk. By considering a reduction of the best arm problem with $n = 2$ in which the value of one arm is known. In this case, the only strategy available is to sample the other arm some number of times to determine if it is less than or greater than the known value. □

Theorem 4.1 implies that in the fixed confidence setting, no best arm procedure can have $\sup \mathbb{P}(\widehat{i} \neq i^*) \leq \delta$ *and* use fewer than $(2 - 4\delta) \sum_i \Delta_i^{-2} \log \log \Delta_i^{-2}$ samples in expectation for all Δ_i.

In brief, the result of Farrell [62] follows by showing a generalized sequential probability ratio test, which compares the running empirical mean of X after t samples against a series of thresholds, is an optimal test. In the limit as t increases, if the thresholds are not at least $\sqrt{(2/t)\log\log(t)}$ then the LIL implies the procedure will fail with probability approaching $1/2$ for small values of Δ. Setting the thresholds to be just greater than $\sqrt{(2/t)\log\log(t)}$, in the limit, one can show the expected number of samples must scale as $\Delta^{-2}\log\log\Delta^{-2}$. As the proof in [62] is quite involved, we provide a short argument for a slightly simpler result in the original publication of this work [9].

Since the original publication of this work, other finite-time law-of-the-iterated-logarithm bounds have appeared in the literature [67, 68]. In particular, a very strong lower bound was proven in [68] that implies that the above bound on the number of measurements also holds with high probability, in addition to just in expectation. This is very satisfying as it corresponds to our upper bounds that hold with high probability.

4.3 Algorithm and Analysis

This section introduces lil'UCB. The procedure operates by sampling the arm with the largest upper confidence bound; the confidence bounds are defined to account for the implications of the LIL. The procedure terminates when one of the arms has been sampled more than a constant times the number of samples collected from all other arms combined. Fig. 4.1 details the algorithm and Theorem 4.2 quantifies performance. In what follows, let $X_{i,s}$, $s = 1, 2, \ldots$ denote independent samples from arm i and let $T_i(t)$ denote the number of times arm i has been sampled up to time t. Define $\widehat{\mu}_{i,T_i(t)} := \frac{1}{T_i(t)}\sum_{s=1}^{T_i(t)} X_{i,s}$ to be the empirical mean of the $T_i(t)$ samples from arm i up to time t. The algorithm of

Fig. 4.1 assumes that the centered realizations of the ith arm are sub-Gaussian[1] with known scale parameter σ.

lil' UCB

input: confidence $\delta > 0$, algorithm parameters ε, λ, $\beta > 0$
initialize: sample each of the n arms once, set $T_i(t) = 1$ for all i and set $t = n$
while $T_i(t) < 1 + \lambda \sum_{j \neq i} T_j(t)$ for all i

sample arm

$$I_t = \operatorname*{argmax}_{i \in \{1,\ldots,n\}} \left\{ \widehat{\mu}_{i,T_i(t)} + (1+\beta)(1+\sqrt{\varepsilon}) \sqrt{\frac{2\sigma^2(1+\varepsilon)\log\left(\frac{\log((1+\varepsilon)T_i(t))}{\delta}\right)}{T_i(t)}} \right\}.$$

set $T_i(t+1) = T_i(t) + 1$ if $I_t = i$, otherwise set $T_i(t+1) = T_i(t)$.
else stop and output $\arg\max_{i \in \{1,\ldots,n\}} T_i(t)$

Figure 4.1: The lil' UCB algorithm.

Define

$$\mathbf{H}_1 = \sum_{i \neq i^*} \frac{1}{\Delta_i^2} \quad \text{and} \quad \mathbf{H}_3 = \sum_{i \neq i^*} \frac{\log\log_+(1/\Delta_i^2)}{\Delta_i^2}$$

where $\log\log_+(x) = \log\log(x)$ if $x \geq e$, and 0 otherwise. Our main result is the following.

Theorem 4.2. *For $\epsilon \in (0,1)$, let $c_\epsilon = \frac{2+\epsilon}{\epsilon}(1/\log(1+\epsilon))^{1+\epsilon}$ and fix $\delta \in (0, \log(1+\epsilon)/(ec_\epsilon))$. Then for any $\beta \in (0,3]$, there exists a constant $\lambda > 0$ such that with probability at least $1 - 4\sqrt{c_\epsilon \delta} - 4c_\epsilon\delta$ lil' UCB stops after at most $c_1\mathbf{H}_1\log(1/\delta) + c_3\mathbf{H}_3$ samples and outputs the optimal arm, where $c_1, c_3 > 0$ are known constants that depend only on $\epsilon, \beta, \sigma^2$.*

Note that the algorithm obtains the optimal query complexity of $\mathbf{H}_1 \log(1/\delta) + \mathbf{H}_3$ up to constant factors. We remark that the theorem holds with any value of λ satisfying (4.7). Inspection of (4.7) shows that as $\delta \to 0$ we can let λ tend to $\left(\frac{2+\beta}{\beta}\right)^2$. We point out

[1] A zero-mean random variable X is said to be sub-Gaussian with scale parameter σ if for all $t \in \mathbb{R}$ we have $\mathbb{E}[\exp\{tX\}] \leq \exp\{\sigma^2 t^2/2\}$. If $a \leq X \leq b$ almost surely than it suffices to take $\sigma^2 = (b-a)^2/4$.

that the sample complexity bound in the theorem can be optimized by choosing ϵ and β. For a setting of these parameters in a way that is more or less faithful to the theory, we recommend taking $\epsilon = 0.01$, $\beta = 1$, and $\lambda = \left(\frac{2+\beta}{\beta}\right)^2$. For improved performance in practice, we recommend applying footnote 2 and setting $\epsilon = 0$, $\beta = 0.5$, $\lambda = 1 + 10/n$ and $\delta \in (0, 1)$, which do not meet the requirements of the theorem, but work very well in our experiments presented later. We prove the theorem via two lemmas, one for the total number of samples taken from the suboptimal arms and one for the correctness of the algorithm. In the lemmas we give precise constants.

4.3.1 Proof of Theorem 4.2

Before stating the two main lemmas that imply the result, we first present a finite form of the law of iterated logarithm. This finite LIL bound is necessary for our analysis and may also prove useful for other applications.

Lemma 4.3. *Let $X_1, X_2, \ldots$ be i.i.d. centered sub-Gaussian random variables with scale parameter σ. For any $\epsilon \in (0, 1)$ and $\delta \in (0, \log(1+\epsilon)/e)$[2] one has with probability at least $1 - \frac{2+\varepsilon}{\varepsilon}\left(\frac{\delta}{\log(1+\varepsilon)}\right)^{1+\epsilon}$ for all $t \geq 1$,*

$$\sum_{s=1}^{t} X_s \leq (1+\sqrt{\varepsilon})\sqrt{2\sigma^2(1+\varepsilon)t\log\left(\frac{\log((1+\varepsilon)t)}{\delta}\right)}.$$

Proof. We denote $S_t = \sum_{s=1}^{t} X_s$, and $\psi(x) = \sqrt{2\sigma^2 x \log\left(\frac{\log(x)}{\delta}\right)}$. We also define by induction the sequence of integers (u_k) as follows: $u_0 = 1$, $u_{k+1} = \lceil (1+\epsilon)u_k \rceil$.

[2] *Note δ is restricted to guarantee that $\log(\frac{\log((1+\varepsilon)t)}{\delta})$ is well defined. This makes the analysis cleaner but in practice one can allow the full range of δ by using $\log(\frac{\log((1+\varepsilon)t+2)}{\delta})$ instead and obtain the same theoretical guarantees.*

Step 1: Control of $S_{u_k}, k \geq 1$. The following inequalities hold true thanks to an union bound together with Chernoff's bound, the fact that $u_k \geq (1+\epsilon)^k$, and a simple sum-integral comparison:

$$\begin{aligned}\mathbb{P}\left(\exists k \geq 1 : S_{u_k} \geq \sqrt{1+\varepsilon}\ \psi(u_k)\right) &\leq \sum_{k=1}^{\infty} \exp\left(-(1+\varepsilon)\log\left(\tfrac{\log(u_k)}{\delta}\right)\right) \\ &\leq \sum_{k=1}^{\infty} \left(\tfrac{\delta}{k\log(1+\varepsilon)}\right)^{1+\epsilon} \leq \left(1+\tfrac{1}{\varepsilon}\right)\left(\tfrac{\delta}{\log(1+\varepsilon)}\right)^{1+\epsilon}.\end{aligned}$$

Step 2: Control of $S_t, t \in (u_k, u_{k+1})$. Adopting the notation $[n] = \{1, \ldots, n\}$, recall that Hoeffding's maximal inequality[3] states that for any $m \geq 1$ and $x > 0$ one has

$$\mathbb{P}(\exists\ t \in [m] \text{ s.t. } S_t \geq x) \leq \exp\left(-\tfrac{x^2}{2\sigma^2 m}\right).$$

Thus the following inequalities hold true (by using trivial manipulations on the sequence (u_k)):

$$\begin{aligned}&\mathbb{P}\Big(\exists\ t \in \{u_k+1, \ldots, u_{k+1}-1\} : S_t - S_{u_k} \geq \sqrt{\epsilon}\ \psi(u_{k+1})\Big) \\ &= \mathbb{P}\left(\exists\ t \in [u_{k+1}-u_k-1] : S_t \geq \sqrt{\epsilon}\ \psi(u_{k+1})\right) \leq \exp\left(-\epsilon\tfrac{u_{k+1}}{u_{k+1}-u_k-1}\log\left(\tfrac{\log(u_{k+1})}{\delta}\right)\right) \\ &\leq \exp\left(-(1+\epsilon)\log\left(\tfrac{\log(u_{k+1})}{\delta}\right)\right) \leq \left(\tfrac{\delta}{(k+1)\log(1+\varepsilon)}\right)^{1+\epsilon}.\end{aligned}$$

Step 3: By putting together the results of Step 1 and Step 2 we obtain that with probability at least $1 - \frac{2+\varepsilon}{\varepsilon}\left(\frac{\delta}{\log(1+\varepsilon)}\right)^{1+\epsilon}$, one has for any $k \geq 0$ and any $t \in \{u_k +$

[3]It is an easy exercise to verify that Azuma-Hoeffding holds for martingale differences with sub-Gaussian increments, which implies Hoeffding's maximal inequality for sub-Gaussian distributions.

$1, \ldots, u_{k+1}\}$,

$$\begin{aligned} S_t &= S_t - S_{u_k} + S_{u_k} \\ &\leq \sqrt{\varepsilon}\ \psi(u_{k+1}) + \sqrt{1+\varepsilon}\ \psi(u_k) \\ &\leq \sqrt{\varepsilon}\ \psi((1+\varepsilon)t) + \sqrt{1+\varepsilon}\ \psi(t) \\ &\leq (1+\sqrt{\varepsilon})\ \psi((1+\varepsilon)t), \end{aligned}$$

which concludes the proof. □

Without loss of generality we assume that $\mu_1 > \mu_2 \geq \ldots \geq \mu_n$. To shorten notation we denote

$$U(t,\omega) = (1+\sqrt{\varepsilon})\sqrt{\tfrac{2\sigma^2(1+\varepsilon)}{t}\log\left(\tfrac{\log((1+\varepsilon)t)}{\omega}\right)}.$$

The following events will be useful in the analysis:

$$\mathcal{E}_i(\omega) = \{\forall t \geq 1, |\widehat{\mu}_{i,t} - \mu_i| \leq U(t,\omega)\}$$

where $\widehat{\mu}_{i,t} = \frac{1}{t}\sum_{j=1}^{t} x_{i,j}$. Note that Lemma 4.3 shows $\mathbb{P}(\mathcal{E}_i(\omega)^c) = O(\omega)$. The following inequalities will also be useful and their proofs can be found in Appendix 4 (the second one is derived from the first inequality and the fact that $\frac{x+a}{x+b} \leq \frac{a}{b}$ for $a \geq b$, $x \geq 0$). For $t \geq 1$, $\epsilon \in (0,1)$, $c > 0$, $0 < \omega \leq 1$,

$$\frac{1}{t}\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right) \geq c \Rightarrow t \leq \frac{1}{c}\log\left(\frac{2\log((1+\varepsilon)/(c\omega))}{\omega}\right), \tag{4.1}$$

and for $t \geq 1$, $s \geq 3$, $\epsilon \in (0,1)$, $c \in (0,1]$, $0 < \omega \leq \delta \leq e^{-e}$,

$$\frac{1}{t}\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right) \geq \frac{c}{s}\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right) \text{ and } \omega \leq \delta \Rightarrow t \leq \frac{s}{c}\frac{\log\left(2\log\left(\frac{1}{c\omega}\right)/\omega\right)}{\log(1/\delta)}. \tag{4.2}$$

Lemma 4.4. *Let β, ϵ, δ be set as in Theorem 4.2 and let $\gamma = 2(2+\beta)^2(1+\sqrt{\varepsilon})^2\sigma^2(1+\varepsilon)$ and $c_\epsilon = \frac{2+\varepsilon}{\varepsilon}\left(\frac{1}{\log(1+\varepsilon)}\right)^{1+\epsilon}$. Then we have with probability at least $1 - 2c_\epsilon\delta$ and any $t \geq 1$,*

$$\sum_{i=2}^{n} T_i(t) \leq n + 5\gamma\mathbf{H}_1\log(e/\delta) + \sum_{i=2}^{n}\gamma\frac{\log(2\max\{1, \log(\gamma(1+\varepsilon)/\Delta_i^2/\delta)\})}{\Delta_i^2}.$$

The proof relies crucially on the fact that the realizations from each arm are independent of each other. This means that if we condition on the event that the realizations from the optimal arm are well-behaved, it is shown that the number of times the ith suboptimal arm is pulled is an independent sub-exponential random variable with mean on the order of $\Delta_i^{-2}\log(\log(\Delta_i^{-2})/\delta)$. We then apply a standard tail bound to the sum of independent sub-exponential random variables to obtain the result.

Proof. We decompose the proof in two steps.

Step 1. Let $i > 1$. Assuming that $\mathcal{E}_1(\delta)$ and $\mathcal{E}_i(\omega)$ hold true and that $I_t = i$ one has

$$\mu_i + U(T_i(t),\omega) + (1+\beta)U(T_i(t),\delta) \geq \widehat{\mu}_{i,T_i(t)} + (1+\beta)U(T_i(t),\delta) \geq \widehat{\mu}_{1,T_1(t)} + (1+\beta)U(T_1(t),\delta) \geq \mu_1,$$

which implies $(2+\beta)U(T_i(t), \min(\omega,\delta)) \geq \Delta_i$. If $\gamma = 2(2+\beta)^2(1+\sqrt{\varepsilon})^2\sigma^2(1+\varepsilon)$ then

using (4.1) with $c = \frac{\Delta_i^2}{\gamma}$ one obtains that if $\mathcal{E}_1(\delta)$ and $\mathcal{E}_i(\omega)$ hold true and $I_t = i$ then

$$\begin{aligned} T_i(t) &\leq \frac{\gamma}{\Delta_i^2} \log\left(\frac{2\log(\gamma(1+\varepsilon)/\Delta_i^2/\min(\omega,\delta))}{\min(\omega,\delta)}\right) \\ &\leq \tau_i + \frac{\gamma}{\Delta_i^2}\log\left(\frac{\log(e/\omega)}{\omega}\right) \leq \tau_i + \frac{2\gamma}{\Delta_i^2}\log\left(\frac{1}{\omega}\right), \end{aligned}$$

where $\tau_i = \frac{\gamma}{\Delta_i^2}\log\left(\frac{2\max\{1,\log(\gamma(1+\varepsilon)/\Delta_i^2/\delta)\}}{\delta}\right)$.

Since $T_i(t)$ only increases when i is played the above argument shows that the following inequality is true for any time $t \geq 1$:

$$T_i(t)\mathbb{1}\{\mathcal{E}_1(\delta) \cap \mathcal{E}_i(\omega)\} \leq 1 + \tau_i + \frac{2\gamma}{\Delta_i^2}\log\left(\frac{1}{\omega}\right). \tag{4.3}$$

Step 2. We define the following random variable:

$$\Omega_i = \max\{\omega \geq 0 : \mathcal{E}_i(\omega) \text{ holds true}\}.$$

Note that Ω_i is well-defined and by Lemma 4.3 it holds that $\mathbb{P}(\Omega_i < \omega) \leq c_\epsilon \omega$ where $c_\epsilon = \frac{2+\varepsilon}{\varepsilon}\left(\frac{1}{\log(1+\varepsilon)}\right)^{1+\epsilon}$. Furthermore one can rewrite (4.3) as

$$T_i(t)\mathbb{1}\{\mathcal{E}_1(\delta)\} \leq 1 + \tau_i + \frac{2\gamma}{\Delta_i^2}\log\left(\frac{1}{\Omega_i}\right). \tag{4.4}$$

We use this equation as follows:

$$\begin{aligned}\mathbb{P}\left(\sum_{i=2}^{n} T_i(t) > x + \sum_{i=2}^{n}(\tau_i+1)\right) &\leq c_\epsilon\delta + \mathbb{P}\left(\sum_{i=2}^{n} T_i(t) > x + \sum_{i=2}^{n}(\tau_i+1)\,\middle|\,\mathcal{E}_1(\delta)\right)\\ &\leq c_\epsilon\delta + \mathbb{P}\left(\sum_{i=2}^{n}\frac{2\gamma}{\Delta_i^2}\log\left(\frac{1}{\Omega_i}\right) > x\right). \qquad (4.5)\end{aligned}$$

Let $Z_i = \frac{2\gamma}{\Delta_i^2}\log\left(\frac{c_\epsilon^{-1}}{\Omega_i}\right)$, $i \in [n]\setminus 1$. Observe that these are independent random variables and since $\mathbb{P}(\Omega_i < \omega) \leq c_\epsilon\omega$ it holds that $\mathbb{P}(Z_i > x) \leq \exp(-x/a_i)$ with $a_i = 2\gamma/\Delta_i^2$. Using standard techniques to bound the sum of sub-exponential random variables one directly obtains that

$$\mathbb{P}\left(\sum_{i=2}^{n}(Z_i - a_i) \geq z\right) \leq \exp\left(-\min\left\{\frac{z^2}{4\|a\|_2^2}, \frac{z}{4\|a\|_\infty}\right\}\right) \leq \exp\left(-\min\left\{\frac{z^2}{4\|a\|_1^2}, \frac{z}{4\|a\|_1}\right\}\right). \qquad (4.6)$$

Putting together (4.5) and (4.6) with $z = 4\|a\|_1\log(1/(c_\epsilon\delta))$, $x = z + ||a||_1\log(ec_\epsilon)$ one obtains

$$\mathbb{P}\left(\sum_{i=2}^{n} T_i(t) > \sum_{i=2}^{n}\left(\frac{4\gamma\log(e/\delta)}{\Delta_i^2} + \tau_i + 1\right)\right) \leq 2c_\epsilon\delta,$$

which concludes the proof. □

Lemma 4.5. *Let β, ϵ, δ be set as in Theorem 4.2 and let $c_\epsilon = \frac{2+\varepsilon}{\varepsilon}\left(\frac{1}{\log(1+\varepsilon)}\right)^{1+\epsilon}$. If*

$$\lambda \geq \frac{1+\frac{\log\left(2\log\left(\left(\frac{2+\beta}{\beta}\right)^2/\delta\right)\right)}{\log(1/\delta)}}{1-(c_\epsilon\delta)-\sqrt{(c_\epsilon\delta)^{1/4}\log(1/(c_\epsilon\delta))}}\left(\frac{2+\beta}{\beta}\right)^2, \qquad (4.7)$$

then for all $i = 2, \ldots, n$ and $t = 1, 2, \ldots,$ we have $T_i(t) < 1 + \lambda\sum_{j\neq i} T_j(t)$ with probability at least $1 - 2c_\epsilon\delta + 4\sqrt{c_\epsilon\delta}$.

Note that the right hand side of (4.7) can be bound by a universal constant for all allowable δ which leads to the simplified statement of Theorem 4.2. Moreover, for any $\nu > 0$ there exists a sufficiently small $\delta \in (0,1)$ such that the right hand side of (4.7) is less than or equal to $(1+\nu)\left(\frac{2+\beta}{\beta}\right)^2$.

Essentially, the proof relies on the fact that given any two arms $j < i$ (i.e. $\mu_j \geq \mu_i$), $T_i(t)$ cannot be larger than a constant times $T_j(t)$ with probability at least $1-\delta$. Considering this fact, it is reasonable to suppose that the probability that $T_i(t)$ is larger than a constant times $\sum_{j=1}^{i-1} T_j(T)$ is decreasing exponentially fast in i. Consequently, our stopping condition is not based on a uniform confidence bound for all arms. Rather, it is based on confidence bounds that grow in size as the arm index i increases.

Proof. We decompose the proof in two steps.

Step 1. Let $i > j$. Assuming that $\mathcal{E}_i(\omega)$ and $\mathcal{E}_j(\delta)$ hold true and that $I_t = i$ one has

$$\begin{aligned}\mu_i + U(T_i(t),\omega) + (1+\beta)U(T_i(t),\delta) &\geq \widehat{\mu}_{i,T_i(t)} + (1+\beta)U(T_i(t),\delta) \\ &\geq \widehat{\mu}_{j,T_j(t)} + (1+\beta)U(T_j(t),\delta) \geq \mu_j + \beta U(T_j(t),\delta),\end{aligned}$$

which implies $(2+\beta)U(T_i(t),\min(\omega,\delta)) \geq \beta U(T_j(t),\delta)$. Thus using (4.2) with $c = \left(\frac{\beta}{2+\beta}\right)^2$ one obtains that if $\mathcal{E}_i(\omega)$ and $\mathcal{E}_j(\delta)$ hold true and $I_t = i$ then

$$T_i(t) \leq \left(\frac{2+\beta}{\beta}\right)^2 \frac{\log\left(2\log\left(\left(\frac{2+\beta}{\beta}\right)^2/\min(\omega,\delta)\right)/\min(\omega,\delta)\right)}{\log(1/\delta)} T_j(t).$$

Similarly to Step 1 in the proof of Lemma 4.4 we use the fact that $T_i(t)$ only increases when I_t is played and the above argument to obtain the following inequality for any time

$t \geq 1$:

$$(T_i(t)-1)\mathbb{1}\{\mathcal{E}_i(\omega)\cap\mathcal{E}_j(\delta)\} \leq \left(\tfrac{2+\beta}{\beta}\right)^2 \frac{\log\left(2\log\left(\left(\tfrac{2+\beta}{\beta}\right)^2/\min(\omega,\delta)\right)/\min(\omega,\delta)\right)}{\log(1/\delta)} T_j(t). \tag{4.8}$$

Step 2. Using (4.8) with $\omega = \delta^{i-1}$ we see that

$$\mathbb{1}\{\mathcal{E}_i(\delta^{i-1})\}\frac{1}{i-1}\sum_{j=1}^{i-1}\mathbb{1}\{\mathcal{E}_j(\delta)\} > 1-\alpha \;\Rightarrow\; (1-\alpha)(T_i(t)-1) \leq \kappa \sum_{j\neq i} T_j(t)$$

where $\kappa = \left(\tfrac{2+\beta}{\beta}\right)^2 \left(1 + \frac{\log\left(2\log\left(\left(\tfrac{2+\beta}{\beta}\right)^2/\delta\right)\right)}{\log(1/\delta)}\right)$. This implies the following, using that $\mathbb{P}(\mathcal{E}_i(\omega)) \geq 1 - c_\epsilon\omega$,

$$\begin{aligned}
&\mathbb{P}\left(\exists\ (i,t) \in \{2,\dots,n\}\times\{1,\dots\} : (1-\alpha)(T_i(t)-1) \geq \kappa\sum_{j\neq i} T_j(t)\right)\\
&\leq \mathbb{P}\left(\exists\ i \in \{2,\dots,n\} : \mathbb{1}\{\mathcal{E}_i(\delta^{i-1})\}\frac{1}{i-1}\sum_{j=1}^{i-1}\mathbb{1}\{\mathcal{E}_j(\delta)\} \leq 1-\alpha\right)\\
&\leq \sum_{i=2}^{n}\mathbb{P}(\mathcal{E}_i(\delta^{i-1}) \text{ does not hold}) + \sum_{i=2}^{n}\mathbb{P}\left(\frac{1}{i-1}\sum_{j=1}^{i-1}\mathbb{1}\{\mathcal{E}_j(\delta)\} \leq 1 - c_\epsilon\delta - (\alpha - c_\epsilon\delta)\right).
\end{aligned}$$

Let $\delta' = c_\epsilon\delta$. Note that by a simple Hoeffding's inequality and a union bound one has

$$\mathbb{P}\left(\frac{1}{i-1}\sum_{j=1}^{i-1}\mathbb{1}\{\mathcal{E}_j(\delta)\} \leq 1-\delta'-(\alpha-\delta')\right) \leq \min((i-1)\delta', \exp(-2(i-1)(\alpha-\delta')^2),$$

and thus if we define $j_* = \lceil \delta'^{-1/4}/2 \rceil$ we obtain with the above calculations

$$\mathbb{P}\left(\exists\,(i,t) \in \{2,\dots,n\} \times \{1,\dots\} : \left(1-\delta'-\sqrt{\delta'^{1/4}\log(1/\delta')}\right)(T_i(t)-1) \geq \kappa \sum_{j\neq i} T_j(t)\right)$$
$$\leq \sum_{i=2}^{n}\left(\delta'^{i-1} + \min\left((i-1)\delta', e^{-2(i-1)\delta'^{1/4}\log\left(\frac{1}{\delta'}\right)}\right)\right) \leq \frac{\delta'}{1-\delta'} + \delta' j_*^2 + \frac{e^{-2j_*\delta'^{1/4}\log\left(\frac{1}{\delta'}\right)}}{1-e^{-2\delta'^{1/4}\log\left(\frac{1}{\delta'}\right)}}$$
$$\leq \frac{\delta'}{1-\delta'} + \tfrac{9}{4}\delta'^{1/2} + \tfrac{3}{2}\delta'^{3/4} \leq 2c_\epsilon\delta + 4\sqrt{c_\epsilon\delta}.$$

□

Treating ϵ, σ^2 and factors of $\log\log(\beta)$ as constants, Lemma 4.4 says that the total number of times the suboptimal arms are sampled does not exceed $(\beta+2)^2\left(c_1\mathbf{H}_1\log(1/\delta) + c_3\mathbf{H}_3\right)$. Lemma 4.5 states that only the optimal arm will meet the stopping condition with $\lambda = c_\lambda\left(\frac{2+\beta}{\beta}\right)^2$ for some c_λ constant defined in the lemma. Combining these results, we observe that the total number of times all the arms are sampled does not exceed $(\beta+2)^2\left(c_1\mathbf{H}_1\log(1/\delta) + c_3\mathbf{H}_3\right)\left(1 + c_\lambda\left(\frac{2+\beta}{\beta}\right)^2\right)$, completing the proof of the theorem. We also observe using the approximation $c_\lambda = 1$, the optimal choice of $\beta \approx 1.66$.

4.4 Implementation and Simulations

In this section we investigate how the state of the art methods for solving the best arm problem compare to lil'UCB in practice. But first, we review the different strategies for identifying the best arm and provide intuition about how they work.

4.4.1 Review of Best-arm Identification Strategies

Most popular best-arm algorithms can be described by essentially one of three kinds of algorithm.

- **Action Elimination (AE) algorithm** - [11, 56, 57, 58, 61] Maintaining a set Ω_k for $k = 1, 2, \ldots$ initialized as $\Omega_1 = [n]$, these algorithms proceed in epochs by sampling the arms indexed by Ω_k a predetermined number of times r_k, and updated to Ω_{k+1} based on the rule:

$$\Omega_{k+1} = \{i \in \Omega_k : \hat{\mu}_{a,T_a(t)} - B_{a,T_a(t)} < \hat{\mu}_{i,T_i(t)} + B_{i,T_i(t)}\}$$

  where $a \in \Omega_k$ is a reference arm (for instance $a = \arg\max_{i\in[n]} \hat{\mu}_{i,T_i(t)} + B_{i,T_i(t)}$) and $B_{i,T_i(t)}$ is a confidence bound that describes the deviation of the empirical mean from its true mean (for instance, (4.11)). The algorithm terminates when $|\Omega_k| = 1$ and outputs the single element of Ω_k.

  In any action elimination algorithm, every arm must be sufficiently sampled before it can be decided with high probability that it is the best arm or not. This kind of algorithm simply keeps sampling all the arms and throws those arms out that it is confident are not the best arm.

- **Upper Confidence Bound (UCB) algorithm** - [55, 65] These algorithms begin by sampling all arms once. For each each time $t > n$ the algorithm samples the arm indexed by

$$\arg\max_{i\in[n]} \hat{\mu}_{i,T_i(t)} + \alpha B_{i,T_i(t)}$$

  where α is some constant and $B_{i,T_i(T)}$ is an appropriately chosen confident bound.

One stopping condition (c.f. [55, 65]) is to stop when

$$\hat{\mu}_{h_t,T_{h_t}(t)} - B_{h_t,T_{h_t}(t)} > \hat{\mu}_{\ell_t,T_{\ell_t}(t)} + B_{\ell_t,T_{\ell_t}(t)} \tag{4.9}$$

and output h_t. Alternatively, as is proposed and shown to work in this manuscript, one can stop when

$$\exists i \in [n] \ : \ T_i(t) > \alpha \sum_{j \neq i} T_j(t) \tag{4.10}$$

and output $\arg\max_i T_i(t)$ for some $\alpha > 0$.

While UCB sampling strategies were originally designed for the regret setting to optimize "exploration versus exploitation" [64], it was shown in [65] that UCB strategies were also effective in the pure exploration (find the best) setting. These algorithms are attractive because they are more sequential than the AE algorithms that tend to act more like uniform sampling for the first several epochs.

- **LUCB (a variation on UCB)** - [60, 69] Sample all arms once. For each time $t > n$ sample the arms indexed by h_t and ℓ_t (i.e. at each time t two arms are sampled) and stop when the criterion defined in (4.9) is met.

  While the LUCB and UCB sampling strategies appear to be only subtly different, the LUCB strategies appear to be better designed for exploration than UCB sampling strategies. For instance, given just two arms, the most reasonable strategy would be to sample both arms the same number of times until a winner could be confidently proclaimed, which is what LUCB would do. On the other hand, UCB strategies would tend to sample the best arm far more than the second-best arm

leading to a strategy that seems to emphasize exploitation over pure exploitation.

If the same confidence bound $B_{i,T_i(t)}$ is used in the analysis of all three algorithms, as is done in [10] using the LIL bound proved in this manuscript, then the overall sample complexity bounds of the *action elimination*, *UCB*, and *LUCB* strategies are very similar, even up to constants. For the very simple case of just $n = 6$ Gaussian arms with linearly decreasing means: $\{1, 4/5, 3/5, 2/5, 1/5, 0\}$ and input confidence $\delta = 0.1$, we have plotted in Figure 4.2 the empirical probability $\mathbb{P}(I_t = i)$ at every time t over 5000 trials where I_t is the index of the arm played by each algorithm at time t. The specific definitions of the algorithms can be found in [10] but they are essentially tuned versions of the above archetypal algorithms. We immediately observe a dramatic difference between the three sampling procedures: the action elimination strategy peels one arm away at a time and the plot of $\mathbb{P}(I_t = i)$ gives little indication of the best arm until many pulls in. On the other hand, the plot of $\mathbb{P}(I_t = i)$ for the LUCB and UCB sampling strategies clearly identifies the best arm very quickly with a large separation between the first and second arm. We remark that these algorithms may vary in performance using different parameters but the qualitative shape of these curves remain the same.

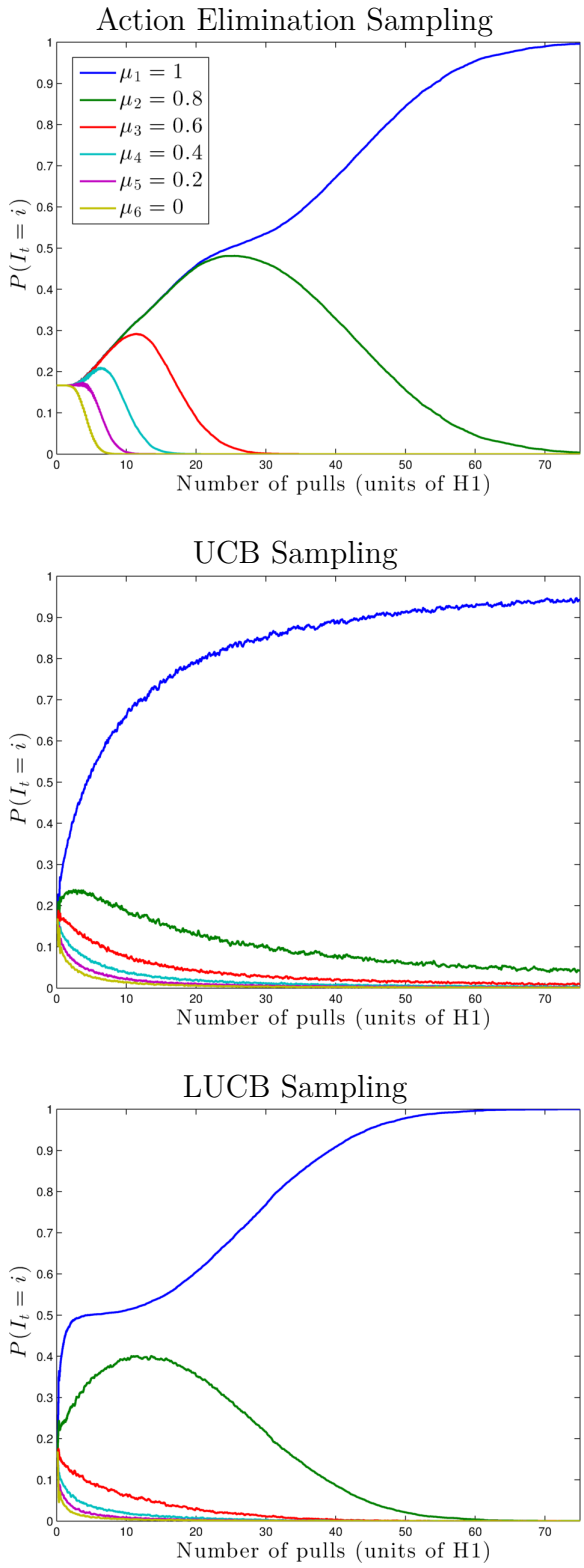


Figure 4.2: Comparison of the sampling strategies for the three main types of best-arm identification algorithms for $n = 6$ arms.

4.4.2 An Empirical Performance Comparison

Before describing each of the specific algorithms in the comparison against lil'UCB, we briefly describe an LIL-based stopping criterion alluded to above that can be applied to any of the algorithms.

LIL Stopping (LS) : For any algorithm and $i \in [n]$, after the t-th time we have that the i-th arm has been sampled $T_i(t)$ times and accumulated a mean $\widehat{\mu}_{i,T_i(t)}$. We can apply Lemma 4.3 (with a union bound) so that with probability at least $1 - \frac{2+\varepsilon}{\varepsilon}\left(\frac{\delta}{\log(1+\varepsilon)}\right)^{1+\epsilon}$

$$\left|\widehat{\mu}_{i,T_i(t)} - \mu_i\right| \leq B_{i,T_i(t)} := (1+\sqrt{\varepsilon})\sqrt{\frac{2\sigma^2(1+\varepsilon)\log\left(\frac{2\log((1+\varepsilon)T_i(t)+2)}{\delta/n}\right)}{T_i(t)}} \tag{4.11}$$

for all $t \geq 1$ and all $i \in [n]$. We may then conclude that if $\widehat{i} := \arg\max_{i\in[n]} \widehat{\mu}_{i,T_i(t)}$ and $\widehat{\mu}_{\widehat{i},T_{\widehat{i}}(t)} - B_{\widehat{i},T_{\widehat{i}}(t)} \geq \widehat{\mu}_{j,T_j(t)} + B_{j,T_j(t)}\ \forall j \neq \widehat{i}$ then with high probability we have that $\widehat{i} = i_*$.

The LIL stopping condition is somewhat naive but often quite effective in practice for smaller size problems when $\log(n)$ is negligible. To implement the strategy for any algorithm with fixed confidence ν, simply run the algorithm with $\nu/2$ in place of ν and assign the other $\nu/2$ confidence to the LIL stopping criterion. Note that to for the LIL bound to hold with probability at least $1-\nu$, one should use $\delta = \log(1+\epsilon)\left(\frac{\nu\varepsilon}{2+\varepsilon}\right)^{1/(1+\varepsilon)}$. The algorithms compared were:

- *Nonadaptive + LS* : Draw a random permutation of $[n]$ and sample the arms in an order defined by cycling through the permutation until the LIL stopping criterion is met. This is in some sense the most naive action elimination strategy.

- *Exponential-Gap Elimination (+LS)* [61] : This action elimination procedure proceeds in stages where at each stage, *median elimination* [58] is used to find an ϵ-optimal reference arm whose mean is guaranteed (with large probability) to be within a specified $\epsilon > 0$ of the mean of the best arm, and then arms are discarded if their empirical mean is sufficiently below the empirical mean of the ϵ-optimal arm. The algorithm terminates when there is only one arm that has not yet been discarded (or when the LIL stopping criterion is met).

- *Successive Elimination* [58] : This action elimination procedure proceeds in the same spirit as *Exponential-Gap Elimination* except the ϵ-optimal arm is equal to $\widehat{i} := \arg\max_{i \in [n]} \widehat{\mu}_{i,T_i(t)}$.

- *lil'UCB (+LS)* : The UCB procedure of Figure 4.1 is run with $\epsilon = 0.01$, $\beta = 1$, $\lambda = (2+\beta)^2/\beta^2 = 9$, and $\delta = \frac{(\sqrt{1+\nu(/2)}-1)^2}{4c_\epsilon}$ for input confidence ν. The algorithm terminates according to Fig. 4.1 (or when the LIL stopping criterion is met). Note that δ is defined as prescribed by Theorem 4.2 but we approximate the leading constant in (4.7) by 1 to define λ.

- *lil'UCB Heuristic* : The UCB procedure of Figure 4.1 is run with $\epsilon = 0$, $\beta = 1/2$, $\lambda = 1 + 10/n$, and $\delta = \nu/5$ for input confidence ν. These parameter settings do not satisfy the conditions of Theorem 4.2, and thus there is no guarantee that this algorithm will find the best arm.

- *LUCB1 (+ LS)* [60] : This LUCB procedure pulls two arms at each time: the arm with the highest empirical mean and the arm with the highest upper confidence bound among the remaining arms. The upper confidence bound was of the form

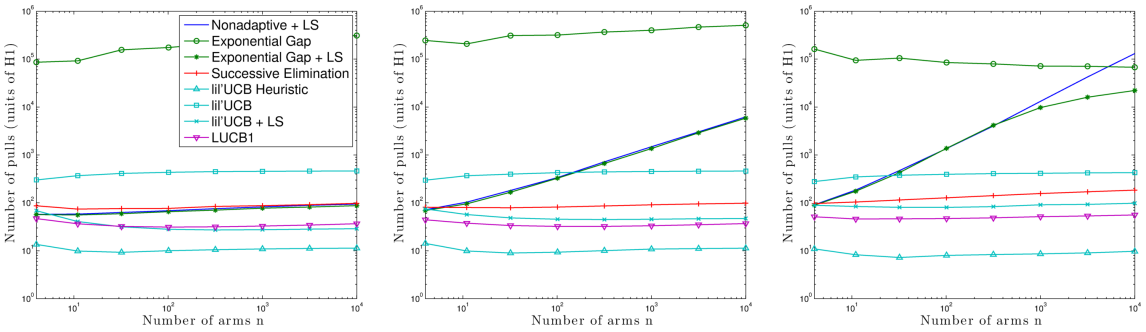
prescribed in the simulations section of [69] and is guaranteed to return the arm with the highest mean with confidence $1 - \delta$.

We did not compare to the action elimination strategy known as *PRISM* of [11] because the algorithm and its empirical performance are very similar to *Exponential-Gap Elimination* so its inclusion in the comparison would provide very little added value. We remark that the first three algorithms require $O(1)$ amortized computation per time step, the lil'UCB algorithms require $O(\log(n))$ computation per time step using smart data structures[4], and LUCB1 requires $O(n)$ computation per time step. LUCB1 was not run on all problem sizes due to poor computational scaling with respect to the problem size.

Three problem scenarios were considered over a variety problem sizes (number of arms). The "1-sparse" scenario sets $\mu_1 = 1/2$ and $\mu_i = 0$ for all $i = 2, \ldots, n$ resulting in a hardness of $\mathbf{H}_1 = 4n$. The "$\alpha = 0.3$" and "$\alpha = 0.6$" scenarios consider $n + 1$ arms with $\mu_0 = 1$ and $\mu_i = 1 - (i/n)^\alpha$ for all $i = 1, \ldots, n$ with respective hardnesses of $\mathbf{H}_1 \approx 3/2n$ and $\mathbf{H}_1 \approx 6n^{1.2}$. That is, the $\alpha = 0.3$ case should be about as hard as the sparse case with increasing problem size while the $\alpha = 0.6$ is considerably more challenging and grows super linearly with the problem size. See [11] for an in-depth study of the α parameterization. All experiments were run with input confidence $\delta = 0.1$. All realizations of the arms were Gaussian random variables with mean μ_i and variance $1/4$[5].

[4]The sufficient statistic for lil'UCB to decide which arm to sample depends only on $\widehat{\mu}_{i,T_i(t)}$ and $T_i(t)$ which only changes for an arm if that particular arm is pulled. Thus, it suffices to maintain an ordered list of the upper confidence bounds in which deleting, updating, and reinserting the arm requires just $O(\log(n))$ computation. Contrast this with a UCB procedure in which the upper confidence bounds depend explicitly on t so that the sufficient statistics for pulling the next arm changes for all arms after each pull, requiring $\Omega(n)$ computation per time step.

[5]The variance was chosen such that the analyses of algorithms that assumed realizations were in $[0, 1]$ and used Hoeffding's inequality were still valid using sub-Gaussian tail bounds with scale parameter $1/2$.



1-sparse, $\mathbf{H}_1 = 4n$ $\alpha = 0.3$, $\mathbf{H}_1 \approx \frac{3}{2}n$ $\alpha = 0.6$, $\mathbf{H}_1 \approx 6n^{1.2}$

Figure 4.3: Stopping times of the algorithms for three scenarios for a variety of problem sizes. The problem scenarios from left to right are the 1-sparse problem ($\mu_1 = 0.5$, $\mu_i = 0 \ \forall i > 1$), $\alpha = 0.3$ ($\mu_i = 1 - (i/n)^\alpha$, $i = 0, 1, \ldots, n$), and $\alpha = 0.6$.

Each algorithm terminates at some finite time with high probability so we first consider the relative stopping times of each of the algorithms in Figure 4.3. Each algorithm was run on each problem scenario and problem size, repeated 50 times. The first observation is that *Exponential-Gap Elimination (+LS)* appears to barely perform better than nonadaptive sampling with the LIL stopping criterion. This confirms our suspicion that the constants in *median elimination* are just too large to make this algorithm practically relevant. While the LIL stopping criterion seems to have measurably improved the *lil'UCB* algorithm, it had no impact on the *lil'UCB Heuristic* variant (not plotted). While *lil'UCB Heuristic* has no theoretical guarantees of outputting the best arm, we remark that over the course of all of our tens of thousands of experiments, the algorithm never failed to terminate with the best arm. The *LUCB* algorithm, despite having worse theoretical guarantees than the *lil'UCB* algorithm, performs surprisingly well. We conjecture that this is because UCB style algorithms tend to lean towards exploiting the top arm versus focusing on increasing the gap between the top two arms, which is the goal of *LUCB*.

In reality, one cannot always wait for an algorithm to run until it terminates on its own so we now explore how the algorithms perform if the algorithm must output an arm at every time step before termination (this is similar to the setting studied in [66]). For each algorithm, at each time we output the arm with the highest empirical mean. Clearly, the probability that a sub-optimal arm is output by any algorithm should very close to 1 in the beginning but then eventually decrease to at least the desired input confidence, and likely, to zero. Figure 4.4 shows the "anytime" performance of the algorithms for the three scenarios and unlike the empirical stopping times of the algorithms, we now observe large differences between the algorithms. Each experiment was repeated 5000 times. Again we see essentially no difference between nonadaptive sampling and the exponential-gap procedure. While in the stopping time plots of Figure 4.3 the *successive elimination* appears competitive with the UCB algorithms, we observe in Figure 4.4 that the UCB algorithms are collecting sufficient information to output the best arm at least twice as fast as *successive elimination*. This tells us that the stopping conditions for the UCB algorithms are still too conservative in practice which motivates the use of the *lil'UCB Heuristic* algorithm which appears to perform very strongly across all metrics. The *LUCB* algorithm again performs strongly here suggesting that LUCB-style algorithms are very well-suited for exploration tasks.

4.5 Discussion

This paper proposed a new procedure for identifying the best arm in a multi-armed bandit problem in the fixed confidence setting, a problem of pure exploration. However, there are some scenarios where one wishes to balance exploration with exploitation and

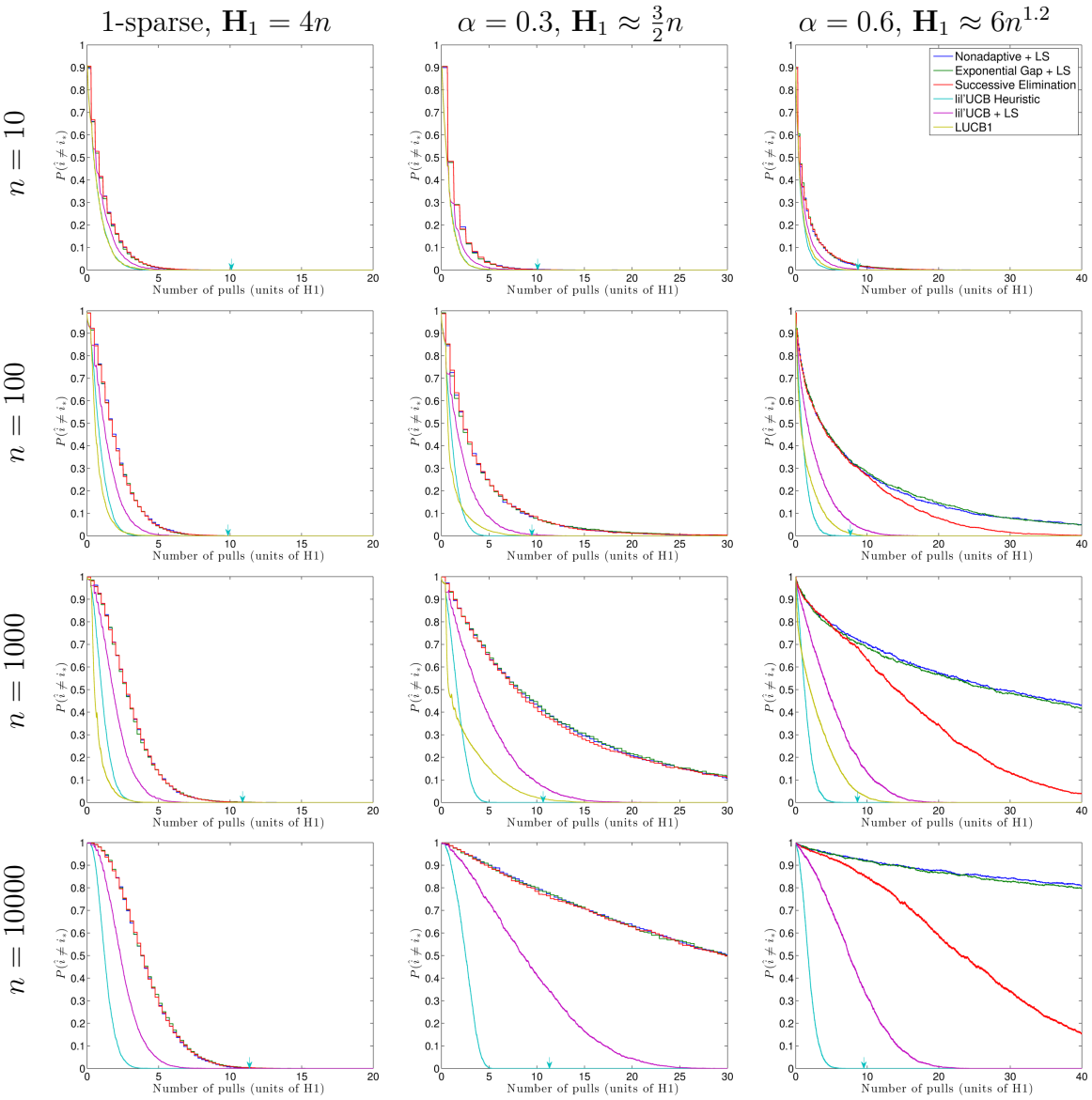


Figure 4.4: At every time, each algorithm outputs an arm $\hat{i}$ that has the highest empirical mean. The $\mathbb{P}(\hat{i} \neq i_*)$ is plotted with respect to the total number of pulls by the algorithm. The problem sizes (number of arms) increase from top to bottom. The problem scenarios from left to right are the 1-sparse problem ($\mu_1 = 0.5$, $\mu_i = 0\ \forall i > 1$) , $\alpha = 0.3$ ($\mu_i = 1 - (i/n)^\alpha$, $i = 0, 1, \ldots, n$), and $\alpha = 0.6$. The arrows indicate the stopping times (if not shown, those algorithms did not terminate within the time window shown). Note that LUCB1 is not plotted for $n = 10000$ due to computational constraints (see text for explanation). Also note that in some plots it is difficult to distinguish between the nonadaptive sampling procedure, the exponential-gap algorithm, and successive elimination due to the curves being on top of each other.

the metric of interest is the *cumulative regret*. We remark that the techniques developed here can be easily extended to show that the lil'UCB algorithm obtains bounded regret with high probability, improving upon the result of [70].

In this work we proved upper and lower bounds over the class of distributions with bounded means and sub-Guassian realizations and presented our results just in terms of the difference between the means of the arms. In contrast to just considering the means of the distributions, [69] studied the Chernoff information between distributions, a quantity related to the KL divergence, that is sharper and can result in improved rates in identifying the best arm in theory and practice (for instance if the realizations from the arms have very different variances). Pursuing methods that exploit distributional characteristics beyond the mean is a good direction for future work.

Finally, an obvious extension of this work is to consider finding the top-m arms instead of just the best arm. This idea has been explored in both the fixed confidence setting [69] and the fixed budget setting [71] but we believe both of these sample complexity results to be suboptimal. It may be possible to adapt the approach developed in this paper to find the top-m arms and obtain gains in theory and practice.

4.6 Bibliographical Remarks

The content of this chapter was based on the author's following publications:

- Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014,

- Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–6. IEEE, 2014,

- Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sebastien Bubeck. On finding the largest mean among many. *Signals, Systems and Computers (ASILOMAR)*, 2013.

Remarkably, within weeks of the first publication of these results, two other publications appeared that also independently derived a form of the finite-time law of the iterated logarithm resembling Lemma 4.3 [67, 68]. The two results focus on tightening the bound for large times at the sacrifice of smaller times. In addition, [68] presents a nearly matching lower bound to the upper bound that may be useful for future lower bounds in the multi-armed bandits literature. On a related note, the proof of the lower bound on the sample complexity of the best-arm identification of [59] was significantly simplified and generalized by [67].

Chapter 5

Non-stochastic Best-arm Identification

In Chapter 4 we studied the stochastic best-arm identification problem where the rewards from each arm were independent random variables with some fixed, unknown mean μ_i and the objective was to discover $\arg\max_i \mu_i$. The fact that the rewards for each arm were independent allowed us to take advantage of concentration inequalities, which informed us of how far the empirical mean of a random variable can deviate from its true mean. While this stochastic setting encompasses many interesting and fundamental problems, there are many natural problems encountered in practice that do not exhibit such structure.

For motivation, consider minimizing a non-convex function with gradient descent. After many iterations, the solver will converge to a local-minima, but because this local-minima may not be the global-minima, a common strategy is to perform gradient descent multiple times, each time starting at a different, random location. If we start with n different random starting positions, we can think of a "pull" of an arm as taking a gradient step (or some fixed number of steps) and computing the function value at the new iterate. As we pull the different arms, they will all start to converge to fixed values, and our objective is to identify the arm that will eventually converge to the lowest function

value. There are many similarities to the stochastic best-arm identification problem, but also many differences. For instance, we know that the function evaluations, like the empirical means in the stochastic case, eventually converge, but unlike the stochastic case we have no confidence bounds to tell us at what *rate* the sequences converge unless something is known about function to be optimized, such as the norm of its gradients being bounded. Without any information about the rate at which the sequences converge, we can also never verify that we have correctly identified the correct arm. And finally, in the stochastic case we assumed that we could observe the raw rewards instantly whereas in the non-stochastic case there may be some cost to evaluating the value of an arm, like computing the value of the objective function. In this chapter, motivated by a hyperparameter tuning problem for machine learning, we address these challenges and propose a new framework for solving the non-stochastic best-arm identification problem.

5.1 Introduction

As supervised learning methods are becoming more widely adopted, hyperparameter optimization has become increasingly important to simplify and speed up the development of data processing pipelines while simultaneously yielding more accurate models. In hyperparameter optimization for supervised learning, we are given labeled training data, a set of hyperparameters associated with our supervised learning methods of interest, and a search space over these hyperparameters. We aim to find a particular configuration of hyperparameters that optimizes some evaluation criterion, e.g., loss on a validation dataset.

Since many machine learning algorithms are iterative in nature, particularly when

working at scale, we can evaluate the quality of intermediate results, i.e., partially trained learning models, resulting in a sequence of losses that eventually converges to the final loss value at convergence. For example, Figure 5.1 shows the sequence of validation losses for various hyperparameter settings for kernel SVM models trained via stochastic gradient descent. The figure shows high variability in model quality across hyperparameter settings. It thus seems natural to ask the question: *Can we terminate these poor-performing hyperparameter settings early in a principled online fashion to speed up hyperparameter optimization?*

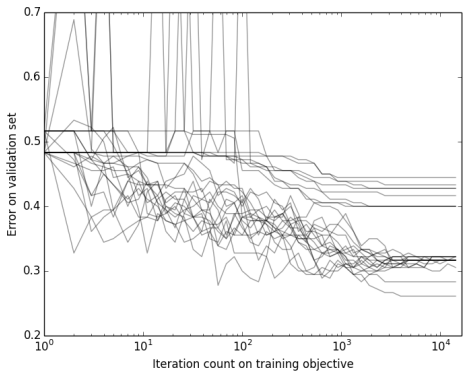


Figure 5.1: Validation error for different hyperparameter choices for a classification task trained using stochastic gradient descent.

Although several hyperparameter optimization methods have been proposed recently, e.g., [72,73,74,75,76], the vast majority of them consider the training of machine learning models to be black-box procedures, and only evaluate models after they are fully trained to convergence. A few recent works have made attempts to exploit intermediate results. However, these works either require explicit forms for the convergence rate behavior of the iterates which is difficult to accurately characterize for all but the simplest cases [77,78], or focus on heuristics lacking theoretical underpinnings [79]. We build upon these previous

works, and in particular study the multi-armed bandit formulation proposed in [77] and [79], where each arm corresponds to a fixed hyperparameter setting, pulling an arm corresponds to a fixed number of training iterations, and the loss corresponds to an intermediate loss on some hold-out set.

We aim to provide a robust, general-purpose, and widely applicable bandit-based solution to hyperparameter optimization. Remarkably, however, the existing multi-armed bandits literature fails to address this natural problem setting: a *non-stochastic best-arm identification* problem. While multi-armed bandits is a thriving area of research, we believe that the existing work fails to adequately address the two main challenges in this setting:

1. We know each arm's sequence of losses eventually converges, but we have no information about the rate of convergence, and the sequence of losses, like those in Figure 5.1, may exhibit a high degree of non-monotonicity and non-smoothness.

2. The cost of obtaining the loss of an arm can be disproportionately more costly than pulling it. For example, in the case of hyperparameter optimization, computing the validation loss is often drastically more expensive than performing a single training iteration.

We thus study this novel bandit setting, which encompasses the hyperparameter optimization problem, and analyze an algorithm we identify as being particularly well-suited for this setting. Moreover, we confirm our theory with empirical studies that demonstrate an order of magnitude speedups relative to standard baselines on a number of real-world supervised learning problems and datasets.

We note that this bandit setting is quite generally applicable. While the problem of hyperparameter optimization inspired this work, the setting itself encompasses the

stochastic best-arm identification problem [80], less-well-behaved stochastic sources like max-bandits [81], exhaustive subset selection for feature extraction, and many optimization problems that "feel" like stochastic best-arm problems but lack the i.i.d. assumptions necessary in that setting.

The remainder of the paper is organized as follows: In Section 5.2 we present the setting of interest, provide a survey of related work, and explain why most existing algorithms and analyses are not well-suited or applicable for our setting. We then study our proposed algorithm in Section 5.3 in our setting of interest, and analyze its performance relative to a natural baseline. We then relate these results to the problem of hyperparameter optimization in Section 5.4, and present our experimental results in Section 5.5.

5.2 Non-stochastic best arm identification

Objective functions for multi-armed bandits problems tend to take on one of two flavors: 1) best arm identification (or pure exploration) in which one is interested in identifying the arm with the highest average payoff, and 2) exploration-versus-exploitation in which we are trying to maximize the cumulative payoff over time [82]. While the latter has been analyzed in both the stochastic and non-stochastic settings, we are unaware of any work that addresses the best arm objective in the non-stochastic setting, which is our setting of interest. Moreover, while related, a strategy that is well-suited for maximizing cumulative payoff is not necessarily well-suited for the best-arm identification task, even in the stochastic setting [80].

The algorithm of Figure 5.2 presents a general form of the best arm problem for

Best Arm Problem for Multi-armed Bandits

input: n arms where $\ell_{i,k}$ denotes the loss observed on the kth pull of the ith arm

initialize: $T_i = 1$ for all $i \in [n]$

for $t = 1, 2, 3, \ldots$

Algorithm chooses an index $I_t \in [n]$

Loss $\ell_{I_t, T_{I_t}}$ is revealed, $T_{I_t} = T_{I_t} + 1$

Algorithm outputs a recommendation $J_t \in [n]$

Receive external stopping signal, otherwise continue

Figure 5.2: A generalization of the best arm problem for multi-armed bandits [80] that applies to both the stochastic and non-stochastic settings.

multi-armed bandits. Intuitively, at each time t the goal is to choose J_t such that the arm associated with J_t has the lowest loss in some sense. Note that while the algorithm gets to observe the value for an arbitrary arm I_t, the algorithm is only evaluated on its recommendation J_t, that it also chooses arbitrarily. This is in contrast to the exploration-versus-exploitation game where the arm that is played is also the arm that the algorithm is evaluated on, namely, I_t.

The best-arm identification problems defined below require that the losses be generated by an oblivious adversary, which essentially means that the loss sequences are independent of the algorithm's actions. Contrast this with an adaptive adversary that can adapt future losses based on all the arms that the algorithm has played up to the current time. If the losses are chosen by an oblivious adversary then without loss of generality we may assume that all the losses were generated before the start of the game. See [82] for more info. We now compare the stochastic and the proposed non-stochastic best-arm identification problems.

Stochastic : For all $i \in [n]$, $k \geq 1$, let $\ell_{i,k}$ be an i.i.d. sample from a probability

distribution supported on $[0, 1]$. For each i, $\mathbb{E}[\ell_{i,k}]$ exists and is equal to some constant μ_i for all $k \geq 1$. The goal is to identify $\arg\min_i \mu_i$ while minimizing $\sum_{i=1}^n T_i$.

Non-stochastic (proposed in this work) : For all $i \in [n]$, $k \geq 1$, let $\ell_{i,k} \in \mathbb{R}$ be generated by an oblivious adversary and assume $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$ exists. The goal is to identify $\arg\min_i \nu_i$ while minimizing $\sum_{i=1}^n T_i$.

These two settings are related in that we can always turn the stochastic setting into the non-stochastic setting by defining $\ell_{i,T_i} = \frac{1}{T_i} \sum_{k=1}^{T_i} \ell'_{i,T_i}$ where ℓ'_{i,T_i} are the losses from the stochastic problem; by the law of large numbers $\lim_{\tau \to \infty} \ell_{i,\tau} = \mathbb{E}[\ell'_{i,1}]$. In fact, we could do something similar with other less-well-behaved statistics like the minimum (or maximum) of the stochastic returns of an arm. As described in [81], we can define $\ell_{i,T_i} = \min\{\ell'_{i,1}, \ell'_{i,2}, \ldots, \ell'_{i,T_i}\}$, which has a limit since $\ell_{i,t}$ is a bounded, monotonically decreasing sequence.

However, the generality of the non-stochastic setting introduces novel challenges. In the stochastic setting, if we set $\widehat{\mu}_{i,T_i} = \frac{1}{T_i} \sum_{k=1}^{T_i} \ell_{i,k}$ then with probability at least $1 - \delta$ we have $|\widehat{\mu}_{i,T_i} - \mu_i| \leq \sqrt{\frac{\log(4nT_i^2)}{2T_i}}$ for all $i \in [n]$ and $T_i > 0$ by applying Hoeffding's inequality and a union bound. In contrast, the non-stochastic setting's assumption that $\lim_{\tau \to \infty} \ell_{i,\tau}$ exists implies that there exists a non-increasing function γ_i such that $|\ell_{i,t} - \lim_{\tau \to \infty} \ell_{i,\tau}| \leq \gamma_i(t)$ and that $\lim_{t \to \infty} \gamma_i(t) = 0$. However, the existence of this limit tells us nothing about how *quickly* $\gamma_i(t)$ approaches 0. The lack of an explicit convergence rate as a function of t presents a problem as even the tightest $\gamma_i(t)$ could decay arbitrarily slowly and we would never know it.

This observation has two consequences. First, we can never *reject* the possibility that

an arm is the "best" arm. Second, we can never *verify* that an arm is the "best" arm or even attain a value within ϵ of the best arm. Despite these challenges, in Section 5.3 we identify an effective algorithm under natural measures of performance, using ideas inspired by the fixed budget setting of the stochastic best arm problem [55, 61, 65].

5.2.1 Related work

Despite dating to back to the late 1950's, the best-arm identification problem for the stochastic setting has experienced a surge of activity in the last decade. The work has two major branches: the fixed budget setting and the fixed confidence setting. In the fixed budget setting, the algorithm is given a set of arms and a budget B and is tasked with maximizing the probability of identifying the best arm by pulling arms without exceeding the total budget. While these algorithms were developed for and analyzed in the stochastic setting, they exhibit attributes that are very amenable to the non-stochastic setting. In fact, the algorithm we propose to use in this paper is exactly the Successive Halving algorithm of [61], though the non-stochastic setting requires its own novel analysis that we present in Section 5.3. Successive Rejects [65] is another fixed budget algorithm that we compare to in our experiments.

The best-arm identification problem in the fixed confidence setting takes an input $\delta \in (0, 1)$ and guarantees to output the best arm with probability at least $1 - \delta$ while attempting to minimize the number of total arm pulls. These algorithms rely on probability theory to determine how many times each arm must be pulled in order to decide if the arm is suboptimal and should no longer be pulled, either by explicitly discarding it, e.g., Successive Elimination [83] and Exponential Gap Elimination [61],

Exploration algorithm	# observed losses
Uniform (baseline) (B)	n
Successive Halving* (B)	$2n+1$
Successive Rejects (B)	$(n+1)n/2$
Successive Elimination (C)	$n\log_2(2B)$
LUCB (C), lil'UCB (C), EXP3 (R)	B

Table 5.1: The number of times an algorithm observes a loss in terms of budget B and number of arms n, where B is known to the algorithm. (B), (C), or (R) indicate whether the algorithm is of the fixed budget, fixed confidence, or cumulative regret variety, respectfully. (*) indicates the algorithm we propose for use in the non-stochastic best arm setting.

or implicitly by other methods, e.g., LUCB [60] and Lil'UCB [84]. For an in-depth review of the stochastic best-arm identification problem, we refer the reader to Chapter 4 Algorithms from the fixed confidence setting are ill-suited for the non-stochastic best-arm identification problem because they rely on statistical bounds that are generally not applicable in the non-stochastic case. These algorithms also exhibit some undesirable behavior with respect to how many losses they observe, which we explore next.

In addition to just the total number of arm pulls, this work also considers the required number of *observed* losses. This is a natural cost to consider when ℓ_{i,T_i} for any i is the result of doing some computation like evaluating a partially trained classifier on a hold-out validation set or releasing a product to the market to probe for demand. In some cases the cost, be it time, effort, or dollars, of an evaluation of the loss of an arm after some number of pulls can dwarf the cost of pulling the arm. Assuming a known time horizon (or budget), Table 5.1 describes the total number of times various algorithms observe a loss as a function of the budget B and the number of arms n. We include in our comparison the EXP3 algorithm [85], a popular approach for minimizing cumulative regret in the non-stochastic setting. In practice $B \gg n$, and thus Successive Halving is

a particular attractive option, as along with the baseline, it is the only algorithm that observes losses proportional to the number of arms and independent of the budget. As we will see in Section 5.5, the performance of these algorithms is quite dependent on the number of observed losses.

5.3 Proposed algorithm and analysis

The proposed Successive Halving algorithm of Figure 5.3 was originally proposed for the stochastic best arm identification problem in the fixed budget setting by [61]. However, our novel analysis in this work shows that it is also effective in the non-stochastic setting. The idea behind the algorithm is simple: given an input budget, uniformly allocate the budget to a set of arms for a predefined amount of iterations, evaluate their performance, throw out the worst half, and repeat until just one arm remains.

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith arm

Initialize: $S_0 = [n]$.

For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$

Pull each arm in S_k for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ additional times and set $R_k = \sum_{j=0}^{k} r_j$.

Let σ_k be a bijection on S_k such that $\ell_{\sigma_k(1),R_k} \leq \ell_{\sigma_k(2),R_k} \leq \cdots \leq \ell_{\sigma_k(|S_k|),R_k}$

$S_{k+1} = \left\{ i \in S_k : \ell_{\sigma_k(i),R_k} \leq \ell_{\sigma_k(\lfloor |S_k|/2 \rfloor),R_k} \right\}$.

output : Singleton element of $S_{\lceil \log_2(n) \rceil}$

Figure 5.3: Successive Halving was originally proposed for the stochastic best arm identification problem in [61] but is also applicable to the non-stochastic setting.

The budget as an input is easily removed by the "doubling trick" that attempts $B \leftarrow n$, then $B \leftarrow 2B$, and so on. This method can reuse existing progress from iteration to iteration and effectively makes the algorithm parameter free. But its most notable quality is that if a budget of B' is necessary to succeed in finding the best arm, by

performing the doubling trick one will have only had to use a budget of $2B'$ in the worst case without ever having to know B' in the first place. Thus, for the remainder of this section we consider a fixed budget.

5.3.1 Analysis of Successive Halving

We first show that the algorithm never takes a total number of samples that exceeds the budget B:

$$\sum_{k=0}^{\lceil \log_2(n) \rceil - 1} |S_k| \left\lfloor \frac{B}{|S_k| \lceil \log(n) \rceil} \right\rfloor \leq \sum_{k=0}^{\lceil \log_2(n) \rceil - 1} \frac{B}{\lceil \log(n) \rceil} \leq B \,.$$

Next we consider how the algorithm performs in terms of identifying the best arm. First, for $i = 1, \ldots, n$ define $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$ which exists by assumption. Without loss of generality, assume that

$$\nu_1 < \nu_2 \leq \cdots \leq \nu_n \,.$$

We next introduce functions that bound the approximation error of $\ell_{i,t}$ with respect to ν_i as a function of t. For each $i = 1, 2, \ldots, n$ let $\gamma_i(t)$ be the point-wise smallest, non-increasing function of t such that

$$|\ell_{i,t} - \nu_i| \leq \gamma_i(t) \quad \forall t.$$

In addition, define $\gamma_i^{-1}(\alpha) = \min\{t \in \mathbb{N} : \gamma_i(t) \leq \alpha\}$ for all $i \in [n]$. With this definition, if $t_i > \gamma_i^{-1}(\frac{\nu_i - \nu_1}{2})$ and $t_1 > \gamma_1^{-1}(\frac{\nu_i - \nu_1}{2})$ then

$$\begin{aligned}\ell_{i,t_i} - \ell_{1,t_1} &= (\ell_{i,t_i} - \nu_i) + (\nu_1 - \ell_{1,t_1}) + 2\left(\tfrac{\nu_i - \nu_1}{2}\right) \\ &\geq -\gamma_i(t_i) - \gamma_1(t_1) + 2\left(\tfrac{\nu_i - \nu_1}{2}\right) > 0.\end{aligned}$$

Indeed, if $\min\{t_i, t_1\} > \max\{\gamma_i^{-1}(\frac{\nu_i - \nu_1}{2}), \gamma_1^{-1}(\frac{\nu_i - \nu_1}{2})\}$ then we are guaranteed to have that $\ell_{i,t_i} > \ell_{1,t_1}$. That is, comparing the intermediate values at t_i and t_1 suffices to determine the ordering of the final values ν_i and ν_1. Intuitively, this condition holds because the envelopes at the given times, namely $\gamma_i(t_i)$ and $\gamma_1(t_1)$, are small relative to the gap between ν_i and ν_1. This line of reasoning is at the heart of the proof of our main result, and the theorem is stated in terms of these quantities.

Theorem 5.1. *Let* $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$, $\bar{\gamma}(t) = \max_{i=1,\ldots,n} \gamma_i(t)$ *and*

$$\begin{aligned}z &= 2\lceil \log_2(n) \rceil \max_{i=2,\ldots,n} i \left(1 + \bar{\gamma}^{-1}\left(\tfrac{\nu_i - \nu_1}{2}\right)\right) \\ &\leq 2\lceil \log_2(n) \rceil \left(n + \bar{\gamma}^{-1}\left(\tfrac{\nu_2 - \nu_1}{2}\right) + \sum_{i=2,\ldots,n} \bar{\gamma}^{-1}\left(\tfrac{\nu_i - \nu_1}{2}\right)\right) < 8\lceil \log_2(n) \rceil \sum_{i=2,\ldots,n} \bar{\gamma}^{-1}\left(\tfrac{\nu_i - \nu_1}{2}\right).\end{aligned}$$

If the budget $B > z$ *then the best arm is returned from the algorithm.*

Proof. For notational ease, define $[\cdot] = \{\{\cdot\}_{t=1}\}_{i=1}^n$ so that $[\ell_{i,t}] = \{\{\ell_{i,t}\}_{t=1}^\infty\}_{i=1}^n$. Without loss of generality, we may assume that the n infinitely long loss sequences $[\ell_{i,t}]$ with limits $\{\nu_i\}_{i=1}^n$ were fixed prior to the start of the game so that the $\gamma_i(t)$ envelopes are also defined for all time and are fixed. Let Ω be the set that contains all possible sets of n

infinitely long sequences of real numbers with limits $\{\nu_i\}_{i=1}^n$ and envelopes $[\bar{\gamma}(t)]$, that is,

$$\Omega = \left\{ [\ell'_{i,t}] : [\ |\ell'_{i,t} - \nu_i| \leq \bar{\gamma}(t)\ ] \wedge \lim_{\tau \to \infty} \ell'_{i,\tau} = \nu_i \quad \forall i \right\}$$

where we recall that $\wedge$ is read as "and" and $\vee$ is read as "or." Clearly, $[\ell_{i,t}]$ is a single element of Ω.

We present a proof by contradiction. We begin by considering the singleton set containing $[\ell_{i,t}]$ under the assumption that the Successive Halving algorithm fails to identify the best arm, i.e., $S_{\lceil \log_2(n) \rceil} \neq 1$. We then consider a sequence of subsets of Ω, with each one contained in the next. The proof is completed by showing that the final subset in our sequence (and thus our original singleton set of interest) is empty when $B > z$, which contradicts our assumption and proves the statement of our theorem.

To reduce clutter in the following arguments, it is understood that S'_k for all k in the following sets is a function of $[\ell'_{i,t}]$ in the sense that it is the state of S_k in the algorithm when it is run with losses $[\ell'_{i,t}]$. We now present our argument in detail, starting with the

singleton set of interest, and using the definition of S_k in Figure 5.3.

$$
\begin{aligned}
&\left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge S'_{\lceil \log_2(n) \rceil} \neq 1 \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge \bigvee_{k=1}^{\lceil \log_2(n) \rceil} \{1 \notin S'_k,\ 1 \in S'_{k-1}\} \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i \in S'_k} \mathbf{1}\{\ell'_{i,R_k} < \ell'_{1,R_k}\} > \lfloor |S'_k|/2 \rfloor \right\} \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i \in S'_k} \mathbf{1}\{\nu_i - \nu_1 < \ell'_{1,R_k} - \nu_1 - \ell'_{i,R_k} + \nu_i\} > \lfloor |S'_k|/2 \rfloor \right\} \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i \in S'_k} \mathbf{1}\{\nu_i - \nu_1 < |\ell'_{1,R_k} - \nu_1| + |\ell'_{i,R_k} - \nu_i|\} > \lfloor |S'_k|/2 \rfloor \right\} \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i \in S'_k} \mathbf{1}\{2\bar{\gamma}(R_k) > \nu_i - \nu_1\} > \lfloor |S'_k|/2 \rfloor \right\} \right\}, \qquad (5.1)
\end{aligned}
$$

where the last set relaxes the original equality condition to just considering the maximum envelope $\bar{\gamma}$ that is encoded in Ω. The summation in Eq. 5.1 only involves the ν_i, and this

summand is maximized if each S'_k contains the first $|S'_k|$ arms. Hence we have,

$$\begin{aligned}
(5.1) &\subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \sum_{i=1}^{|S'_k|} \mathbf{1}\{2\bar{\gamma}(R_k) > \nu_i - \nu_1\} > \lfloor |S'_k|/2 \rfloor \right\} \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ 2\bar{\gamma}(R_k) > \nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1 \right\} \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ R_k < \bar{\gamma}^{-1} \left(\frac{\nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1}{2} \right) \right\} \right\}, \qquad (5.2)
\end{aligned}$$

where we use the definition of γ^{-1} in Eq. 5.2. Next, we recall that $R_k = \sum_{j=0}^{k} \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor \geq \frac{B/2}{(\lfloor |S_k|/2 \rfloor + 1) \lceil \log_2(n) \rceil} - 1$ since $|S_k| \leq 2(\lfloor |S_k|/2 \rfloor + 1)$. We note that we are underestimating by almost a factor of 2 to account for integer effects in favor of a simpler form. By plugging in this value for R_k and rearranging we have that

$$\begin{aligned}
(5.2) &\subseteq \left\{ [\ell'_{i,t}] \in \Omega : \bigvee_{k=0}^{\lceil \log_2(n) \rceil - 1} \left\{ \frac{B/2}{\lceil \log_2(n) \rceil} < (\lfloor |S'_k|/2 \rfloor + 1)(1 + \bar{\gamma}^{-1} \left(\frac{\nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1}{2} \right)) \right\} \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : \frac{B/2}{\lceil \log_2(n) \rceil} < \max_{k=0,\ldots,\lceil \log_2(n) \rceil - 1} (\lfloor |S'_k|/2 \rfloor + 1)(1 + \bar{\gamma}^{-1} \left(\frac{\nu_{\lfloor |S'_k|/2 \rfloor + 1} - \nu_1}{2} \right)) \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : B < 2\lceil \log_2(n) \rceil \max_{i=2,\ldots,n} i \, (\bar{\gamma}^{-1} \left(\frac{\nu_i - \nu_1}{2} \right) + 1) \right\} = \emptyset
\end{aligned}$$

where the last equality holds if $B > z$.

The second, looser, but perhaps more interpretable form of z is thanks to [65] who

showed that for a decreasing sequence of numbers $x_1, \ldots, x_n$

$$\max_{i=1,\ldots,n} i\, x_i \leq \sum_{i=1,\ldots,n} x_i \leq \log_2(2n) \max_{i=1,\ldots,n} i\, x_i$$

where both inequalities are achievable with particular settings of the x_i variables. □

The representation of z on the right-hand-side of the inequality is very intuitive: if $\bar{\gamma}(t) = \gamma_i(t) \;\; \forall i$ and an oracle gave us an explicit form for $\bar{\gamma}(t)$, then to merely verify that the ith arm's final value is higher than the best arm's, one must pull each of the two arms at least a number of times equal to the ith term in the sum (this becomes clear by inspecting the proof of Theorem 5.3). Repeating this argument for all $i = 2, \ldots, n$ explains the sum over all $n - 1$ arms. While clearly not a proof, this argument along with known lower bounds for the stochastic setting [65,67], a subset of the non-stochastic setting, suggest that the above result may be nearly tight in a minimax sense up to log factors.

Example 1. *Consider a feature-selection problem where you are given a dataset* $\{(x_i, y_i)\}_{i=1}^n$ *where each* $x_i \in \mathbb{R}^D$ *and you are tasked with identifying the best subset of features of size* d *that linearly predicts* y_i *in terms of the least-squares metric. In our framework, each* d*-subset is an arm and there are* $n = \binom{D}{d}$ *arms. Least squares is a convex quadratic optimization problem that can be efficiently solved with stochastic gradient descent. Using known bounds for the rates of convergence [86] one can show that* $\gamma_a(t) \leq \frac{\sigma_a \log(nt/\delta)}{t}$ *for all* $a = 1, \ldots, n$ *arms and all* $t \geq 1$ *with probability at least* $1 - \delta$ *where* σ_a *is a constant that depends on the condition number of the quadratic defined by the* d*-subset. Then in Theorem 5.1,* $\bar{\gamma}(t) = \frac{\sigma_{\max} \log(nt/\delta)}{t}$ *with* $\sigma_{\max} = \max_{a=1,\ldots,n} \sigma_a$ *so after inverting* $\bar{\gamma}$ *we*

find that $z = 2\lceil \log_2(n) \rceil \max_{a=2,\dots,n} a \frac{4\sigma_{\max} \log\left(\frac{2n\sigma_{\max}}{\delta(\nu_a - \nu_1)}\right)}{\nu_a - \nu_1}$ *is a sufficient budget to identify the best arm. Later we put this result in context by comparing to a baseline strategy.*

In the above example we computed upper bounds on the γ_i functions in terms of problem dependent parameters to provide us with a sample complexity by plugging these values into our theorem. However, we stress that constructing tight bounds for the γ_i functions is very difficult outside of very simple problems like the one described above, and even then we have unspecified constants. Fortunately, because our algorithm is agnostic to these γ_i functions, it is also in some sense *adaptive* to them: the faster the arms' losses converge, the faster the best arm is discovered, without ever changing the algorithm. This behavior is in stark contrast to the hyperparameter tuning work of [78] and [77], in which the algorithms explicitly take upper bounds on these γ_i functions as input, meaning the performance of the algorithm is only as good as the tightness of these difficult to calculate bounds.

5.3.2 Comparison to a uniform allocation strategy

We can also derive a result for the naive uniform budget allocation strategy. For simplicity, let B be a multiple of n so that at the end of the budget we have $T_i = B/n$ for all $i \in [n]$ and the output arm is equal to $\widehat{i} = \arg\min_i \ell_{i,B/n}$.

Theorem 5.2. *(Uniform strategy – sufficiency) Let* $\nu_i = \lim_{\tau \to \infty} \ell_{i,\tau}$, $\bar{\gamma}(t) = \max_{i=1,\dots,n} \gamma_i(t)$ *and*

$$z = \max_{i=2,\dots,n} n\bar{\gamma}^{-1}\left(\tfrac{\nu_i - \nu_1}{2}\right).$$

If $B > z$ then the uniform strategy returns the best arm.

Proof. Recall the notation from the proof of Theorem 5.1 and let $\widehat{i}([\ell'_{i,t}])$ be the output of the uniform allocation strategy with input losses $[\ell'_{i,t}]$.

$$\begin{aligned}
\left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge \widehat{i}([\ell'_{i,t}]) \neq 1 \right\} &= \left\{ [\ell'_{i,t}] \in \Omega : [\ell'_{i,t} = \ell_{i,t}] \wedge \ell'_{1,B/n} \geq \min_{i=2,\ldots,n} \ell'_{i,B/n} \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : 2\bar{\gamma}(B/n) \geq \min_{i=2,\ldots,n} \nu_i - \nu_1 \right\} \\
&= \left\{ [\ell'_{i,t}] \in \Omega : 2\bar{\gamma}(B/n) \geq \nu_2 - \nu_1 \right\} \\
&\subseteq \left\{ [\ell'_{i,t}] \in \Omega : B \leq n\bar{\gamma}^{-1}\left(\tfrac{\nu_2-\nu_1}{2}\right) \right\} = \emptyset
\end{aligned}$$

where the last equality follows from the fact that $B > z$ which implies $\widehat{i}([\ell_{i,t}]) = 1$. □

Theorem 5.2 is just a sufficiency statement so it is unclear how the performance of the method actually compares to the Successive Halving result of Theorem 5.1. The next theorem says that the above result is tight in a worst-case sense, exposing the real gap between the algorithm of Figure 5.3 and the naive uniform allocation strategy.

Theorem 5.3. *(Uniform strategy – necessity) For any given budget B and final values $\nu_1 < \nu_2 \leq \cdots \leq \nu_n$ there exists a sequence of losses $\{\ell_{i,t}\}_{t=1}^{\infty}$, $i = 1, 2, \ldots, n$ such that if*

$$B < \max_{i=2,\ldots,n} n\bar{\gamma}^{-1}\left(\tfrac{\nu_i-\nu_1}{2}\right)$$

then the uniform budget allocation strategy will not return the best arm.

Proof. Let $\beta(t)$ be an arbitrary, monotonically decreasing function of t with $\lim_{t\to\infty}\beta(t) = 0$. Define $\ell_{1,t} = \nu_1 + \beta(t)$ and $\ell_{i,t} = \nu_i - \beta(t)$ for all i. Note that for all i, $\gamma_i(t) = \bar{\gamma}(t) = \beta(t)$ so that

$$\begin{aligned}
\widehat{i} = 1 &\iff \ell_{1,B/n} < \min_{i=2,\dots,n} \ell_{i,B/n} \\
&\iff \nu_1 + \bar{\gamma}(B/n) < \min_{i=2,\dots,n} \nu_i - \bar{\gamma}(B/n) \\
&\iff \nu_1 + \bar{\gamma}(B/n) < \nu_2 - \bar{\gamma}(B/n) \\
&\iff \bar{\gamma}(B/n) < \frac{\nu_2 - \nu_1}{2} \\
&\iff B \geq n\bar{\gamma}^{-1}\left(\tfrac{\nu_2-\nu_1}{2}\right).
\end{aligned}$$

□

If we consider the second, looser representation of z on the right-hand-side of the inequality in Theorem 5.1 and multiply this quantity by $\frac{n-1}{n-1}$ we see that the sufficient number of pulls for the Successive Halving algorithm essentially behaves like $(n-1)\log_2(n)$ times the *average* $\frac{1}{n-1}\sum_{i=2,\dots,n}\bar{\gamma}^{-1}\left(\frac{\nu_i-\nu_1}{2}\right)$ whereas the necessary result of the uniform allocation strategy of Theorem 5.3 behaves like n times the *maximum* $\max_{i=2,\dots,n}\bar{\gamma}^{-1}\left(\frac{\nu_i-\nu_1}{2}\right)$. The next example shows that the difference between this average and max can be very significant.

Example 2. *Recall Example 1 and now assume that $\sigma_a = \sigma_{\max}$ for all $a = 1,\dots,n$. Then Theorem 5.3 says that the uniform allocation budget must be at least $n\frac{4\sigma_{\max}\log\left(\frac{2n\sigma_{\max}}{\delta(\nu_2-\nu_1)}\right)}{\nu_2-\nu_1}$ to identify the best arm. To see how this result compares with that of Successive Halving, let us parameterize the ν_a limiting values such that $\nu_a = a/n$ for $a = 1,\dots,n$. Then a sufficient budget for the Successive Halving algorithm to identify the best arm is just*

$8n\lceil\log_2(n)\rceil\sigma_{\max}\log\left(\frac{n^2\sigma_{\max}}{\delta}\right)$ *while the uniform allocation strategy would require a budget of at least* $2n^2\sigma_{\max}\log\left(\frac{n^2\sigma_{\max}}{\delta}\right)$. *This is a difference of essentially* $4n\log_2(n)$ *versus* n^2.

5.3.3 A pretty good arm

Up to this point we have been concerned with identifying the *best* arm: $\nu_1 = \arg\min_i \nu_i$ where we recall that $\nu_i = \lim_{\tau\to\infty} \ell_{i,\tau}$. But in practice one may be satisfied with merely an ϵ-good arm i_ϵ in the sense that $\nu_{i_\epsilon} - \nu_1 \leq \epsilon$. However, with our minimal assumptions, such a statement is impossible to make since we have no knowledge of the γ_i functions to determine that an arm's final value is within ϵ of any value, much less the unknown final converged value of the best arm. However, as we show in Theorem 5.4, the Successive Halving algorithm cannot do much worse than the uniform allocation strategy.

Theorem 5.4. *For a budget* B *and set of* n *arms, define* $\widehat{i}_{SH}$ *as the output of the Successive Halving algorithm. Then*

$$\nu_{\widehat{i}_{SH}} - \nu_1 \leq \lceil\log_2(n)\rceil 2\bar{\gamma}\left(\lfloor\tfrac{B}{n\lceil\log_2(n)\rceil}\rfloor\right).$$

Moreover, $\widehat{i}_U$*, the output of the uniform strategy, satisfies*

$$\nu_{\widehat{i}_U} - \nu_1 \leq \ell_{\widehat{i},B/n} - \ell_{1,B/n} + 2\bar{\gamma}(B/n) \leq 2\bar{\gamma}(B/n).$$

Proof. We can guarantee for the Successive Halving algorithm of Figure 5.3 that the

output arm $\widehat{i}$ satisfies

$$\begin{aligned}
\nu_{\widehat{i}} - \nu_1 &= \min_{i \in S_{\lceil \log_2(n) \rceil}} \nu_i - \nu_1 \\
&= \sum_{k=0}^{\lceil \log_2(n) \rceil - 1} \min_{i \in S_{k+1}} \nu_i - \min_{i \in S_k} \nu_i \\
&\leq \sum_{k=0}^{\lceil \log_2(n) \rceil - 1} \min_{i \in S_{k+1}} \ell_{i,R_k} - \min_{i \in S_k} \ell_{i,R_k} + 2\bar{\gamma}(R_k) \\
&= \sum_{k=0}^{\lceil \log_2(n) \rceil - 1} 2\bar{\gamma}(R_k) \leq \lceil \log_2(n) \rceil 2\bar{\gamma} \left(\lfloor \tfrac{B}{n \lceil \log_2(n) \rceil} \rfloor \right)
\end{aligned}$$

simply by inspecting how the algorithm eliminates arms and plugging in a trivial lower bound for R_k for all k in the last step. □

Example 3. *Recall Example 1. Both the Successive Halving algorithm and the uniform allocation strategy satisfy* $\widehat{\nu}_i - \nu_1 \leq \widetilde{O}(n/B)$ *where* $\widehat{i}$ *is the output of either algorithm and* $\widetilde{O}$ *suppresses poly*log *factors.*

We stress that this result is merely a fall-back guarantee, ensuring that we can never do much worse than uniform. However, it does not rule out the possibility of the Successive Halving algorithm far outperforming the uniform allocation strategy in practice. Indeed, we observe order of magnitude speed ups in our experimental results.

5.4 Hyperparameter optimization for supervised learning

In supervised learning we are given a dataset that is composed of pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ for $i = 1, \ldots, n$ sampled i.i.d. from some unknown joint distribution $P_{X,Y}$, and we are tasked

with finding a map (or model) $f : \mathcal{X} \to \mathcal{Y}$ that minimizes $\mathbb{E}_{(X,Y)\sim P_{X,Y}}\left[\texttt{loss}(f(X), Y)\right]$ for some known loss function $\texttt{loss} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Since $P_{X,Y}$ is unknown, we cannot compute $\mathbb{E}_{(X,Y)\sim P_{XY}}[\texttt{loss}(f(X), Y)]$ directly, but given m additional samples drawn i.i.d. from $P_{X,Y}$ we can approximate it with an empirical estimate, that is, $\frac{1}{m}\sum_{i=1}^{m}\texttt{loss}(f(x_i), y_i)$. We do not consider arbitrary mappings $\mathcal{X} \to \mathcal{Y}$ but only those that are the output of running a fixed, possibly randomized, algorithm $\mathcal{A}$ that takes a dataset $\{(x_i, y_i)\}_{i=1}^{n}$ and algorithm-specific parameters $\theta \in \Theta$ as input so that for any θ we have $f_\theta = \mathcal{A}\left(\{(x_i, y_i)\}_{i=1}^{n}, \theta\right)$ where $f_\theta : \mathcal{X} \to \mathcal{Y}$. For a fixed dataset $\{(x_i, y_i)\}_{i=1}^{n}$ the parameters $\theta \in \Theta$ index the different functions f_θ, and will henceforth be referred to as *hyperparameters*. We adopt the train-validate-test framework for choosing hyperparameters [87]:

1. Partition the total dataset into $\texttt{TRAIN}$, $\texttt{VAL}$, and $\texttt{TEST}$ sets with $\texttt{TRAIN}\cup\texttt{VAL}\cup\texttt{TEST} = \{(x_i, y_i)\}_{i=1}^{m}$.
2. Use $\texttt{TRAIN}$ to train a model $f_\theta = \mathcal{A}\left(\{(x_i, y_i)\}_{i\in\texttt{TRAIN}}, \theta\right)$ for each $\theta \in \Theta$,
3. Choose the hyperparameters that minimize the empirical loss on the examples in $\texttt{VAL}$: $\widehat{\theta} = \arg\min_{\theta\in\Theta}\frac{1}{|\texttt{VAL}|}\sum_{i\in\texttt{VAL}}\texttt{loss}(f_\theta(x_i), y_i)$
4. Report the empirical loss of $\widehat{\theta}$ on the test error: $\frac{1}{|\texttt{TEST}|}\sum_{i\in\texttt{TEST}}\texttt{loss}(f_{\widehat{\theta}}(x_i), y_i)$.

Example 4. *Consider a linear classification example where* $\mathcal{X}\times\mathcal{Y} = \mathbb{R}^d \times \{-1, 1\}$, $\Theta \subset \mathbb{R}_+$, $f_\theta = \mathcal{A}\left(\{(x_i, y_i)\}_{i\in\texttt{TRAIN}}, \theta\right)$ *where* $f_\theta(x) = \langle w_\theta, x\rangle$ *with* $w_\theta = \arg\min_w \frac{1}{|\texttt{TRAIN}|}\sum_{i\in\texttt{TRAIN}}\max(0, 1 - y_i\langle w, x_i\rangle) + \theta||w||_2^2$, *and finally* $\widehat{\theta} = \arg\min_{\theta\in\Theta}\frac{1}{|\texttt{VAL}|}\sum_{i\in\texttt{VAL}}\mathbf{1}\{y\, f_\theta(x) < 0\}$.

In the simple above example involving a single hyperparameter, we emphasize that for each θ we have that f_θ can be efficiently computed using an iterative algorithm [88], however, the selection of $\widehat{f}$ is the minimization of a function that is not necessarily even

continuous, much less convex. This pattern is more often the rule than the exception. We next attempt to generalize and exploit this observation.

5.4.1 Posing as a best arm non-stochastic bandits problem

Let us assume that the algorithm $\mathcal{A}$ is iterative so that for a given $\{(x_i, y_i)\}_{i \in \texttt{TRAIN}}$ and θ, the algorithm outputs a function $f_{\theta,t}$ every iteration $t > 1$ and we may compute

$$\ell_{\theta,t} = \frac{1}{|\texttt{VAL}|} \sum_{i \in \texttt{VAL}} \texttt{loss}(f_{\theta,t}(x_i), y_i).$$

We assume that the limit $\lim_{t\to\infty} \ell_{\theta,t}$ exists[1] and is equal to $\frac{1}{|\texttt{VAL}|} \sum_{i \in \texttt{VAL}} \texttt{loss}(f_\theta(x_i), y_i)$.

With this transformation we are in the position to put the hyperparameter optimization problem into the framework of Figure 5.2 and, namely, the non-stochastic best-arm identification formulation developed in the above sections. We generate the arms (different hyperparameter settings) uniformly at random (possibly on a log scale) from within the region of valid hyperparameters (i.e. all hyperparameters within some minimum and maximum ranges) and sample enough arms to ensure a sufficient cover of the space [76]. Alternatively, one could input a uniform grid over the parameters of interest. We note that random search and grid search remain the default choices for many open source machine learning packages such as LibSVM [89], scikit-learn [90] and MLlib [91]. As described in Figure 5.2, the bandit algorithm will choose I_t, and we will use the convention that $J_t = \arg\min_\theta \ell_{\theta,T_\theta}$. The arm selected by J_t will be evaluated on the test set following the work-flow introduced above.

[1] We note that $f_\theta = \lim_{t\to\infty} f_{\theta,t}$ is not enough to conclude that $\lim_{t\to\infty} \ell_{\theta,t}$ exists (for instance, for classification with 0/1 loss this is not necessarily true) but these technical issues can usually be usurped for real datasets and losses (for instance, by replacing $\mathbf{1}\{z < 0\}$ with a very steep sigmoid). We ignore this technicality in our experiments.

5.4.2 Related work

We aim to leverage the iterative nature of standard machine learning algorithms to speed up hyperparameter optimization in a robust and principled fashion. We now review related work in the context of our results. In Section 5.3.3 we show that no algorithm can provably identify a hyperparameter with a value within ϵ of the optimal without known, explicit functions γ_i, which means no algorithm can reject a hyperparameter setting with absolute confidence without making potentially unrealistic assumptions. [78] explicitly defines the γ_i functions in an ad-hoc, algorithm-specific, and data-specific fashion which leads to strong ϵ-good claims. A related line of work explicitly defines γ_i-like functions for optimizing the computational efficiency of structural risk minimization, yielding bounds [77]. We stress that these results are only as good as the tightness and correctness of the γ_i bounds, and we view our work as an empirical, data-driven driven approach to the pursuits of [77]. Also, [79] empirically studies an early stopping heuristic for hyperparameter optimization similar in spirit to the Successive Halving algorithm.

We further note that we fix the hyperparameter settings (or arms) under consideration and adaptively allocate our budget to each arm. In contrast, Bayesian optimization advocates choosing hyperparameter settings adaptively, but with the exception of [78], allocates a fixed budget to each selected hyperparameter setting [72, 73, 74, 75, 76]. These Bayesian optimization methods, though heuristic in nature as they attempt to simultaneously fit and optimize a non-convex and potentially high-dimensional function, yield promising empirical results. We view our approach as complementary and orthogonal to the method used for choosing hyperparameter settings, and extending our approach in a principled fashion to adaptively choose arms, e.g., in a mini-batch setting, is an

interesting avenue for future work.

5.5 Experiment results

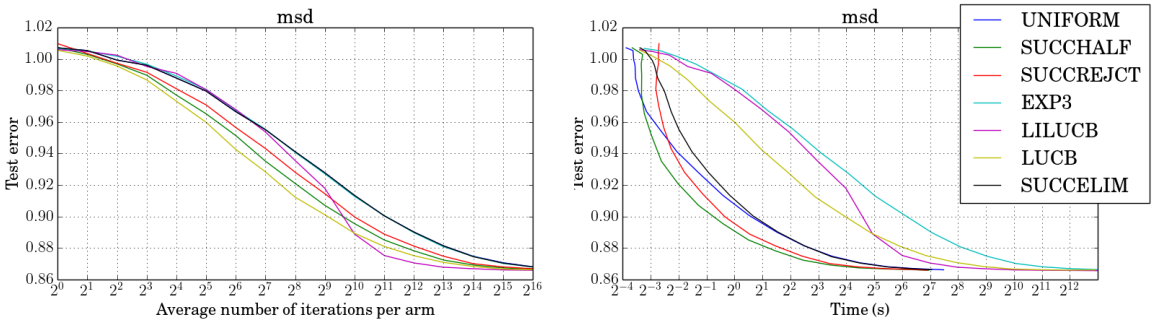


Figure 5.4: Ridge Regression. Test error with respect to both the number of iterations (left) and wall-clock time (right). Note that in the left plot, uniform, EXP3, and Successive Elimination are plotted on top of each other.

In this section we compare the proposed algorithm to a number of other algorithms, including the baseline uniform allocation strategy, on a number of supervised learning hyperparameter optimization problems using the experimental setup outlined in Section 5.4.1. Each experiment was implemented in Python and run in parallel using the multiprocessing library on an Amazon EC2 c3.8xlarge instance with 32 cores and 60 GB of memory. In all cases, full datasets were partitioned into a training-base dataset and a test (`TEST`) dataset with a 90/10 split. The training-base dataset was then partitioned into a training (`TRAIN`) and validation (`VAL`) datasets with an 80/20 split. All plots report loss on the test error.

To evaluate the different search algorithms' performance, we fix a total budget of iterations and allow the search algorithms to decide how to divide it up amongst the different arms. The curves are produced by implementing the doubling trick by simply

doubling the measurement budget each time. For the purpose of interpretability, we reset all iteration counters to 0 at each doubling of the budget, i.e., we do not warm start upon doubling. All datasets, aside from the collaborative filtering experiments, are normalized so that each dimension has mean 0 and variance 1.

Ridge regression

We first consider a ridge regression problem trained with stochastic gradient descent on this objective function with step size $.01/\sqrt{2+T_\lambda}$. The ℓ_2 penalty hyperparameter $\lambda \in [10^{-6}, 10^0]$ was chosen uniformly at random on a log scale per trial, wth 10 values (i.e., arms) selected per trial. We use the Million Song Dataset year prediction task [92] where we have down sampled the dataset by a factor of 10 and normalized the years such that they are mean zero and variance 1 with respect to the training set. The experiment was repeated for 32 trials. Error on the `VAL` and `TEST` was calculated using mean-squared-error. In the left panel of Figure 5.4 we note that LUCB, lil'UCB perform the best in the sense that they achieve a small test error two to four times faster, in terms of iterations, than most other methods. However, in the right panel the same data is plotted but with respect to wall-clock time rather than iterations and we now observe that Successive Halving and Successive Rejects are the top performers. This is explainable by Table 5.1: EXP3, lil'UCB, and LUCB must evaluate the validation loss on every iteration requiring much greater compute time. This pattern is observed in all experiments so in the sequel we only consider the uniform allocation, Successive Halving, and Successive Rejects algorithm.

Kernel SVM

We now consider learning a kernel SVM using the RBF kernel $\kappa_\gamma(x, z) = e^{-\gamma||x-z||_2^2}$. The SVM is trained using Pegasos [88] with ℓ_2 penalty hyperparameter $\lambda \in [10^{-6}, 10^0]$ and kernel width $\gamma \in [10^0, 10^3]$ both chosen uniformly at random on a log scale per trial. Each hyperparameter was allocated 10 samples resulting in $10^2 = 100$ total arms. The experiment was repeated for 64 trials. Error on the `VAL` and `TEST` was calculated using 0/1 loss. Kernel evaluations were computed online (i.e. not precomputed and stored). We observe in Figure 5.5 that Successive Halving obtains the same low error more than an order of magnitude faster than both uniform and Successive Rejects with respect to wall-clock time, despite Successive Halving and Success Rejects performing comparably in terms of iterations (not plotted).

Collaborative filtering

We next consider a matrix completion problem using the Movielens 100k dataset trained using stochastic gradient descent on the bi-convex objective with step sizes as described in [93]. To account for the non-convex objective, we initialize the user and item variables with entries drawn from a normal distribution with variance σ^2/d, hence each arm has hyperparameters d (rank), λ (Frobenium norm regularization), and σ (initial conditions). $d \in [2, 50]$ and $\sigma \in [.01, 3]$ were chosen uniformly at random from a linear scale, and $\lambda \in [10^{-6}, 10^0]$ was chosen uniformly at random on a log scale. Each hyperparameter is given 4 samples resulting in $4^3 = 64$ total arms. The experiment was repeated for 32 trials. Error on the `VAL` and `TEST` was calculated using mean-squared-error. One observes in Figure 5.6 that the uniform allocation takes two to eight times longer to achieve a

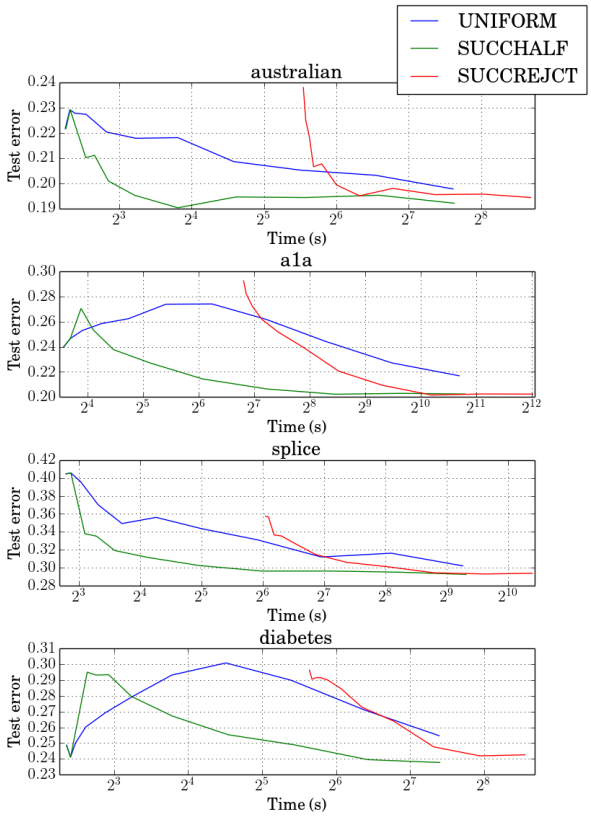


Figure 5.5: Kernel SVM. Successive Halving and Successive Rejects are separated by an order of magnitude in wall-clock time.

particular error rate than Successive Halving or Successive Rejects.

5.6 Discussion

Our theoretical results are presented in terms of $\max_i \gamma_i(t)$. An interesting future direction is to consider algorithms and analyses that take into account the specific convergence rates $\gamma_i(t)$ of each arm, analogous to considering arms with different variances in the stochastic case [67]. Incorporating pairwise switching costs into the framework could model the time of moving very large intermediate models in and out of memory to perform iterations, along with the degree to which resources are shared across various

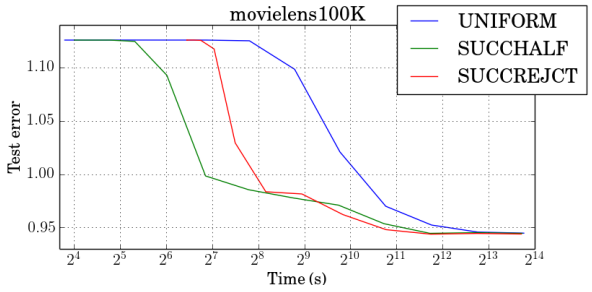


Figure 5.6: Matrix Completion (bi-convex formulation).

models (resulting in lower switching costs). Finally, balancing solution quality and time by adaptively sampling hyperparameters as is done in Bayesian methods is of considerable practical interest.

5.7 Bibliographical Remarks

The work presented in this chapter was based on the author's preprint

- Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *arXiv preprint arXiv:1502.07943*, 2015

that is currently under review.

Part III

Stochastic Optimization with Comparative Judgments

Chapter 6

Dueling Bandits with the Borda Voting Rule

In this chapter we revisit the use of pairwise comparisons under the bandit framework. As discussed in Chapter 1, pairwise comparisons can be an excellent way of collecting information from humans, and from time to time we wish to identify a "most" preferred item by polling the crowd. For instance, suppose the the computer sciences department decided to make a new T-shirt and held a contest among contenders in which the students decided which T-shirt to use. If the amount of T-shirts is large, then the users won't be able to rate all the T-shirts so the department asks students to rate on a, say, 1-10 scale, this scale could change because students are not calibrated: they may disagree on what a score of "7" means and moreover, they may change their own scales over time as they see more T-shirts. With pairwise comparisons, this calibration issue is nonexistent. In Chapter 4 we learned how to adaptively select T-shirts to show to students in order to identify the top rated T-shirt as soon as possible. However, with pairwise comparisons when there are no scores observed, there is no obvious strategy. Indeed, defining a map from pairwise comparisons to a "winner" has been an intensely studied topic in social choice theory for hundreds of years [94]. The candidates for mapping comparisons to "winners" range widely from statistically rigorous definitions that make very few modeling

assumptions to heuristics that seem to work well in practice [95].

In this chapter we use the Borda rule to map pairwise comparisons to a winner and design an algorithm with this metric in mind. To make this mathematically rigorous, we assume that whenever we request if a is preferred to b, we receive an independent Bernoulli random variable $X_{a,b}$ with expectation $P_{a,b}$ (note, repeated draws of $X_{a,b}$ need not be identically distributed, but independence is essential, as is natural if each response is coming from a different person in a crowd). We define the Borda score of item i with respect to the other n objects as $s_i := \frac{1}{n-1}\sum_{j\neq i} p_{i,j}$ so that s_i can be interpreted as the expected value of the result of picking a second object J not equal to i uniformly at random from $[n] \setminus \{i\}$ and comparing with i, i.e. $s_i = \mathbb{E}[p_{i,J}] = \mathbb{E}[\mathbb{E}[X_{i,j}|J = j]]$. The careful reader will realize that we can turn this "Borda bandits" game into the standard multi-armed bandit game of 4 where $\mu_i = s_i$ and a pull of the ith arm is equal to $X_{i,J}$ where J is drawn uniformly at random from $[n] \setminus \{i\}$. This chapter explores whether this is the best we can do or whether there is additional structure that can be taken advantage of.

6.1 Introduction

The dueling bandit is a variation of the classic multi-armed bandit problem in which the actions are noisy comparisons between arms, rather than observations from the arms themselves [96]. Each action provides 1 bit indicating which of two arms is probably better. For example, the arms could represent objects and the bits could be responses from people asked to compare pairs of objects. In this paper, we focus on the pure *exploration* problem of finding the "best" arm from noisy pairwise comparisons. This

problem is different from the *explore-exploit* problem studied in [96]. There can be different notions of "best" in the dueling framework, including the Condorcet and Borda criteria (defined below).

Most of the dueling-bandit algorithms are primarily concerned with finding the Condorcet winner (the arm that is probably as good or better than every other arm). There are two drawbacks to this. First, a Condorcet winner does not exist unless the underlying probability matrix governing the outcomes of pairwise comparisons satisfies certain restrictions. These restrictions may not be met in many situations. In fact, we show that a Condorcet winner doesn't exist in our experiment with real data presented below. Second, the best known upper bounds on the sample complexity of finding the Condorcet winner (assuming it exists) grow quadratically (at least) with the number of arms. This makes Condorcet algorithms impractical for large numbers of arms.

To address these drawbacks, we consider the Borda criterion instead. The Borda score of an arm is the probability that the arm is preferred to another arm chosen uniformly at random. A Borda winner (arm with the largest Borda score) always exists for every possible probability matrix. We assume throughout this paper that there exists a unique Borda winner. Finding the Borda winner with probability at least $1-\delta$ can be reduced to solving an instance of the standard multi-armed bandit problem resulting in a sufficient sample complexity of $O\left(\sum_{i>1}(s_1-s_i)^{-2}\log\left(\log((s_1-s_i)^{-2})/\delta\right)\right)$, where s_i denotes Borda score of arm i and $s_1 > s_2 > \cdots > s_n$ are the scores in descending order (c.f. Chapter 4 or [61, 97]). In favorable cases, for instance, if $s_1 - s_i \geq c$, a constant for all $i > 1$, then this sample complexity is linear in n as opposed to the quadratic sample complexity necessary to find the Condorcet winner. In this paper we show that this upper bound is essentially tight, thereby apparently "closing" the Borda winner

identification problem. However, in this paper we consider a specific type of structure that is motivated by its existence in real datasets that complicates this apparently simple story. In particular, we show that the reduction to a standard multi-armed bandit problem can result in very bad performance when compared to an algorithm that exploits this observed structure.

We explore the sample complexity dependence in more detail and consider structural constraints on the matrix (a particular form of sparsity natural to this problem) that can significantly reduce the sample complexity. The sparsity model captures the commonly observed behavior in elections in which there are a small set of "top" candidates that are competing to be the winner but only differ on a small number of attributes, while a large set of "others" are mostly irrelevant as far as predicting the winner is concerned in the sense that they would always lose in a pairwise matchup against one of the "top" candidates.

This motivates a new algorithm called Successive Elimination with Comparison Sparsity (SECS). SECS takes advantage of this structure by determining which of two arms is better on the basis of their performance with respect to a sparse set of "comparison" arms. Experimental results with real data demonstrate the practicality of the sparsity model and show that SECS can provide significant improvements over standard approaches.

The main contributions of this work are as follows:

- A distribution dependent lower bound for the sample complexity of identifying the Borda winner that essentially shows that the Borda reduction to the standard multi-armed bandit problem (explained in detail later) is essentially optimal up to logarithmic factors, given no prior structural information.

- A new structural assumption for the n-armed dueling bandits problem in which the top arms can be distinguished by duels with a sparse set of other arms.
- An algorithm for the dueling bandits problem under this assumption, with theoretical performance guarantees showing significant sample complexity improvements compared to naive reductions to standard multi-armed bandit algorithms.
- Experimental results, based on real-world applications, demonstrating the superior performance of our algorithm compared to existing methods.

6.2 Problem Setup

The *n-armed dueling bandits problem* [96] is a modification of the *n-armed bandit problem*, where instead of pulling a single arm, we choose a pair of arms (i, j) to duel, and receive one bit indicating which of the two is better or preferred, with the probability of i winning the duel is equal to a constant $p_{i,j}$ and that of j equal to $p_{j,i} = 1 - p_{i,j}$. We define the *probabilty matrix* $P = [p_{i,j}]$, whose (i, j)th entry is $p_{i,j}$.

Almost all existing n-armed dueling bandit methods [96, 98, 99, 100, 101] focus on the explore-exploit problem and furthermore make a variety of assumptions on the preference matrix P. In particular, those works assume the existence of a Condorcet winner: an arm, c, such that $p_{c,j} > \frac{1}{2}$ for all $j \neq c$. The *Borda* winner is an arm b that satisfies $\sum_{j \neq b} p_{b,j} \geq \sum_{j \neq i} p_{i,j}$ for all $i = 1, \cdots, n$. In other words, the Borda winner is the arm with the highest average probability of winning against other arms, or said another way, the arm that has the highest probability of winning against an arm selected uniformly at random from the remaining arms. The Condorcet winner has been given

more attention than the Borda, the reasons being: 1) Given a choice between the Borda and the Condorcet winner, the latter is preferred in a direct comparison between the two. 2) As pointed out in [99, 100] the Borda winner can be found by reducing the dueling bandit problem to a standard multi-armed bandit problem as follows.

Definition 6.1. Borda Reduction. *The action of pulling arm i with reward $\frac{1}{n-1}\sum_{j\neq i} p_{i,j}$ can be simulated by dueling arm i with another arm chosen uniformly at random.*

However, we feel that the Borda problem has received far less attention than it deserves. Firstly, the Borda winner *always exists*, the Condorcet does not. For example, a Condorcet winner does not exist in the MSLR-WEB10k datasets considered in this paper. Assuming the existence of a Condorcet winner severely restricts the class of allowed P matrices: only those P matrices are allowed which have a row with all entries $\geq \frac{1}{2}$. In fact, [96, 98] require that the comparison probabilities $p_{i,j}$ satisfy additional transitivity conditions that are often violated in practice. Secondly, there are many cases where the Borda winner and the Condorcet winner are distinct, and the Borda winner would be preferred in many cases. Lets assume that arm c is the Condorcet winner, with $p_{c,i} = 0.51$ for $i \neq c$. Let arm b be the Borda winner with $p_{b,i} = 1$ for $i \neq b, c$, and $p_{b,c} = 0.49$. It is reasonable that arm c is only marginally better than the other arms, while arm b is significantly preferred over all other arms except against arm c where it is marginally rejected. In this example - chosen extreme to highlight the pervasiveness of situations where the Borda arm is preferred - it is clear that arm b should be the winner: think of the arms representing objects being contested such as t-shirt designs, and the P matrix is generated by showing users a pair of items and asking them to choose the better among the two. This example also shows that the Borda winner is more robust

to estimation errors in the P matrix (for instance, when the P matrix is estimated by asking a small sample of the entire population to vote among pairwise choices). The Condorcet winner is sensitive to entries in the Condorcet arm's row that are close to $\frac{1}{2}$, which is not the case for the Borda winner. Finally, there are important cases (explained next) where the winner can be found in fewer number of duels than would be required by Borda reduction.

6.3 Motivation

We define the *Borda score* of an arm i to be the probability of the i^{th} arm winning a duel with another arm chosen uniformly at random:

$$s_i = \frac{1}{n-1} \sum_{j \neq i} p_{i,j} \, .$$

Without loss of generality, we assume that $s_1 > s_2 \geq \cdots \geq s_n$ but that this ordering is unknown to the algorithm. As mentioned above, if the Borda reduction is used then the dueling bandit problem becomes a regular multi-armed bandit problem and lower bounds for the multi-armed bandit problem [67, 102] suggest that the number of samples required should scale like $\Omega\left(\sum_{i \neq 1} \frac{1}{(s_1 - s_i)^2} \log \frac{1}{\delta}\right)$, which depends only on the Borda scores, and not the individual entries of the preference matrix. This would imply that any preference matrix P with Borda scores s_i is just as hard as another matrix P' with Borda scores s'_i as long as $(s_1 - s_i) = (s'_1 - s'_i)$. Of course, this lower bound only applies to algorithms using the Borda reduction, and not any algorithm for identifying the Borda winner that may, for instance, collect the duels in a more deliberate way. Next we consider specific P

$$P_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \cdots & n \end{array} \\ \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{3}{4} & \cdots & \frac{3}{4}+\epsilon \\ \frac{1}{2} & \frac{1}{2} & \frac{3}{4} & \cdots & \frac{3}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{4}-\epsilon & \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} \end{pmatrix} \end{array} \begin{array}{cc} s_i & s_1 - s_i \\ \frac{\frac{1}{2}+\epsilon}{n-1} + \frac{3}{4}\frac{n-2}{n-1} & 0 \\ \frac{\frac{1}{2}}{n-1} + \frac{3}{4}\frac{n-2}{n-1} & \frac{\epsilon}{n-1} \\ \frac{1}{2}\frac{n-2}{n-1} & \frac{\frac{1}{2}+\epsilon}{n-1} + \frac{1}{4}\frac{n-2}{n-1} \\ \vdots & \vdots \\ -\frac{\epsilon}{n-1} + \frac{1}{2}\frac{n-2}{n-1} & \frac{\frac{1}{2}+2\epsilon}{n-1} + \frac{1}{4}\frac{n-2}{n-1} \end{array} \tag{6.1}$$

$$P_2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \cdots & n \end{array} \\ \begin{pmatrix} \frac{1}{2} & \frac{1}{2}+\frac{\epsilon}{n-1} & \frac{3}{4}+\frac{\epsilon}{n-1} & \cdots & \frac{3}{4}+\frac{\epsilon}{n-1} \\ \frac{1}{2}-\frac{\epsilon}{n-1} & \frac{1}{2} & \frac{3}{4} & \cdots & \frac{3}{4} \\ \frac{1}{4}-\frac{\epsilon}{n-1} & \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{4}-\frac{\epsilon}{n-1} & \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} \end{pmatrix} \end{array} \begin{array}{cc} s_i & s_1 - s_i \\ \frac{\frac{1}{2}+\epsilon}{n-1} + \frac{3}{4}\frac{n-2}{n-1} & 0 \\ \frac{\frac{1}{2}-\frac{\epsilon}{n-1}}{n-1} + \frac{3}{4}\frac{n-2}{n-1} & \frac{\epsilon}{n-1} + \frac{\epsilon}{(n-1)^2} \\ \frac{-\frac{\epsilon}{n-1}}{n-1} + \frac{1}{2}\frac{n-2}{n-1} & \frac{\frac{1}{2}+\epsilon+\frac{\epsilon}{n-1}}{n-1} + \frac{1}{4}\frac{n-2}{n-1} \\ \vdots & \vdots \\ \frac{-\frac{\epsilon}{n-1}}{n-1} + \frac{1}{2}\frac{n-2}{n-1} & \frac{\frac{1}{2}+\epsilon+\frac{\epsilon}{n-1}}{n-1} + \frac{1}{4}\frac{n-2}{n-1} \end{array} \tag{6.2}$$

matrices that exhibit two very different kinds of structure but have the same differences in Borda scores which motivates the structure considered in this paper.

6.3.1 Preference Matrix P known up to permutation of indices

Shown below in equations (6.1) and (6.2) are two preference matrices P_1 and P_2 indexed by the number of arms n that essentially have the same Borda gaps – $(s_1 - s_i)$ is either like $\frac{\epsilon}{n}$ or approximately $1/4$ – but we will argue that P_1 is much "easier" than P_2 in a certain sense (assume ϵ is an unknown constant, like $\epsilon = 1/5$). Specifically, if given P_1 and P_2 up to a permutation of the labels of their indices (i.e. given $\Lambda P_1 \Lambda^T$ for some unknown permutation matrix Λ), how many comparisons does it take to find the Borda winner in each case for different values of n?

Recall from above that if we ignore the fact that we know the matrices up to a permutation and use the Borda reduction technique, we can use a multi-armed bandit algorithm (e.g. Chapter 4 or [61, 97]) and find the best arm for both P_1 and P_2 using $O\left(n^2 \log(\log(n))\right)$ samples. We next argue that given P_1 and P_2 up to a permutation, there exists an algorithm that can identify the Borda winner of P_1 with just $O(n \log(n))$ samples while the identification of the Borda winner for P_2 requires at least $\Omega(n^2)$ samples. This shows that given the probability matrices up to a permutation, the sample complexity of identifying the Borda winner does not rely just on the Borda differences, but on the particular structure of the probability matrix.

Consider P_1. We claim that there exists a procedure that exploits the structure of the matrix to find the best arm of P_1 using just $O(n \log(n))$ samples. Here's how: For each arm, duel it with $32 \log \frac{n}{\delta}$ other arms chosen uniformly at random. By Hoeffding's

inequality, with probability at least $1-\delta$ our empirical estimate of the Borda score will be within $1/8$ of its true value for all n arms and we can remove the bottom $(n-2)$ arms due to the fact that their Borda gaps exceed $1/4$. Having reduced the possible winners to just two arms, we can identify which rows in the matrix they correspond to and duel each of these two arms against all of the remaining $(n-2)$ arms $O(\frac{1}{\epsilon^2})$ times to find out which one has the larger Borda score using just $O\left(\frac{2(n-2)}{\epsilon^2}\right)$ samples, giving an overall sample complexity of $O\left(n\log n\right)$. We have improved the sample complexity from $O(n^2\log(\log(n)))$ using the Borda reduction to just $O(n\log(n))$.

Consider P_2. We claim that given this matrix up to a permutation of its indices, no algorithm can determine the winner of P_2 without requesting $\Omega(n^2)$ samples. To see this, suppose an oracle has made the problem easier by reducing the problem down to just the top two rows of the P_2 matrix. This is a binary hypothesis test for which Fano's inequality implies that to guarantee that the probability of error is not above some constant level, the number of samples to identify the Borda winner must scale like $\min_{j\in[n]\setminus\{1,2\}}\frac{1}{KL(p_{1,j},p_{2,j})}\geq\min_{j\in[n]\setminus\{1,2\}}\frac{c}{(p_{1,j}-p_{2,j})^2}=\Omega((n/\epsilon)^2)$ where the inequality holds for some c by Lemma 6.6 in the Appendix.

We just argued that the structure of the P matrix, and not just the Borda gaps, can dramatically influence the sample complexity of finding the Borda winner. This leads us to ask the question: if we don't know anything about the P matrix beforehand (i.e. do not know the matrix up to a permutation of its indices), can we learn and exploit this kind of structural information in an online fashion and improve over the Borda reduction scheme? The answer is no, as we argue next.

6.3.2 Distribution-Dependent Lower Bound

We prove a distribution-dependent lower bound on the complexity of finding the best Borda arm for a general P matrix. This is a result important in its own right as it shows that the lower bound obtained for an algorithm using the Borda reduction is tight, that is, this result implies that barring any structural assumptions, the Borda reduction is optimal.

Definition 6.2. δ-PAC dueling bandits algorithm*: A δ-PAC dueling bandits algorithm is an algorithm that selects duels between arms and based on the outcomes finds the Borda winner with probability greater than or equal to $1-\delta$.*

Theorem 6.3. *(Distribution-Dependent Lower Bound) Consider a matrix P such that $\frac{3}{8} \leq p_{i,j} \leq \frac{5}{8}, \forall i,j \in [n]$ with $n \geq 4$. Let τ be the total number of duels. Then for $\delta \leq 0.15$, any δ-PAC dueling bandits algorithm to find the Borda winner has*

$$\mathbb{E}_P[\tau] \geq C \log \frac{1}{2\delta} \sum_{i \neq 1} \frac{1}{(s_1 - s_i)^2}$$

where $s_i = \frac{1}{n-1} \sum_{j \neq i} p_{i,j}$ denotes the Borda score of arm i. Furthermore, C can be chosen to be $1/90$.

Remark 1. *Recalling the sample complexity of identifying the best arm for the Borda reduction scheme, Theorem 6.3 says that for any two preference matrices P and P' that have the same Borda scores, the sample complexity to identify the best arm of either of them is nearly the same, regardless of how the how the matrices are structured. In particular, the theorem implies that any algorithm that does not make any additional*

structural assumptions requires as many samples to find the best arm of P_1 as it does to find the best arm of P_2, where P_1, P_2 are the matrices of above. Next we argue that the particular structure found in P_1 is an extreme case of a more general structural phenomenon found in real datasets and that it is a natural structure to assume and design algorithms to exploit.

Before proving the theorem we need a few technical lemmas. At the heart of the proof of the lower bound is Lemma 1 of [67] restated here for completeness.

Lemma 6.4. *Let ν and ν' be two bandit models defined over n arms. Let σ be a stopping time with respect to $(\mathcal{F}_t)$ and let $A \in \mathcal{F}_\sigma$ be an event such that $0 < \mathbb{P}_\nu(A) < 1$. Then*

$$\sum_{a=1}^{n} \mathbb{E}_\nu[N_a(\sigma)] KL(\nu_a, \nu'_a) \geq d(\mathbb{P}_\nu(A), \mathbb{P}_{\nu'}(A))$$

where $d(x, y) = x \log(x/y) + (1 - x) \log((1 - x)/(1 - y))$.

Note that the function d is exactly the KL-divergence between two Bernoulli distributions.

Corollary 6.5. *Let $N_{i,j} = N_{j,i}$ denote the number of duels between arms i and j. For the duelling bandits problem with n arms, we have $\frac{(n-1)(n-2)}{2}$ free parameters (or arms). These are the numbers in the upper triangle of the P matrix. Then, if P' is an alternate matrix, we have from Lemma 6.4,*

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}_P[N_{i,j}] d(p_{i,j}, p'_{i,j}) \geq d(\mathbb{P}_P(A), \mathbb{P}_{P'}(A))$$

The above corollary relates the cumulative number of duels of a subset of arms to the uncertainty between the actual distribution and an alternative distribution. In deference

to interpretability rather than preciseness, we will use the following bound of the KL divergence.

Lemma 6.6. *(Upper bound on KL Divergence for Bernoullis) Consider two Bernoulli random variables with means p and q, $0 < p, q < 1$. Then $d(p, q) \leq \frac{(p-q)^2}{q(1-q)}$.*

Proof.

$$d(p,q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q} \leq p\frac{p-q}{q} + (1-p)\frac{q-p}{1-q} = \frac{(p-q)^2}{q(1-q)}$$

where we use the fact that $\log x \leq x - 1$ for $x > 0$. □

We are now in a position to restate and prove the lower bound theorem.

Proof of Theorem 6.3. Consider an alternate hypothesis P' where arm b is the best arm, and such that P' differs from P only in the indices $\{bj : j \notin \{1, b\}\}$. Note that the Borda score of arm 1 is unaffected in the alternate hypothesis. Corollary 6.5 then gives us:

$$\sum_{j\in[n]\setminus\{1,b\}} \mathbb{E}_P[N_{b,j}]d(p_{b,j}, p'_{b,j}) \geq d(\mathbb{P}(A), \mathbb{P}(A')) \tag{6.3}$$

Let A be the event that the algorithm selects arm 1 as the best arm. Since we assume a δ-PAC algorithm, $\mathbb{P}_P(A) \geq 1 - \delta$, $\mathbb{P}_{P'}(A) \leq \delta$. It can be shown that for $\delta \leq 0.15$, $d(\mathbb{P}_P(A), \mathbb{P}_{P'}(A)) \geq \log\frac{1}{2\delta}$.

Define $N_b = \sum_{j \neq b} N_{b,j}$. Consider

$$\begin{aligned}
\left(\max_{j \notin \{1,b\}} \frac{(p_{b,j} - p'_{b,j})^2}{p'_{b,j}(1 - p'_{b,j})} \right) \mathbb{E}_P[N_b] &\geq \left(\max_{j \notin \{1,b\}} d(p_{b,j}, p'_{b,j}) \right) \mathbb{E}_P[N_b] \\
&= \left(\max_{j \notin \{1,b\}} d(p_{b,j}, p'_{b,j}) \right) \left(\sum_{j \neq b} \mathbb{E}_P[N_{b,j}] \right) \\
&\geq \left(\max_{j \notin \{1,b\}} d(p_{b,j}, p'_{b,j}) \right) \left(\sum_{j \notin \{1,b\}} \mathbb{E}_P[N_{b,j}] \right) \\
&\geq \sum_{j \in [n] \setminus \{1,b\}} \mathbb{E}_P[N_{b,j}] d(p_{b,j}, p'_{b,j}) \\
&\geq \log \frac{1}{2\delta}. \quad \text{(by (6.3))} \qquad (6.4)
\end{aligned}$$

In particular, choose $p'_{b,j} = p_{b,j} + \frac{n-1}{n-2}(s_1 - s_b) + \varepsilon$, $j \notin \{1, b\}$. As required, under hypothesis P', arm b is the best arm.

Since $p_{b,j} \leq \frac{5}{8}$, $s_1 \leq \frac{5}{8}$, and $s_b \geq \frac{3}{8}$, as $\varepsilon \searrow 0$, $\lim_{\varepsilon \searrow 0} p'_{b,j} \leq \frac{15}{16}$. This implies $\frac{1}{p'_{b,j}(1-p'_{b,j})} \leq \frac{256}{15} \leq 20$. (**??**) implies

$$\begin{aligned}
20 \left(\frac{n-1}{n-2}(s_1 - s_b) + \varepsilon \right)^2 \mathbb{E}_P[N_b] &\geq \log \frac{1}{2\delta} \\
\Rightarrow \mathbb{E}_P[N_b] &\geq \frac{1}{20} \left(\frac{n-2}{n-1} \right)^2 \frac{1}{(s_1 - s_b)^2} \log \frac{1}{2\delta} \qquad (6.5)
\end{aligned}$$

where we let $\varepsilon \searrow 0$.

Finally, iterating over all arms $b \neq 1$, we have

$$\mathbb{E}_P[\tau] = \frac{1}{2}\sum_{b=1}^{n}\sum_{j\neq b}\mathbb{E}_P[N_{b,j}] = \frac{1}{2}\sum_{b=1}^{n}\mathbb{E}_P[N_b]$$
$$\geq \frac{1}{2}\sum_{b=2}^{n}\mathbb{E}_P[N_b] \geq \frac{1}{40}\left(\frac{n-2}{n-1}\right)^2\left(\sum_{b\neq 1}\frac{1}{(s_1-s_b)^2}\right)\log\frac{1}{2\delta}.$$

□

6.3.3 Motivation from Real-World Data

The matrices P_1 and P_2 above illustrate a key structural aspect that can make it easier to find the Borda winner. If the arms with the top Borda scores are distinguished by duels with a small subset of the arms (as exemplified in P_1), then finding the Borda winner may be easier than in the general case. Before formalizing a model for this sort of structure, let us look at two real-world datasets, which motivate the model.

We consider the Microsoft Learning to Rank web search datasets MSLR-WEB10k [103] and MQ2008-list [104] (see the experimental section for a descrptions). Each dataset is used to construct a corresponding probability matrix P. We use these datasets to test the hypothesis that comparisons with a small subset of the arms may suffice to determine which of two arms has a greater Borda score.

Specifically, we will consider the Borda score of the best arm (arm 1) and every other arm. For any other arm $i > 1$ and any positive integer $k \in [n-2]$, let $\Omega_{i,k}$ be a set of cardinality k containing the indices $j \in [n] \setminus \{1, i\}$ with the k largest discrepancies $|p_{1,j} - p_{i,j}|$. These are the duels that, individually, display the greatest differences between arm 1 and i. For each k, define $\alpha_i(k) = 2(p_{1,i} - \frac{1}{2}) + \sum_{j\in\Omega_{i,k}}(p_{1,j} - p_{i,j})$. If the hypothesis

holds, then the duels with a small number of (appropriately chosen) arms should indicate that arm 1 is better than arm i. In other words, $\alpha_i(k)$ should become and stay positive as soon as k reaches a relatively small value. Plots of these α_i curves for two datasets are presented in Figures 6.1, and indicate that the Borda winner is apparent for small k. This behavior is explained by the fact that the individual discrepancies $|p_{1,j} - p_{i,j}|$, decay quickly when ordered from largest to smallest, as shown in Figure 6.2.

The take away message is that it is unnecessary to estimate the difference or gap between the Borda scores of two arms. It suffices to compute the *partial* Borda gap based on duels with a small subset of the arms. An appropriately chosen subset of the duels will correctly indicate which arm has a larger Borda score. The algorithm proposed in the next section automatically exploits this structure.

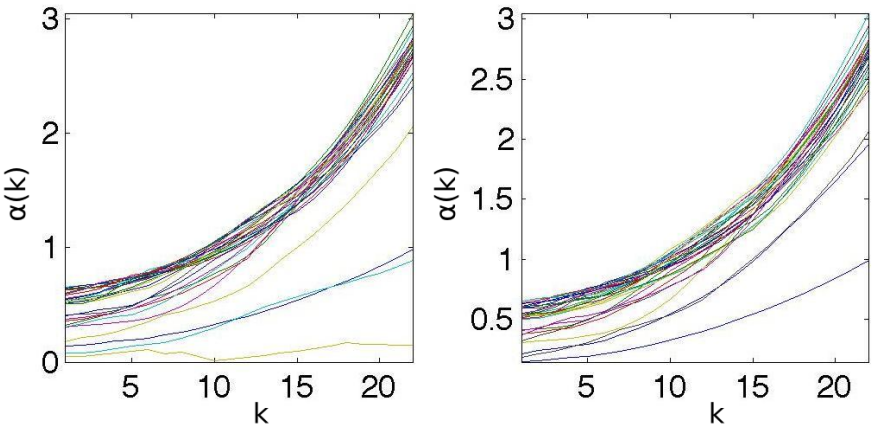


Figure 6.1: Plots of $\alpha_i(k) = 2(p_{1,i} - \frac{1}{2}) + \sum_{j \in \Omega_{i,k}} (p_{1,j} - p_{1,j})$ vs. k for 30 randomly chosen arms (for visualization purposes); MSLR-WEB10k on left, MQ2008-list on right. The curves are strictly positive after a small number of duels.

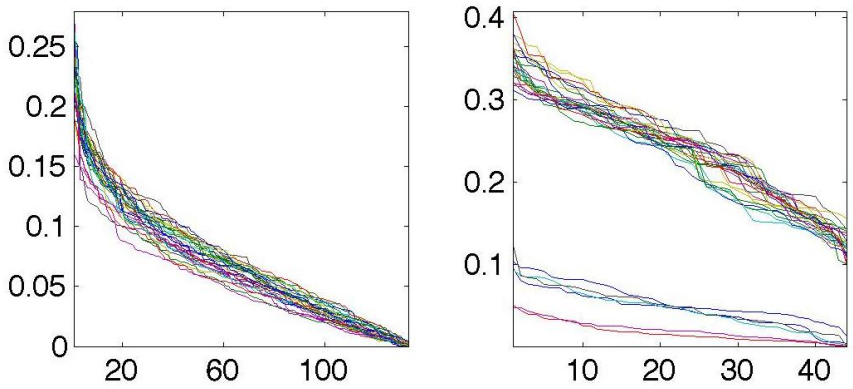


Figure 6.2: Plots of discrepancies $|p_{1,j} - p_{i,j}|$ in descending order for 30 randomly chosen arms (for visualization purposes); MSLR-WEB10k on left, MQ2008-list on right.

6.4 Algorithm and Analysis

In this section we propose a new algorithm that exploits the kind of structure just described above and prove a sample complexity bound. The algorithm is inspired by the Successive Elimination (SE) algorithm of [83] for standard multi-armed bandit problems. Essentially, the proposed algorithm below implements SE with the Borda reduction and an additional elimination criterion that exploits sparsity (condition 1 in the algorithm). We call the algorithm Successive Elimination with Comparison Sparsity (SECS).

We will use $\mathbf{1}_E$ to denote the indicator of the event E and $[n] = \{1, 2, \ldots, n\}$. The algorithm maintains an active set of arms A_t such that if $j \notin A_t$ then the algorithm has concluded that arm j is not the Borda winner. At each time t, the algorithm chooses an arm I_t uniformly at random from $[n]$ and compares it with all the arms in A_t. Note that $A_k \subseteq A_\ell$ for all $k \geq \ell$. Let $Z_{i,j}^{(t)} \in \{0, 1\}$ be independent Bernoulli random variables with $\mathbb{E}[Z_{i,j}^{(t)}] = p_{i,j}$, each denoting the outcome of "dueling" $i, j \in [n]$ at time t (define $Z_{i,j}^{(t)} = 0$

Algorithm 1 Sparse Borda Algorithm

Input sparsity level $k \in [n-2]$, time gate $T_0 \geq 0$
Start with active set $A_1 = \{1, 2, \cdots, n\}$, $t = 1$
Let $C_t = \sqrt{\frac{2\log(4n^2t^2/\delta)}{t/n}} + \frac{2\log(4n^2t^2/\delta)}{3t/n}$
While$|A_t| > 1$ Choose I_t uniformly at random $[n]$.
For$j \in A_t$ Observe $Z_{j,I_t}^{(t)}$ and update $\widehat{p}_{j,I_t,t} = \frac{n}{t}\sum_{\ell=1}^{t} Z_{j,I_\ell}^{(\ell)} \mathbf{1}_{I_\ell = I_t}$, $\widehat{s}_{j,t} = \frac{n/(n-1)}{t}\sum_{\ell=1}^{t} Z_{j,I_\ell}^{(\ell)}$.
$A_{t+1} = A_t \setminus \Big\{ j \in A_t : \exists i \in A_t$ with
1) $\mathbf{1}_{\{t>T_0\}}\, \widehat{\Delta}_{i,j,t}\left(\arg\max_{\Omega\subset[n]:|\Omega|=k} \widehat{\nabla}_{i,j,t}(\Omega)\right) > 6(k+1)C_t$
OR 2) $\widehat{s}_{i,t} > \widehat{s}_{j,t} + \frac{n}{n-1}\sqrt{\frac{2\log(4nt^2/\delta)}{t}} \Big\}$
$t \leftarrow t+1$

for $i = j$). For any $t \geq 1$, $i \in [n]$, and $j \in A_t$ define

$$\widehat{p}_{j,i,t} = \frac{n}{t}\sum_{\ell=1}^{t} Z_{j,I_\ell}^{(\ell)} \mathbf{1}_{I_\ell = i}$$

so that $\mathbb{E}\left[\widehat{p}_{j,i,t}\right] = p_{j,i}$. Furthermore, for any $t \geq 1$, $j \in A_t$ define

$$\widehat{s}_{j,t} = \frac{n/(n-1)}{t}\sum_{\ell=1}^{t} Z_{j,I_\ell}^{(\ell)}$$

so that $\mathbb{E}\left[\widehat{s}_{j,t}\right] = s_j$. For any $\Omega \subset [n]$ and $i, j \in [n]$ define

$$\begin{aligned}
\Delta_{i,j}(\Omega) &= 2(p_{i,j} - \tfrac{1}{2}) + \sum_{\omega \in \Omega : \omega \neq i \neq j} (p_{i,\omega} - p_{j,\omega}) \\
\widehat{\Delta}_{i,j,t}(\Omega) &= 2(\widehat{p}_{i,j,t} - \tfrac{1}{2}) + \sum_{\omega \in \Omega : \omega \neq i \neq j} (\widehat{p}_{i,\omega,t} - \widehat{p}_{j,\omega,t}) \\
\nabla_{i,j}(\Omega) &= \sum_{\omega \in \Omega : \omega \neq i \neq j} |p_{i,\omega} - p_{j,\omega}| \\
\widehat{\nabla}_{i,j}(\Omega) &= \sum_{\omega \in \Omega : \omega \neq i \neq j} |\widehat{p}_{i,\omega,t} - \widehat{p}_{j,\omega,t}| \ .
\end{aligned}$$

The quantity $\Delta_{i,j}(\Omega)$ is the *partial* gap between the Borda scores for i and j, based on only the comparisons with the arms in Ω. Note that $\frac{1}{n-1}\Delta_{i,j}([n]) = s_i - s_j$. The quantity $\arg\max_{\Omega \subset [n] : |\Omega| = k} \nabla_{i,j}(\Omega)$ selects the indices ω yielding the largest discrepancies $|p_{i,\omega} - p_{j,\omega}|$. $\widehat{\Delta}$ and $\widehat{\nabla}$ are empirical analogs of these quantities.

Definition 6.7. *For any $i \in [n] \setminus 1$ we say the set $\{(p_{1,\omega} - p_{i,\omega})\}_{\omega \neq 1 \neq i}$ is (γ, k)-approximately sparse if*

$$\max_{\Omega \in [n] : |\Omega| \leq k} \nabla_{1,i}(\Omega \setminus \Omega_i) \ \leq \ \gamma \Delta_{1,i}(\Omega_i)$$

where $\Omega_i = \arg \max_{\Omega \subset [n] : |\Omega| = k} \nabla_{1,i}(\Omega)$.

Instead of the strong assumption that the set $\{(p_{1,\omega} - p_{i,\omega})\}_{\omega \neq 1 \neq i}$ has no more than k non-zero coefficients, the above definition relaxes this idea and just assumes that the absolute value of the coefficients outside the largest k are small relative to the partial Borda gap. This definition is inspired by the structure described in previous sections and will allow us to find the Borda winner faster.

The parameter T_0 is specified (see Theorem 6.8) to guarantee that all arms with sufficiently large gaps $s_1 - s_i$ are eliminated by time step T_0 (condition 2). Once $t > T_0$, condition 1 also becomes active and the algorithm starts removing arms with large partial Borda gaps, exploiting the assumption that the top arms can be distinguished by comparisons with a sparse set of other arms. The algorithm terminates when only one arm remains.

Theorem 6.8. *Let $k \geq 0$ and $T_0 > 0$ be inputs to the above algorithm and let R be the solution to $\frac{32}{R^2} \log\left(\frac{32n/\delta}{R^2}\right) = T_0$. If for all $i \in [n] \setminus 1$, at least one of the following holds:*

1. *$\{(p_{1,\omega} - p_{i,\omega})\}_{\omega \neq 1 \neq i}$ is $(\frac{1}{3}, k)$-approximately sparse,*
2. *$(s_1 - s_i) \geq R$,*

then with probability at least $1 - 3\delta$, the algorithm returns the best arm after no more than

$$c \sum_{j>1} \min \left\{ \max \left\{ \frac{1}{R^2} \log\left(\frac{n/\delta}{R^2}\right), \frac{(k+1)^2/n}{\Delta_j^2} \log\left(\frac{n/\delta}{\Delta_j^2}\right) \right\}, \frac{1}{\Delta_j^2} \log\left(\frac{n/\delta}{\Delta_j^2}\right) \right\}$$

samples where $\Delta_j := s_1 - s_j$ and $c > 0$ is an absolute constant.

Remark 2. *In the above theorem, the second argument of the* min *is precisely the result one would obtain by running Successive Elimination with the Borda reduction [83]. Thus, under the stated assumptions, the algorithm never does worse than the Borda reduction scheme. The first argument of the* min *indicates the potential improvement gained by exploiting the sparsity assumption. The first argument of the* max *is the result of throwing out the arms with large Borda differences and the second argument is the result of throwing out arms where a* partial *Borda difference was observed to be large.*

Remark 3. *Consider the P_1 matrix discussed above, then Theorem 6.8 implies that by setting $T_0 = \frac{32}{R^2}\log\left(\frac{32n/\delta}{R^2}\right)$ with $R = \frac{1/2+\epsilon}{n-1} + \frac{1}{4}\frac{n-2}{n-1} \approx \frac{1}{4}$ and $k = 1$ we obtain a sample complexity of $O(\epsilon^{-2}n\log(n))$ for the proposed algorithm compared to the standard Borda reduction sample complexity of $\Omega(n^2)$. In practice it is difficult to optimize the choice of T_0 and k, but motivated by the results shown in the experiments section, we recommend setting $T_0 = 0$ and $k = 5$ for typical problems.*

To prove Theorem 6.8 we first need a technical lemma.

Lemma 6.9. *For all $s \in \mathbb{N}$, let I_s be drawn independently and uniformly at random from $[n]$ and let $Z_{i,j}^{(s)}$ be a Bernoulli random variable with mean $p_{i,j}$. If $\widehat{p}_{i,j,t} = \frac{n}{t}\sum_{s=1}^{t} Z_{i,j}^{(s)}\mathbf{1}_{I_s=j}$ for all $i \in [n]$ and $C_t = \sqrt{\frac{2\log(4n^2t^2/\delta)}{t/n}} + \frac{2\log(4n^2t^2/\delta)}{3t/n}$ then*

$$\mathbb{P}\left(\bigcup_{(i,j)\in[n]^2:i\neq j}\bigcup_{t=1}^{\infty}\{|\widehat{p}_{i,j,t} - p_{i,j}| > C_t\}\right) \leq \delta.$$

Proof. Note that $t\widehat{p}_{i,j,t} = \sum_{s=1}^{t} nZ_{i,j}^{(s)}\mathbf{1}_{I_s=j}$ is a sum of i.i.d. random variables taking values in $[0,n]$ with $\mathbb{E}\left[\left(nZ_{i,j}^{(s)}\mathbf{1}_{I_s=j}\right)^2\right] \leq n^2\mathbb{E}\left[\mathbf{1}_{I_s=j}\right] \leq n$. A direct application of Bernstein's inequality [105] and union bounding over all pairs $(i,j) \in [n]^2$ and time t gives the result. □

A consequence of the lemma is that by repeated application of the triangle inequality,

$$\begin{aligned}\left|\widehat{\nabla}_{i,j,t}(\Omega) - \nabla_{i,j}(\Omega)\right| &= \left|\sum_{\omega\in\Omega:\omega\neq i\neq j} |\widehat{p}_{i,\omega,t} - \widehat{p}_{j,\omega,t}| - |p_{i,\omega} - p_{j,\omega}|\right| \\ &\leq \sum_{\omega\in\Omega:\omega\neq i\neq j} |\widehat{p}_{i,\omega,t} - p_{i,\omega}| + |p_{j,\omega} - \widehat{p}_{j,\omega,t}| \\ &\leq 2|\Omega|C_t\end{aligned}$$

and similarly $\left|\widehat{\Delta}_{i,j,t}(\Omega) - \Delta_{i,j}(\Omega)\right| \leq 2(1+|\Omega|)C_t$ for all $i,j \in [n]$ with $i \neq j$, all $t \in \mathbb{N}$ and all $\Omega \subset [n]$. We are now ready to prove Theorem 6.8.

Proof. We begin the proof by defining $C_t(\Omega) = 2(1+|\Omega|)C_t$ and considering the events

$$\begin{aligned}&\bigcap_{t=1}^{\infty}\bigcap_{\Omega\subset[n]}\left\{|\widehat{\Delta}_{i,j,t}(\Omega) - \Delta_{i,j}(\Omega)| < C_t(\Omega)\right\}, \\ &\bigcap_{t=1}^{\infty}\bigcap_{\Omega\subset[n]}\left\{|\widehat{\nabla}_{i,j,t}(\Omega) - \nabla_{i,j}(\Omega)| < C_t(\Omega)\right\}, \\ &\bigcap_{t=1}^{\infty}\bigcap_{i=1}^{n}\left\{|\widehat{s}_{i,t} - s_i| < \frac{n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}}\right\},\end{aligned}$$

that each hold with probability at least $1-\delta$. The first set of events are a consequence of Lemma 6.9 and the last set of events are proved using a straightforward Hoeffding bound [105] and a union bound similar to that in Lemma 6.9. In what follows assume these events hold.

Step 1: If $t > T_0$ and $s_1 - s_j > R$, then $j \notin A_t$.

We begin by considering all those $j \in [n] \setminus 1$ such that $s_1 - s_j \geq R$ and show that with the prescribed value of T_0, these arms are thrown out before $t > T_0$. By the events

defined above, for arbitrary $i \in [n] \setminus 1$ we have

$$\begin{aligned}\widehat{s}_{i,t} - \widehat{s}_{1,t} &= \widehat{s}_{i,t} - s_i + s_1 - \widehat{s}_{1,t} + s_i - s_1 \\ &\leq s_i - s_1 + \frac{2n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}} \leq \frac{2n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}}\end{aligned}$$

since by definition $s_1 > s_i$. This proves that the best arm will never be thrown out using the Borda reduction which implies that $1 \in A_t$ for all $t \leq T_0$. On the other hand, for any $j \in [n] \setminus 1$ such that $s_1 - s_j \geq R$ and $t \leq T_0$ we have

$$\begin{aligned}\max_{i \in A_t} \widehat{s}_{i,t} - \widehat{s}_{j,t} &\geq \widehat{s}_{1,t} - \widehat{s}_{j,t} \\ &\geq s_1 - s_j - \frac{2n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}} \\ &= \frac{\Delta_{1,j}([n])}{n-1} - \frac{2n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}}\,.\end{aligned}$$

If τ_j is the first time t that the right hand side of the above is greater than or equal to $\frac{2n}{n-1}\sqrt{\frac{\log(4nt^2/\delta)}{2t}}$ then

$$\tau_j \leq \frac{32n^2}{\Delta_{1,j}^2([n])} \log\left(\frac{32n^3/\delta}{\Delta_{1,j}^2([n])}\right),$$

since for all positive a, b, t with $a/b \geq e$ we have $t \geq \frac{2\log(a/b)}{b} \implies b \geq \frac{\log(at)}{t}$. Thus, any j with $\frac{\Delta_{1,j}([n])}{n-1} = s_1 - s_j \geq R$ has $\tau_j \leq T_0$ which implies that any $i \in A_t$ for $t > T_0$ has $s_1 - s_i \leq R$.

Step 2: For all t, $1 \in A_t$.

We showed above that the Borda reduction will never remove the best arm from A_t. We

now show that the sparse-structured discard condition will not remove the best arm. At any time $t > T_0$, let $i \in [n] \setminus 1$ be arbitrary and let $\widehat{\Omega}_i = \arg \max_{\Omega \subset [n]:|\Omega|=k} \widehat{\nabla}_{i,1,t}(\Omega)$ and $\Omega_i = \arg \max_{\Omega \subset [n]:|\Omega|=k} \nabla_{i,1}(\Omega)$. Note that for any $\Omega \subset [n]$ we have $\nabla_{i,1}(\Omega) = \nabla_{1,i}(\Omega)$ but $\Delta_{i,1}(\Omega) = -\Delta_{1,i}(\Omega)$ and

$$\begin{aligned}
\widehat{\Delta}_{i,1,t}(\widehat{\Omega}_i) &\leq \Delta_{i,1}(\widehat{\Omega}_i) + C_t(\widehat{\Omega}_i) \\
&= \Delta_{i,1}(\widehat{\Omega}_i) - \Delta_{i,1}(\Omega_i) + \Delta_{i,1}(\Omega_i) + C_t(\widehat{\Omega}_i) \\
&= \left(\sum_{\omega \in \widehat{\Omega}_i} (p_{i,\omega} - p_{1,\omega}) \right) - \left(\sum_{\omega \in \Omega_i} (p_{i,\omega} - p_{1,\omega}) \right) - \Delta_{1,i}(\Omega_i) + C_t(\widehat{\Omega}_i) \\
&\leq - \left(\sum_{\omega \in \Omega_i \setminus \widehat{\Omega}_i} (p_{i,\omega} - p_{1,\omega}) \right) - \frac{2}{3} \Delta_{1,i}(\Omega_i) + C_t(\widehat{\Omega}_i)
\end{aligned}$$

since $\left(\sum_{\omega \in \widehat{\Omega}_i \setminus \Omega_i} (p_{i,\omega} - p_{1,\omega}) \right) \leq \nabla_{1,i} \left(\widehat{\Omega}_i \setminus \Omega_i \right) \leq \frac{1}{3} \Delta_{1,i}(\Omega_i)$ by the conditions of the

theorem. Continuing,

$$\begin{aligned}
\widehat{\Delta}_{i,1,t}(\widehat{\Omega}_i) &\leq -\left(\sum_{\omega\in\Omega_i\setminus\widehat{\Omega}_i}(p_{i,\omega}-p_{1,\omega})\right)-\frac{2}{3}\Delta_{1,i}(\Omega_i)+C_t(\widehat{\Omega}_i)\\
&\leq \left(\sum_{\omega\in\Omega_i\setminus\widehat{\Omega}_i}|\widehat{p}_{i,\omega,t}-\widehat{p}_{1,\omega,t}|\right)-\frac{2}{3}\Delta_{1,i}(\Omega_i)+C_t(\widehat{\Omega}_i)+C_t(\Omega_i\setminus\widehat{\Omega}_i)\\
&\leq \left(\sum_{\omega\in\widehat{\Omega}_i\setminus\Omega_i}|\widehat{p}_{i,\omega,t}-\widehat{p}_{1,\omega,t}|\right)-\frac{2}{3}\Delta_{1,i}(\Omega_i)+C_t(\widehat{\Omega}_i)+C_t(\Omega_i\setminus\widehat{\Omega}_i)\\
&\leq \left(\sum_{\omega\in\widehat{\Omega}_i\setminus\Omega_i}|p_{i,\omega}-p_{1,\omega}|\right)-\frac{2}{3}\Delta_{1,i}(\Omega_i)+C_t(\widehat{\Omega}_i)+C_t(\Omega_i\setminus\widehat{\Omega}_i)+C_t(\widehat{\Omega}_i\setminus\Omega_i)\\
&\leq -\frac{1}{3}\Delta_{1,i}(\Omega_i)+C_t(\widehat{\Omega}_i)+C_t(\Omega_i\setminus\widehat{\Omega}_i)+C_t(\widehat{\Omega}_i\setminus\Omega_i)\\
&\leq 3\max_{\Omega\subset[n]:|\Omega|\leq k}C_t(\Omega)=6(1+k)C_t
\end{aligned}$$

where the third inequality follows from the fact that $\widehat{\nabla}_{i,1,t}\left(\Omega_i\setminus\widehat{\Omega}_i\right)\leq\widehat{\nabla}_{i,1,t}\left(\widehat{\Omega}_i\setminus\Omega_i\right)$ by definition, and the second-to-last line follows again by the same theorem condition used above. Thus, combining both steps one and two, we have that $1\in A_t$ for all t.

Step 3 : Sample Complexity

At any time $t>T_0$, let $j\in[n]\setminus 1$ be arbitrary and let $\widehat{\Omega}_i=\arg\max_{\Omega\subset[n]:|\Omega|=k}\widehat{\nabla}_{1,j,t}(\Omega)$ and

$\Omega_i = \arg \max_{\Omega \subset [n]:|\Omega|=k} \nabla_{1,j}(\Omega)$. We begin with

$$\begin{aligned}
\max_{i \in [n] \setminus j} \widehat{\Delta}_{i,j,t}\left(\widehat{\Omega}_i\right) &\geq \widehat{\Delta}_{1,j,t}(\widehat{\Omega}_i) \\
&\geq \Delta_{1,j}(\widehat{\Omega}_i) - C_t(\widehat{\Omega}_i) \\
&\geq \Delta_{1,j}(\widehat{\Omega}_i) - \Delta_{1,j}(\Omega_i) + \Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) \\
&= \left(\sum_{\omega \in \widehat{\Omega}} (p_{1,\omega} - p_{j,\omega})\right) - \left(\sum_{\omega \in \Omega_i} (p_{1,\omega} - p_{j,\omega})\right) + \Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) \\
&\geq -\left(\sum_{\omega \in \Omega_i \setminus \widehat{\Omega}} (p_{i,\omega} - p_{1,\omega})\right) + \frac{2}{3}\Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) \\
&\geq -\left(\sum_{\omega \in \Omega_i \setminus \widehat{\Omega}} |\widehat{p}_{i,\omega,t} - \widehat{p}_{1,\omega,t}|\right) + \frac{2}{3}\Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) - C_t(\Omega_i \setminus \widehat{\Omega}_i) \\
&\geq -\left(\sum_{\omega \in \widehat{\Omega}_i \setminus \Omega_i} |\widehat{p}_{i,\omega,t} - \widehat{p}_{1,\omega,t}|\right) + \frac{2}{3}\Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) - C_t(\Omega_i \setminus \widehat{\Omega}_i) \\
&\geq -\left(\sum_{\omega \in \widehat{\Omega}_i \setminus \Omega_i} |p_{i,\omega} - p_{1,\omega}|\right) + \frac{2}{3}\Delta_{1,j}(\Omega_i) - C_t(\widehat{\Omega}_i) - C_t(\Omega_i \setminus \widehat{\Omega}_i) - C_t(\widehat{\Omega}_i \setminus \Omega_i) \\
&\geq \frac{1}{3}\Delta_{1,j}(\Omega_i) - 3 \max_{\Omega \subset [n]:|\Omega| \leq k} C_t(\Omega) = \frac{1}{3}\Delta_{1,j}(\Omega_i) - 6(1+k)C_t
\end{aligned}$$

by a series of steps as analogous to those in Step 2. If τ_j is the first time $t > T_0$ such that the right hand side is greater than or equal to $6(1+k)C_t$, the point at which j would be removed, we have that

$$\tau_j \leq \frac{20736n(k+1)^2}{\Delta_{1,j}^2(\Omega_i)} \log\left(\frac{20736n^2(k+1)^2}{\Delta_{1,j}^2(\Omega_i)\,\delta}\right)$$

using the same inequality as above in Step 2. Combining steps one and three we have

that the total number of samples taken is bounded by

$$\sum_{j>1} \min\left\{\max\left\{T_0, \frac{20736n(k+1)^2}{\Delta_{1,j}^2(\Omega_i)}\log\left(\frac{20736n^2(k+1)^2}{\Delta_{1,j}^2(\Omega_i)\,\delta}\right)\right\}, \frac{32n^2}{\Delta_{1,j}^2([n])}\log\left(\frac{32n^3/\delta}{\Delta_{1,j}^2([n])}\right)\right\}$$

with probability at least $1-3\delta$. The result follows from recalling that $\frac{\Delta_{1,j}(\Omega_i)}{n-1} = s_1 - s_j$ and noticing that $\frac{n}{n-1} \leq 2$ for $n \geq 2$. □

6.5 Experiments

The goal of this section is not to obtain the best possible sample complexity results for the specified datasets, but to show the *relative* performance gain of exploiting structure using the proposed SECS algorithm with respect to the Borda reduction. That is, we just want to measure the effect of exploiting sparsity while keeping all other parts of the algorithms constant. Thus, the algorithm we compare to that uses the simple Borda reduction is simply the SECS algorithm described above but with $T_0 = \infty$ so that the sparse condition never becomes activated. Running the algorithm in this way, it is very closely related to the Successive Elimination algorithm of [83]. In what follows, our proposed algorithm will be called SECS and the benchmark algorithm will be denoted as just the Borda reduction (BR) algorithm.

We experiment on both simulated data and two real-world datasets. During all experiments, both the BR and SECS algorithms were run with $\delta = 0.1$. For the SECS algorithm we set $T_0 = 0$ to enable condition 1 from the very beginning (recall for BR we set $T_0 = \infty$). Also, while the algorithm has a constant factor of 6 multiplying $(k+1)C_t$, we feel that the analysis that led to this constant is very loose so in practice

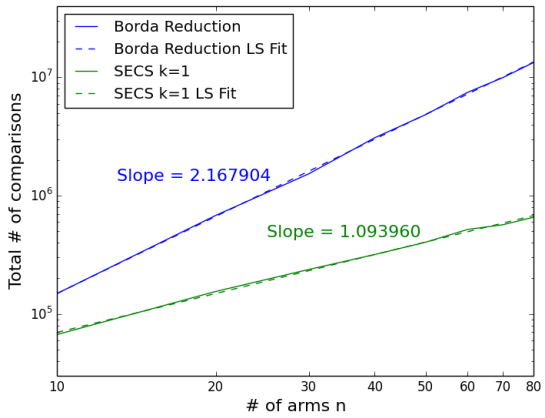


Figure 6.3: Comparison of the Borda reduction algorithm and the proposed SECS algorithm ran on the P_1 matrix for different values of n. Plot is on log-log scale so that the sample complexity grows like n^s where s is the slope of the line.

we recommend the use of a constant of $1/2$ which was used in our experiments. While the change of this constant invalidates the guarantee of Theorem 6.8, we note that in all of the experiments to be presented here, neither algorithm ever failed to return the best arm. This observation also suggests that the SECS algorithm is robust to possible inconsistencies of the model assumptions.

6.5.1 Synthetic Preference matrix

Both algorithms were tasked with finding the best arm using the P_1 matrix of (6.1) with $\epsilon = 1/5$ for problem sizes equal to $n = 10, 20, 30, 40, 50, 60, 70, 80$ arms. Inspecting the P_1 matrix, we see that a value of $k = 1$ in the SECS algorithm suffices so this is used for all problem sizes. The entries of the preference matrix $P_{i,j}$ are used to simulate comparisons between the respective arms and each experiment was repeated 75 times.

Recall from Section 6.3 that any algorithm using the Borda reduction on the P_1 matrix has a sample complexity of $\Omega(n^2)$. Moreover, inspecting the proof of Theorem 6.8

one concludes that the BR algorithm has a sample complexity of $O(n^2 \log(n))$ for the P_1 matrix. On the other hand, Theorem 6.8 states that the SECS algorithm should have a sample complexity no worse than $O(n \log(n))$ for the P_1 matrix. Figure 6.3 plots the sample complexities of SECS and BR on a log-log plot. On this scale, to match our sample complexity hypotheses, the slope of the BR line should be about 2 while the slope of the SECS line should be about 1, which is exactly what we observe.

6.5.2 Web search data

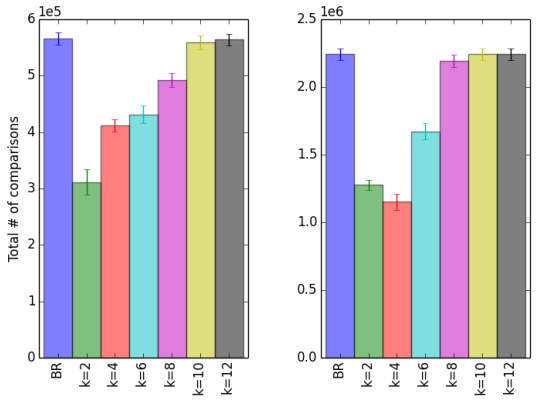
We consider two web search data sets. The first is the MSLR-WEB10k Microsoft Learning to Rank data set [103] that is characterized by approximately 30,000 search queries over a number of documents from search results. The data also contains the values of 136 features and corresponding user labelled relevance factors with respect to each query-document pair. We use the training set of Fold 1, which comprises of about 2,000 queries. The second data set is the MQ2008-list from the Microsoft Learning to Rank 4.0 (MQ2008) data set [104]. We use the training set of Fold 1, which has about 550 queries. Each query has a list of documents with 46 features and corresponding user labelled relevance factors.

For each data set, we create a set of rankers, each corresponding to a feature from the feature list. The aim of this task is be to determine the feature whose ranking of query-document pairs is the most relevant. To compare two rankers, we randomly choose a pair of documents and compare their relevance rankings with those of the features. Whenever a mismatch occurs between the rankings returned by the two features, the feature whose ranking matches that of the relevance factors of the two documents "wins

the duel". If both features rank the documents similarly, the duel is deemed to have resulted in a tie and we flip a fair coin. We run a Monte Carlo simulation on both data sets to obtain a preference matrix P corresponding to their respective feature sets. As with the previous setup, the entries of the preference matrices ($[P]_{i,j} = p_{i,j}$) are used to simulate comparisons between the respective arms and each experiment was repeated 75 times.

From the MSLR-WEB10k data set, a single arm was removed for our experiments as its Borda score was unreasonably close to the arm with the best Borda score and behaved unlike any other arm in the dataset with respect to its α_i curves, confounding our model. For these real datasets, we consider a range of different k values for the SECS algorithm. As noted above, while there is no guarantee that the SECS algorithm will return the true Borda winner, in all of our trials for all values of k reported we never observed a single error. This is remarkable as it shows that the correctness of the algorithm is insensitive to the value of k on at least these two real datasets. The sample complexities of BR and SECS on both datasets are reported in Figure 6.4. We observe that the SECS algorithm, for small values of k, can identify the Borda winner using as few as *half* the number required using the Borda reduction method. As k grows, the performance of the SECS algorithm becomes that of the BR algorithm, as predicted by Theorem 6.8.

Lastly, the preference matrices of the two data sets support the argument for finding the Borda winner over the Condorcet winner. The MSLR-WEB10k data set has no Condorcet winner arm. However, while the MQ2008 data set has a Condorcet winner, when we consider the Borda scores of the arms, it ranks second.



(a) MSLR-WEB10k (b) MQ2008

Figure 6.4: Comparison of an action elimination-style algorithm using the Borda reduction (denoted as BR) and the proposed SECS algorithm with different values of k on the two datasets.

6.6 Discussion

This chapter studied the dueling bandits best-arm identification problem using the Borda voting rule. We proved a distribution dependent lower bound for this problem that nearly matches the upper bound achieved by using the so-called Borda reduction and a standard multi-armed bandit algorithm, e.g. the lil'UCB algorithm of Chapter 4. However, we showed that there exists naturally occurring structure found in real datasets that, when assumed to be there, can be exploited by adaptive sampling to accelerate the identification of the best arm both in theory and practice. This structure is characterized in our algorithm by two parameters describing a notion of sparsity and a threshold separating easy from difficult arms. Our lower bound implies that it is impossible to be adaptive to both parameters, but perhaps these two parameters can be reduced down to a single, intuitive parameter that can be estimated for different problems in a natural way. Another future direction is coming up with a new algorithm for this setting. Chapter 4

suggests that Successive Elimination, the algorithm that the proposed algorithm in this work is based off of, may be a poor algorithm for practice. An open question is whether an algorithm like lil'UCB can be adapted to this setting.

6.7 Bibliographical Remarks

The work presented in this chapter was based on the author's publication

- Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *arXiv preprint arXiv:1502.07943*, 2015.

Chapter 7

Stochastic Derivative-Free Optimization

Up until this point, this thesis has considered a finite set of objects to rank, embed, or find the best of. In this chapter we consider an optimization problem in which the space of objects is uncountable, e.g. an object is identified by its location in $\mathbb{R}^d$ of which there are infinitely many. To motivate this setting, consider getting fit for prescription lenses. Prescriptions are specified by 6 numbers taking values in a continuum and the doctor attempts to search the space of prescriptions for some acceptable solution by asking a series of questions to the patient of the form "better or worse?" In a small number of questions of this form, the doctor optimizes a function, namely, the patient's ability to see, over these 6 continuous-valued dimensions. This chapter attempts to understand how hard this problem is and explain how one might automate such a process.

Mathematically, the objective of this chapter is to study the difficulty of minimizing a convex function $f : \mathbb{R}^d \to \mathbb{R}$ under different measurement methods. We consider two types of measurements:

- function evaluations : for any point $x \in \mathbb{R}^d$ we may observe the independent random variable $E_f(x)$ where $\mathbb{E}[E_f(x)] = f(x)$,

- function comparisons : for any points $x, y \in \mathbb{R}^d$ we may observe the independent random variable $C_f(x, y)$ where $\mathbb{P}\left(C_f(x, y) = \text{sign}(f(y) - f(x))\right) > 1/2$.

Both measurement types have previously appeared in this thesis, and it would be understandable if the reader assumed there was some tradeoff between these two: function evaluations may provided more information (e.g. more than 1-bit) but function comparisons are more convenient to use when gathering measurements from humans. This chapter questions this intuitive hypothesis and in a sense, shows it to be false. In particular, we propose an algorithm that uses noisy pairwise comparisons to minimize a convex function to within accuracy ϵ and requires no more than a constant multiple of the number of queries required by the best algorithm that uses noisy function evaluations.

7.1 Introduction

Optimizing large-scale complex systems often requires the tuning of many parameters. With training data or simulations one can evaluate the relative merit, or incurred loss, of different parameter settings, but it may be unclear how each parameter influences the overall objective function. In such cases, derivatives of the objective function with respect to the parameters are unavailable. Thus, we have seen a resurgence of interest in Derivative Free Optimization (DFO) [106, 107, 108, 109, 110, 111, 112, 113]. When function evaluations are noiseless, DFO methods can achieve the same rates of convergence as noiseless gradient methods up to a small factor depending on a low-order polynomial of the dimension [110, 114, 115]. This leads one to wonder if the same equivalence can be extended to the case when function evaluations and gradients are noisy.

Sadly, we prove otherwise. We show that when function evaluations are noisy, the

optimization error of *any* DFO is $\Omega(\sqrt{1/T})$, where T is the number of evaluations. This lower bound holds even for strongly convex functions. In contrast, noisy gradient methods exhibit $\Theta(1/T)$ error scaling for strongly convex functions [114, 116]. A consequence of our theory is that finite differencing cannot achieve the rates of gradient methods when the function evaluations are noisy.

On the positive side, we also present a new derivative-free algorithm that achieves this lower bound with near optimal dimension dependence. Moreover, the algorithm uses only boolean comparisons of function values, not actual function values. This makes the algorithm applicable to situations in which the optimization is only able to probably correctly decide if the value of one configuration is better than the value of another. This is especially interesting in optimization based on human subject feedback, where paired comparisons are often used instead of numerical scoring. The convergence rate of the new algorithm is optimal in terms of T and near-optimal in terms of its dependence on the ambient dimension. Surprisingly, our lower bounds show that this new algorithm that uses only function comparisons achieves the same rate in terms of T as any algorithm that has access to function evaluations.

7.2 Problem formulation and background

We now formalize the notation and conventions for our analysis of DFO. A function f is *strongly convex with constant* τ on a convex set $\mathcal{B} \subset \mathbb{R}^d$ if there exists a constant $\tau > 0$ such that

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\tau}{2}||x - y||^2$$

for all $x, y \in \mathcal{B}$. The gradient of f, if it exists, denoted ∇f, is *Lipschitz with constant* L if $||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$ for some $L > 0$. The class of strongly convex functions with Lipschitz gradients defined on a nonempty, convex set $\mathcal{B} \subset \mathbb{R}^d$ which take their minimum in $\mathcal{B}$ with parameters τ and L is denoted by $\mathcal{F}_{\tau,L,\mathcal{B}}$. For background on these concepts and convex optimization in general, see [52, 53].

The problem we consider is minimizing a function $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$. The function f is not explicitly known. An optimization procedure may only query the function in one of the following two ways.

Function Evaluation Oracle: For any point $x \in \mathcal{B}$ an optimization procedure can observe

$$E_f(x) = f(x) + w$$

where $w \in \mathbb{R}$ is a random variable with $\mathbb{E}[w] = 0$ and $\mathbb{E}[w^2] = \sigma^2$.

Function Comparison Oracle: For any pair of points $x, y \in \mathcal{B}$ an optimization procedure can observe a binary random variable $C_f(x, y)$ satisfying

$$\mathbb{P}\left(C_f(x,y) = \text{sign}\{f(y) - f(x)\}\right) \geq \frac{1}{2} + \min\left\{\delta_0, \mu|f(y) - f(x)|^{\kappa-1}\right\} \tag{7.1}$$

for some $0 < \delta_0 \leq 1/2$, $\mu > 0$ and $\kappa \geq 1$. When $\kappa = 1$, without loss of generality assume $\mu \leq \delta_0 \leq 1/2$. Note $\kappa = 1$ implies that the comparison oracle is correct with a probability that is greater than $1/2$ and independent of x, y. If $\kappa > 1$, then the oracle's reliability decreases as the difference between $f(x)$ and $f(y)$ decreases.

To illustrate how the function comparison oracle and function evaluation oracles

relate to each other, suppose $C_f(x,y) = \text{sign}\{E_f(y) - E_f(x)\}$ where $E_f(x)$ is a function evaluation oracle with additive noise w. If w is Gaussian distributed with mean zero and variance σ^2 then $\kappa = 2$ and $\mu \geq (4\pi\sigma^2 e)^{-1/2}$ (see Appendix C.1). In fact, this choice of w corresponds to Thurston's law of comparative judgment which is a popular model for outcomes of pairwise comparisons from human subjects [117]. If w is a "spikier" distribution such as a two-sided Gamma distribution with shape parameter in the range of $(0,1]$ then all values of $\kappa \in (1,2]$ can be realized (see Appendix C.1).

Interest in the function comparison oracle is motivated by certain popular derivative-free optimization procedures that use only comparisons of function evaluations (e.g. [112]) and by optimization problems involving human subjects making paired comparisons (for instance, getting fitted for prescription lenses or a hearing aid where unknown parameters specific to each person are tuned with the familiar queries "better or worse?"). Pairwise comparisons have also been suggested as a novel way to tune web-search algorithms [118, 119]. Pairwise comparison strategies have previously been analyzed in the finite setting where the task is to identify the best alternative among a finite set of alternatives (sometimes referred to as the dueling-bandit problem) [51, 118]. A similar pairwise comparison oracle in the continuous domain has also been considered previously and we compare to these results below [119]. The function comparison oracle presented in this work and its analysis are novel. The main contributions of this work and new art are as follows (i) lower bounds for the function evaluation oracle in the presence of measurement noise (ii) lower bounds for the function comparison oracle in the presence of noise and (iii) an algorithm for the function comparison oracle, which can also be applied to the function evaluation oracle setting, that nearly matches both the lower bounds of (i) and (ii).

We prove our lower bounds for strongly convex functions with Lipschitz gradients defined on a compact, convex set $\mathcal{B}$, and because these problems are a subset of those involving all convex functions (and have non-empty intersection with problems where f is merely Lipschitz), the lower bound also applies to these larger classes. While there are known theoretical results for DFO in the noiseless setting [110, 115, 120], to the best of our knowledge we are the first to characterize lower bounds for DFO in the stochastic setting. Moreover, we believe we are the first to show a near-optimal upper bound for stochastic DFO using a function comparison oracle, which also applies to the function evaluation oracle (the work of [119] predates our results but they achieve suboptimal rates). However, there are algorithms with upper bounds on the rates of convergence for stochastic DFO with the function evaluation oracle [120, 121]. We discuss the relevant results in the next section following the lower bounds .

While there remains many open problems in stochastic DFO, rates of convergence with a stochastic gradient oracle are well known and were first lower bounded by Nemirovski and Yudin [120]. These classic results were recently tightened to show a dependence on the dimension of the problem [122]. And then tightened again to show a better dependence on the noise [116] which matches the upper bound achieved by stochastic gradient descent [114]. The aim of this work is to start filling in the knowledge gaps of stochastic DFO so that it is as well understood as the stochastic gradient oracle. Our bounds are based on simple techniques borrowed from the statistical learning literature that use natural functions and oracles in the same spirit of [116].

7.3 Main results

The results below are presented with simplifying constants that encompass many factors to aid in exposition. Explicit constants are given in the proofs in Sections 7.4 and 7.5. Throughout, we denote the minimizer of f as x_f^*. The expectation in the bounds is with respect to the noise in the oracle queries and (possible) optimization algorithm randomization.

7.3.1 Query complexity of the function comparison oracle

Theorem 7.1. *For some $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$ let C_f be a function comparison oracle with parameters (κ, μ, δ_0). Then for $d \geq 8$ and sufficiently large T*

$$\inf_{\widehat{x}_T} \sup_{f \in \mathcal{F}_{\tau,L,\mathcal{B}}} \mathbb{E}\left[f(\widehat{x}_T) - f(x_f^*)\right] \geq \begin{cases} c_1 \exp\left\{-c_2 \frac{T}{d}\right\} & \text{if } \kappa = 1 \\ c_3 \left(\frac{d}{T}\right)^{\frac{1}{2(\kappa-1)}} & \text{if } \kappa > 1 \end{cases}$$

where the infimum is over the collection of all possible estimators of x_f^* using at most T queries to a function comparison oracle and the supremum is taken with respect to all problems in $\mathcal{F}_{\tau,L,\mathcal{B}}$ and function comparison oracles with parameters (κ, μ, δ_0). The constants c_1, c_2, c_3 depend the oracle and function class parameters, as well as the geometry of $\mathcal{B}$, but are independent of T and d.

For upper bounds we propose a specific algorithm based on coordinate-descent in Section 7.5 and prove the following theorem for the case of unconstrained optimization, that is, $\mathcal{B} = \mathbb{R}^d$.

Theorem 7.2. *For some $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$ with $\mathcal{B} = \mathbb{R}^d$ let C_f be a function comparison oracle with parameters (κ, μ, δ_0). Then there exists a coordinate-descent algorithm that is adaptive to unknown $\kappa \geq 1$ that outputs an estimate $\widehat{x}_T$ after T function comparison queries such that with probability $1 - \delta$*

$$\sup_{f \in \mathcal{F}_{\tau,L,B}} \mathbb{E}\left[f(\widehat{x}_T) - f(x_f^*)\right] \leq \begin{cases} c_1 \exp\left\{-c_2\sqrt{\frac{T}{d}}\right\} & \text{if } \kappa = 1 \\ c_3 d \left(\frac{d}{T}\right)^{\frac{1}{2(\kappa-1)}} & \text{if } \kappa > 1 \end{cases}$$

where c_1, c_2, c_3 depend the oracle and function class parameters as well as T,d, and $1/\delta$, but only poly-logarithmically.

7.3.2 Query complexity of the function evaluation oracle

Theorem 7.3. *For some $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$ let E_f be a function evaluation oracle with variance σ^2. Then for $d \geq 8$ and sufficiently large T*

$$\inf_{\widehat{x}_T} \sup_{f \in \mathcal{F}_{\tau,L,\mathcal{B}}} \mathbb{E}\left[f(\widehat{x}_T) - f(x_f^*)\right] \geq c\left(\frac{d\sigma^2}{T}\right)^{\frac{1}{2}}$$

where the infimum is taken with respect to the collection of all possible estimators of x_f^* using just T queries to a function evaluation oracle and the supremum is taken with respect to all problems in $\mathcal{F}_{\tau,L,\mathcal{B}}$ and function evaluation oracles with variance σ^2. The constant c depends on the oracle and function class parameters, as well as the geometry of $\mathcal{B}$, but is independent of T and d.

Because a function evaluation oracle can always be turned into a function comparison oracle (see discussion above), the algorithm and upper bound in Theorem 2 with $\kappa = 2$

applies to many typical function evaluation oracles (e.g. additive Gaussian noise), yielding an upper bound of $(d^3\sigma^2/T)^{1/2}$ ignoring constants and log factors. This matches the rate of convergence as a function of T and σ^2, but has worse dependence on the dimension d.

Alternatively, under a less restrictive setting (i.e. not strongly convex), Nemirovski and Yudin proposed two algorithms for the class of convex, Lipschitz functions that obtain rates of $d^{1/2}/T^{1/4}$ and $p(d)/T^{1/2}$, respectively, where $p(d)$ was left as an unspecified polynomial of d [120]. Yue and Joachims in [119] built off the work of Flaxman [123] and showed that a pairwise comparison oracle can achieve the same $d^{1/2}/T^{1/4}$ rate achieved by function evaluations. While focusing on stochastic DFO with bandit feedback, Agarwal *et. al.* built on the ideas developed in [120] to obtain a result that implies a convergence rate of $d^{16}/T^{1/2}$ [121]. Whether or not these rates can be improved under the more restrictive function classes we consider is an open question.

A related but fundamentally different problem that is somewhat related with the setting considered in this paper is described as online (or stochastic) convex optimization with multi-point feedback [110, 124, 125]. Essentially, this setting allows the algorithm to probe the value of the function f plus noise at multiple locations where the noise changes at each time step, but each set of samples at each time experiences the *same* noise. Because the noise model of that work is incompatible with the one considered here, no comparisons should be made between the two.

7.4 Lower Bounds

The lower bounds in Theorems 1 and 3 are proved using a general minimax bound [126, Thm. 2.5]. Our proofs are most related to the approach developed in [127] for active

learning, which like optimization involves a Markovian sampling process. Roughly speaking, the lower bounds are established by considering a simple case of the optimization problem in which the global minimum is known a priori to belong to a finite set. Since the simple case is "easier" than the original optimization, the minimum number of queries required for a desired level of accuracy in this case yields a lower bound for the original problem.

The following theorem is used to prove the bounds. In the terms of the theorem, f is a function to be minimized and P_f is the probability model governing the noise associated with queries when f is the true function.

Theorem 7.4. *[126, Thm. 2.5] Consider a class of functions $\mathcal{F}$ and an associated family of probability measures $\{P_f\}_{f\in\mathcal{F}}$. Let $M \geq 2$ be an integer and $f_0, f_1, \ldots, f_M$ be functions in $\mathcal{F}$. Let $d(\cdot,\cdot) : \mathcal{F}\times\mathcal{F} \to \mathbb{R}$ be a semi-distance and assume that:*

1. *$d(f_i, f_j) \geq 2s > 0$, for all $0 \leq i < j \leq M$,*
2. *$\frac{1}{M}\sum_{j=1}^{M} KL(P_i||P_0) \leq a \log M$,*

where the Kullback-Leibler divergence $KL(P_i||P_0) := \int \log \frac{dP_i}{dP_0} dP_i$ is assumed to be well-defined (i.e., P_0 is a dominating measure) and $0 < a < 1/8$. Then

$$\inf_{\widehat{f}} \sup_{f\in\mathcal{F}} \mathbb{P}(d(\widehat{f}, f) \geq s) \;\geq\; \inf_{\widehat{f}} \max_{f\in\{f_0,\ldots,f_M\}} \mathbb{P}(d(\widehat{f}, f) \geq s) \;\geq\; \frac{\sqrt{M}}{1+\sqrt{M}}\left(1 - 2a - 2\sqrt{\frac{a}{\log M}}\right) > 0\,,$$

where the infimum is taken over all possible estimators based on a sample from P_f.

We are concerned with the functions in the class $\mathcal{F} := \mathcal{F}_{\tau,L,\mathcal{B}}$. The volume of $\mathcal{B}$ will affect only constant factors in our bounds, so we will simply denote the class of functions by $\mathcal{F}$ and refer explicitly to $\mathcal{B}$ only when necessary. Let $x_f := \arg\min_x f(x)$, for all

$f \in \mathcal{F}$. The semi-distance we use is $d(f,g) := \|x_f - x_g\|$, for all $f, g \in \mathcal{F}$. Note that each point in $\mathcal{B}$ can be specified by one of many $f \in \mathcal{F}$. So the problem of selecting an f is equivalent to selecting a point $x \in \mathcal{B}$. Indeed, the semi-distance defines a collection of equivalence classes in $\mathcal{F}$ (i.e., all functions having a minimum at $x \in \mathcal{B}$ are equivalent). For every $f \in \mathcal{F}$ we have $\inf_{g\in\mathcal{F}} f(x_g) = \inf_{x\in\mathcal{B}} f(x)$, which is a useful identity to keep in mind.

We now construct the functions $f_0, f_1, \ldots, f_M$ that will be used for our proofs. Let $\Omega = \{-1, 1\}^d$ so that each $\omega \in \Omega$ is a vertex of the d-dimensional hypercube. Let $\mathcal{V} \subset \Omega$ with cardinality $|\mathcal{V}| \geq 2^{d/8}$ such that for all $\omega \neq \omega' \in \mathcal{V}$, we have $\rho(\omega, \omega') \geq d/8$ where $\rho(\cdot,\cdot)$ is the Hamming distance. It is known that such a set exists by the Varshamov-Gilbert bound [126, Lemma 2.9]. Denote the elements of $\mathcal{V}$ by $\omega_0, \omega_1, \ldots, \omega_M$. Next we state some elementary bounds on the functions that will be used in our analysis.

Lemma 7.5. *For $\epsilon > 0$ define the set $\mathcal{B} \subset \mathbb{R}^d$ to be the ℓ_∞ ball of radius ϵ and define the functions on $\mathcal{B}$: $f_i(x) := \frac{\tau}{2}||x - \epsilon\omega_i||^2$, for $i = 0, \ldots, M$, $\omega_i \in \mathcal{V}$, and $x_i := \arg\min_x f_i(x) = \epsilon\omega_i$. Then for all $0 \leq i < j \leq M$ and $x \in \mathcal{B}$ the functions $f_i(x)$ satisfy*

1. *f_i is strongly convex-τ with Lipschitz-L gradients and $x_i \in \mathcal{B}$*
2. *$||x_i - x_j|| \geq \epsilon\sqrt{\frac{d}{2}}$*
3. *$|f_i(x) - f_j(x)| \leq 2\tau d\epsilon^2$.*

We are now ready to prove Theorems 1 and 3. Each proof uses the functions $f_0, \ldots, f_M$ a bit differently, and since the noise model is also different in each case, the KL divergence is bounded differently in each proof. We use the fact that if X and Y are random variables

distributed according to Bernoulli distributions P_X and P_Y with parameters $1/2+\mu$ and $1/2-\mu$, then $\mathrm{KL}(P_X||P_Y) \leq 4\mu^2/(1/2-\mu)$. Also, if $X \sim \mathcal{N}(\mu_X, \sigma^2) =: P_X$ and $Y \sim \mathcal{N}(\mu_Y, \sigma^2) =: P_y$ then $\mathrm{KL}(P_X||P_Y) = \frac{1}{2\sigma^2}||\mu_X - \mu_Y||^2$.

7.4.1 Proof of Theorem 7.1

First we will obtain the bound for the case $\kappa > 1$. Let the comparison oracle satisfy

$$\mathbb{P}\left(C_{f_i}(x,y) = \mathrm{sign}\{f_i(y) - f_i(x)\}\right) \;=\; \frac{1}{2} + \min\left\{\mu|f_i(y)-f_i(x)|^{\kappa-1}, \delta_0\right\}.$$

In words, $C_{f_i}(x,y)$ is correct with probability as large as the right-hand-side of above and is monotonic increasing in $f_i(y)-f_i(x)$. Let $\{x_k, y_k\}_{k=1}^T$ be a sequence of T pairs in $\mathcal{B}$ and let $\{C_{f_i}(x_k,y_k)\}_{k=1}^T$ be the corresponding sequence of noisy comparisons. We allow the sequence $\{x_k,y_k\}_{k=1}^T$ to be generated in any way subject to the Markovian assumption that $C_{f_i}(x_k,y_k)$ given (x_k,y_k) is conditionally independent of $\{x_i,y_i\}_{i<k}$. For $i=0,\ldots,M$, and $\ell = 1,\ldots,T$ let $P_{i,\ell}$ denote the joint probability distribution of $\{x_k, y_k, C_{f_i}(x_k,y_k)\}_{k=1}^{\ell}$, let $Q_{i,\ell}$ denote the conditional distribution of $C_{f_i}(x_\ell,y_\ell)$ given (x_ℓ,y_ℓ), and let S_ℓ denote the conditional distribution of (x_ℓ,y_ℓ) given $\{x_k,y_k,C_{f_i}(x_k,y_k)\}_{k=1}^{\ell-1}$. Note that S_ℓ is only a function of the underlying optimization algorithm and does not depend on i.

$$\begin{aligned}\mathrm{KL}(P_{i,T}||P_{j,T}) &= \mathbb{E}_{P_{i,T}}\left[\log\frac{P_{i,T}}{P_{j,T}}\right] = \mathbb{E}_{P_{i,T}}\left[\log\frac{\prod_{\ell=1}^T Q_{i,\ell}S_\ell}{\prod_{\ell=1}^T Q_{j,\ell}S_\ell}\right] = \mathbb{E}_{P_{i,T}}\left[\log\frac{\prod_{\ell=1}^T Q_{i,\ell}}{\prod_{\ell=1}^T Q_{j,\ell}}\right] \\ &= \sum_{\ell=1}^T \mathbb{E}_{P_{i,T}}\left[\mathbb{E}_{P_{i,T}}\left[\log\frac{Q_{i,\ell}}{Q_{j,\ell}}\middle|\{x_k,y_k\}_{k=1}^T\right]\right] \leq T\sup_{x_1,y_1\in\mathcal{B}}\mathbb{E}_{P_{i,1}}\left[\mathbb{E}_{P_{i,1}}\left[\log\frac{Q_{i,1}}{Q_{j,1}}\middle|x_1,y_1\right]\right]\end{aligned}$$

By the second claim of Lemma 7.5, $|f_i(x)-f_j(x)| \leq 2\tau d\epsilon^2$, and therefore the bound above is less than or equal to the KL divergence between the Bernoulli distributions with

parameters $\frac{1}{2} \pm \mu \left(2\tau d\epsilon^2\right)^{(\kappa-1)}$, yielding the bound

$$\mathrm{KL}(P_{i,T}|P_{j,T}) \leq \frac{4T\mu^2 \left(2\tau d\epsilon^2\right)^{2(\kappa-1)}}{1/2 - \mu \left(2\tau d\epsilon^2\right)^{(\kappa-1)}} \leq 16T\mu^2 \left(2\tau d\epsilon^2\right)^{2(\kappa-1)}$$

provided ϵ is sufficiently small. We also assume ϵ (or, equivalently, $\mathcal{B}$) is sufficiently small so that $|f_i(x) - f_j(x)|^{\kappa-1} \leq \delta_0$. We are now ready to apply Theorem 7.4. Recalling that $M \geq 2^{d/8}$, we want to choose ϵ such that

$$\mathrm{KL}(P_{i,T}|P_{j,T}) \leq 16T\mu^2 \left(2\tau d\epsilon^2\right)^{2(\kappa-1)} \leq a\frac{d}{8}\log(2) \leq a \log M$$

with an a small enough so that we can apply the theorem. By setting $a = 1/16$ and equating the two sides of the equation we have $\epsilon = \epsilon_T := \frac{1}{2\sqrt{d}} \left(\frac{2}{\tau}\right)^{1/2} \left(\frac{d\log(2)}{2048\mu^2 T}\right)^{\frac{1}{4(\kappa-1)}}$ (note that this also implies a sequence of sets $\mathcal{B}_T$ by the definition of the functions in Lemma 7.5). Thus, the semi-distance satisfies

$$d(f_j, f_i) = ||x_j - x_i|| \geq \sqrt{d/2}\epsilon_T \geq \frac{1}{2\sqrt{2}} \left(\frac{2}{\tau}\right)^{1/2} \left(\frac{d\log(2)}{2048\mu^2 T}\right)^{\frac{1}{4(\kappa-1)}} =: 2s_T \ .$$

Applying Theorem 7.4 we have

$$\begin{aligned}\inf_{\widehat{f}} \sup_{f\in\mathcal{F}} \mathbb{P}(\|x_{\widehat{f}} - x_f\| \geq s_T) \geq \inf_{\widehat{f}} \max_{i\in\{0,\ldots,M\}} \mathbb{P}(\|x_{\widehat{f}} - x_i\| \geq s_T) = \inf_{\widehat{f}} \max_{i\in\{0,\ldots,M\}} \mathbb{P}(d(\widehat{f}, f_i) \geq s_T) \\ \geq \frac{\sqrt{M}}{1+\sqrt{M}} \left(1 - 2a - 2\sqrt{\frac{a}{\log M}}\right) > 1/7 \,,\end{aligned}$$

where the final inequality holds since $M \geq 2$ and $a = 1/16$. Strong convexity implies that $f(x) - f(x_f) \geq \frac{\tau}{2}||x - x_f||^2$ for all $f \in \mathcal{F}$ and $x \in \mathcal{B}$. Therefore

$$\begin{aligned}\inf_{\widehat{f}} \sup_{f\in\mathcal{F}} \mathbb{P}\left(f(x_{\widehat{f}}) - f(x_f) \geq \frac{\tau}{2}s_T^2\right) &\geq \inf_{\widehat{f}} \max_{i\in\{0,\dots,M\}} \mathbb{P}\left(f_i(x_{\widehat{f}}) - f_i(x_i) \geq \frac{\tau}{2}s_T^2\right) \\ &\geq \inf_{\widehat{f}} \max_{i\in\{0,\dots,M\}} \mathbb{P}\left(\frac{\tau}{2}\|x_{\widehat{f}} - x_i\|^2 \geq \frac{\tau}{2}s_T^2\right) \\ &= \inf_{\widehat{f}} \max_{i\in\{0,\dots,M\}} \mathbb{P}\left(\|x_{\widehat{f}} - x_i\| \geq s_T\right) > 1/7\,.\end{aligned}$$

Finally, applying Markov's inequality we have

$$\inf_{\widehat{f}} \sup_{f\in\mathcal{F}} \mathbb{E}\left[f(x_{\widehat{f}}) - f(x_f)\right] \geq \frac{1}{7}\left(\frac{1}{32}\right)\left(\frac{d\log(2)}{2048\mu^2 T}.\right)^{\frac{1}{2(\kappa-1)}}$$

7.4.2 Proof of Theorem 7.1 for $\kappa = 1$

To handle the case when $\kappa = 1$ we use functions of the same form, but the construction is slightly different. Let ℓ be a positive integer and let $M = \ell^d$. Let $\{\xi_i\}_{i=1}^M$ be a set of uniformly space points in $\mathcal{B}$ which we define to be the unit cube in $\mathbb{R}^d$, so that $\|\xi_i - \xi_j\| \geq \ell^{-1}$ for all $i \neq j$. Define $f_i(x) := \frac{\tau}{2}||x - \xi_i||^2$, $i = 1, \dots, M$. Let $s := \frac{1}{2\ell}$ so that $d(f_i, f_j) := ||x_i^* - x_j^*|| \geq 2s$. Because $\kappa = 1$, we have $\mathbb{P}\left(C_{f_i}(x, y) = \text{sign}\{f_i(y) - f_i(x)\}\right) \geq \mu$ for some $\mu > 0$, all $i \in \{1, \dots, M\}$, and all $x, y \in \mathcal{B}$. We bound $\text{KL}(P_{i,T}||P_{j,T})$ in exactly the same way as we bounded it in Section 7.4.1 except that now we have $C_{f_i}(x_k, y_k) \sim \text{Bernoulli}(\frac{1}{2} + \mu)$ and $C_{f_j}(x_k, y_k) \sim \text{Bernoulli}(\frac{1}{2} - \mu)$. It then follows that if we wish to apply the theorem, we want to choose s so that

$$\text{KL}(P_{i,T}|P_{j,T}) \leq 2T\mu^2/(1/2 - \mu) \leq a\log M = ad\log\left(\tfrac{1}{2s}\right)$$

for some $a < 1/8$. Using the same sequence of steps as in Section 7.4.1 we have

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}} \mathbb{E}\left[f(x_{\widehat{f}}) - f(x_f)\right] \geq \frac{1}{7}\frac{\tau}{2}\left(\frac{1}{2}\right)^2 \exp\left\{-\frac{128T\mu^2}{d(1/2-\mu)}\right\}.$$

7.4.3 Proof of Theorem 7.3

Let f_i for all $i = 0, \ldots, M$ be the functions considered in Lemma 7.5. Recall that the evaluation oracle is defined to be $E_f(x) := f(x) + w$, where w is a random variable (independent of all other random variables under consideration) with $\mathbb{E}[w] = 0$ and $\mathbb{E}[w^2] = \sigma^2 > 0$. Let $\{x_k\}_{k=1}^T$ be a sequence of points in $\mathcal{B} \subset \mathbb{R}^d$ and let $\{E_f(x_k)\}_{k=1}^T$ denote the corresponding sequence of noisy evaluations of $f \in \mathcal{F}$. For $\ell = 1, \ldots, T$ let $P_{i,\ell}$ denote the joint probability distribution of $\{x_k, E_{f_i}(x_k)\}_{k=1}^{\ell}$, let $Q_{i,\ell}$ denote the conditional distribution of $E_{f_i}(x_k)$ given x_k, and let S_ℓ denote the conditional distribution of x_ℓ given $\{x_k, E_f(x_k)\}_{k=1}^{\ell-1}$. S_ℓ is a function of the underlying optimization algorithm and does not depend on i. We can now bound the KL divergence between any two hypotheses as in Section 7.4.1:

$$\mathrm{KL}(P_{i,T}||P_{j,T}) \leq T \sup_{x_1 \in \mathcal{B}} \mathbb{E}_{P_{i,1}}\left[\mathbb{E}_{P_{i,1}}\left[\log \frac{Q_{i,1}}{Q_{j,1}}\middle| x_1\right]\right].$$

To compute a bound, let us assume that w is Gaussian distributed. Then

$$\begin{aligned}\mathrm{KL}(P_{i,T}||P_{j,T}) &\leq T \sup_{z\in\mathcal{B}} \mathrm{KL}\left(\mathcal{N}(f_i(z),\sigma^2)||\mathcal{N}(f_j(z),\sigma^2)\right) \\ &= \frac{T}{2\sigma^2}\sup_{z\in\mathcal{B}}|f_i(z)-f_j(z)|^2 \leq \frac{T}{2\sigma^2}\left(2\tau d\epsilon^2\right)^2\end{aligned}$$

by the third claim of Lemma 7.5. We then repeat the same procedure as in Section 7.4.1 to attain

$$\inf_{\widehat{f}}\sup_{f\in\mathcal{F}}\mathbb{E}\left[f(x_{\widehat{f}})-f(x_f)\right] \geq \frac{1}{7}\left(\frac{1}{32}\right)\left(\frac{d\sigma^2\log(2)}{64T}\right)^{\frac{1}{2}}.$$

7.5 Upper bounds

The algorithm that achieves the upper bound using a pairwise comparison oracle is a combination of a few standard techniques and methods pulled from the convex optimization and statistical learning literature. The algorithm can be summarized as follows. At each iteration the algorithm picks a coordinate uniformly at random from the n possible dimensions and then performs an approximate line search. By exploiting the fact that the function is strongly convex with Lipschitz gradients, one guarantees using standard arguments that the approximate line search makes a sufficient decrease in the objective function value in expectation [52, Ch.9.3]. If the pairwise comparison oracle made no errors then the approximate line search is accomplished by a binary-search-like scheme

that is known in the literature as the golden section line-search algorithm [128]. However, when responses from the oracle are only probably correct we make the line-search robust to errors by repeating the same query until we can be confident about the true, uncorrupted direction of the pairwise comparison using a standard procedure from the active learning literature [129].

7.5.1 Coordinate descent algorithm

n-dimensional Pairwise comparison algorithm

Input: $x_0 \in \mathbb{R}^d$, $\eta \geq 0$

For k=0,1,2,...

Choose $v_k = \mathbf{e}_i$ for $i \in \{1, \ldots, d\}$ chosen uniformly at random

Obtain α_k from a line-search such that

$|\alpha_k - \alpha^*| \leq \eta$ where $\alpha^* = \arg\min_\alpha f(x_k + \alpha\, v_k)$

$x_{k+1} = x_k + \alpha_k\, v_k$

end

Figure 7.1: Algorithm to minimize a convex function in d dimensions. Here $\mathbf{e}_i$ is understood to be a vector of all zeros with a one in the ith position.

Theorem 7.6. *Let $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$ with $\mathcal{B} = \mathbb{R}^d$. For any $\eta > 0$ assume the line search in the algorithm of Figure 7.1 requires at most $T_\ell(\eta)$ queries from the pairwise comparison oracle. If x_K is an estimate of $x^* = \arg\min_x f(x)$ after requesting no more than K pairwise comparisons, then*

$$\sup_f \mathbb{E}[f(x_K) - f(x_*)] \leq \frac{4dL^2\eta^2}{\tau} \qquad \textit{whenever} \qquad K \geq \frac{4dL}{\tau} \log\left(\frac{f(x_0) - f(x^*)}{\eta^2 2dL^2/\tau}\right) T_\ell(\eta)$$

where the expectation is with respect to the random choice of v_k at each iteration.

Proof. First note that $||v_k|| = 1$ for all k with probability 1. Because the gradients of f

are Lipschitz (L) we have from Taylor's theorem

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), \alpha_k v_k \rangle + \frac{\alpha_k^2 L}{2}.$$

Note that the right-hand-side is convex in α_k and is minimized by

$$\widehat{\alpha}_k = -\frac{\langle \nabla f(x_k), v_k \rangle}{L}.$$

However, recalling how α_k is chosen, if $\alpha^* = \arg\min_\alpha f(x_k + \alpha\, v_k)$ then we have

$$f(x_k + \alpha_k\, v_k) - f(x_k + \alpha^*\, v_k) \leq \frac{L}{2}||(\alpha_k - \alpha^*)v_k||^2 = \frac{L}{2}|\alpha_k - \alpha^*|^2 \leq \frac{L}{2}\eta^2.$$

This implies

$$\begin{aligned} f(x_k + \alpha_k v_k) - f(x_k) &\leq f(x_k + \alpha^* v_k) - f(x_k) + \frac{L}{2}\eta^2 \\ &\leq f(x_k + \hat{\alpha}_k v_k) - f(x_k) + \frac{L}{2}\eta^2 \\ &\leq -\frac{\langle \nabla f(x_k), v_k \rangle^2}{2L} + \frac{L}{2}\eta^2. \end{aligned}$$

Taking the expectation with respect to v_k, we have

$$\begin{aligned} \mathbb{E}\left[f(x_{k+1})\right] &\leq \mathbb{E}\left[f(x_k)\right] - \mathbb{E}\left[\frac{\langle \nabla f(x_k), v_k \rangle^2}{2L}\right] + \frac{L}{2}\eta^2 \\ &= \mathbb{E}\left[f(x_k)\right] - \mathbb{E}\left[\mathbb{E}\left[\frac{\langle \nabla f(x_k), v_k \rangle^2}{2L}\middle| v_0, \ldots, v_{k-1}\right]\right] + \frac{L}{2}\eta^2 \\ &= \mathbb{E}\left[f(x_k)\right] - \mathbb{E}\left[\frac{||\nabla f(x_k)||^2}{2dL}\right] + \frac{L}{2}\eta^2 \end{aligned}$$

where we applied the law of iterated expectation. Let $x^* = \arg\min_x f(x)$ and note that x^* is a unique minimizer by strong convexity (τ). Using the previous calculation we have

$$\mathbb{E}\left[f(x_{k+1}) - f(x^*)\right] - \tfrac{L}{2}\eta^2 \leq \mathbb{E}\left[f(x_k) - f(x^*)\right] - \tfrac{\mathbb{E}\left[||\nabla f(x_k)||^2\right]}{2dL} \leq \mathbb{E}\left[f(x_k) - f(x^*)\right]\left(1 - \tfrac{\tau}{4dL}\right)$$

where the second inequality follows from

$$\begin{aligned}(f(x_k) - f(x^*))^2 \leq& \left(\langle \nabla f(x_k), x_k - x^* \rangle\right)^2 \\ \leq& ||\nabla f(x_k)||^2 ||x_k - x^*||^2 \leq ||\nabla f(x_k)||^2 \left(\frac{\tau}{2}\right)^{-1} \left(f(x_k) - f(x^*)\right).\end{aligned}$$

If we define $\rho_k := \mathbb{E}\left[f(x_k) - f(x^*)\right]$ then we equivalently have

$$\rho_{k+1} - \frac{2dL^2\eta^2}{\tau} \leq \left(1 - \frac{\tau}{4dL}\right)\left(\rho_k - \frac{2dL^2\eta^2}{\tau}\right) \leq \left(1 - \frac{\tau}{4dL}\right)^k \left(\rho_0 - \frac{2dL^2\eta^2}{\tau}\right)$$

which completes the proof. □

This implies that if we wish $\sup_f \mathbb{E}[f(x_K) - f(x_*)] \leq \epsilon$ it suffices to take $\eta = \sqrt{\frac{\epsilon\tau}{4dL^2}}$ so that at most $\frac{4dL}{\tau} \log\left(\frac{f(x_0) - f(x^*)}{\epsilon/2}\right) T_\ell\left(\sqrt{\frac{\epsilon\tau}{4dL^2}}\right)$ pairwise comparisons are requested.

7.5.2 Line search

This section is concerned with minimizing a function $f(x_k + \alpha\, v_k)$ over some $\alpha \in \mathbb{R}$. Because we are minimizing over a single variable, α, we will restart the indexing at 0 such that the line search algorithm produces a sequence $\alpha_0, \alpha_1, \ldots, \alpha_{K'}$. This indexing should not be confused with the indexing of the iterates $x_1, x_2, \ldots, x_K$. We will first present an algorithm that assumes the pairwise comparison oracle makes no errors and

then extend the algorithm to account for the noise model introduced in Section 7.2.

Consider the algorithm of Figure 7.2. At each iteration, one is guaranteed to eliminate at least 1/2 the search space at each iteration such that at least 1/4 the search space is discarded for every pairwise comparison that is requested. However, with a slight modification to the algorithm, one can guarantee a greater fraction of removal (see the golden section line-search algorithm). We use this sub-optimal version for simplicity because it will help provide intuition for how the robust version of the algorithm works.

One Dimensional Pairwise comparison algorithm

Input: $x \in \mathbb{R}^d$, $v \in \mathbb{R}^d$, $\eta > 0$
Initialize: $\alpha_0 = 0$, $\alpha_0^+ = \alpha_0 + 1$, $\alpha_0^- = \alpha_0 - 1$, $k = 0$
If $C_f(x, x + \alpha_0^+ v) > 0$ and $C_f(x, x + \alpha_0^- v) < 0$
  $\alpha_0^+ = 0$
end
If $C_f(x, x + \alpha_0^- v) > 0$ and $C_f(x, x + \alpha_0^+ v) < 0$
  $\alpha_0^- = 0$
end
While $C_f(x, x + \alpha_k^+ v) < 0$
  $\alpha_{k+1}^+ = 2\alpha_k^+$, $k = k + 1$
end
While $C_f(x, x + \alpha_k^- v) < 0$
  $\alpha_{k+1}^- = 2\alpha_k^-$, $k = k + 1$
end
$\alpha_k = \frac{1}{2}(\alpha_k^- + \alpha_k^+)$
While $|\alpha_k^+ - \alpha_k^-| \geq \eta/2$
  if $C_f(x + \alpha_k v, x + \frac{1}{2}(\alpha_k + \alpha_k^+) v) < 0$
    $\alpha_{k+1} = \frac{1}{2}(\alpha_k + \alpha_k^+)$, $\alpha_{k+1}^+ = \alpha_k^+$, $\alpha_{k+1}^- = \alpha_k$
  else if $C_f(x + \alpha_k v, x + \frac{1}{2}(\alpha_k + \alpha_k^-) v) < 0$
    $\alpha_{k+1} = \frac{1}{2}(\alpha_k + \alpha_k^-)$, $\alpha_{k+1}^+ = \alpha_k$, $\alpha_{k+1}^- = \alpha_k^-$
  else
    $\alpha_{k+1} = \alpha_k$, $\alpha_{k+1}^+ = \frac{1}{2}(\alpha_k + \alpha_k^+)$, $\alpha_{k+1}^- = \frac{1}{2}(\alpha_k + \alpha_k^-)$
  end
end
Output: α_k

Figure 7.2: Algorithm to minimize a convex function in one dimension.

Theorem 7.7. *Let $f \in \mathcal{F}_{\tau,L,\mathcal{B}}$ with $\mathcal{B} = \mathbb{R}^d$ and let C_f be a function comparison oracle that makes no errors. Let $x \in \mathbb{R}^d$ be an initial position and let $v \in \mathbb{R}^d$ be a search direction with $||v|| = 1$. If α_K is an estimate of $\alpha^* = \arg\min_\alpha f(x + \alpha\, v)$ that is output from the algorithm of Figure 7.2 after requesting no more than K pairwise comparisons, then for any $\eta > 0$*

$$|\alpha_K - \alpha^*| \leq \eta \qquad \textit{whenever} \qquad K \geq 2\log_2\left(\frac{256L\left(f(x) - f(x+\alpha^* v)\right)}{\tau^2\eta^2}\right).$$

Proof. First note that if α_K is output from the algorithm, we have $\frac{1}{2}|\alpha_K - \alpha^*| \leq |\alpha_K^+ - \alpha_K^-| \leq \frac{1}{2}\eta$, as desired.

We will handle the cases when $|\alpha^*|$ is greater than one and less than one separately. First assume that $|\alpha^*| \geq 1$. Using the fact that f is strongly convex (τ), it is straightforward to show that immediately after exiting the initial while loops, (i) at most $2 + \frac{1}{2}\log_2\left(\frac{8}{\tau}\left(f(x) - f(x+\alpha^* v)\right)\right)$ pairwise comparisons were requested, (ii) $\alpha_* \in [\alpha_k^-,\, \alpha_k^+]$, and (iii) $|\alpha_k^+ - \alpha_k^-| \leq \left(\frac{8}{\tau}\left(f(x) - f(x+\alpha^* v)\right)\right)^{1/2}$. We also have that $\alpha_* \in [\alpha_{k+1}^-,\, \alpha_{k+1}^+]$ if $\alpha_* \in [\alpha_k^-,\, \alpha_k^+]$ for all k. Thus, it follows that

$$|\alpha_{k+l}^+ - \alpha_{k+l}^-| = 2^{-l}|\alpha_k^+ - \alpha_k^-| \leq 2^{-l}\left(\frac{8}{\tau}\left(f(x) - f(x+\alpha^* v)\right)\right)^{1/2}.$$

To make the right-hand-side less than or equal to $\eta/2$, set $l = \log_2\left(\frac{\left(\frac{8}{\tau}(f(x)-f(x+\alpha^* v))\right)^{1/2}}{\eta/2}\right)$. This brings the total number of pairwise comparison requests to no more than $2\log_2\left(\frac{32(f(x)-f(x+\alpha^* v))}{\tau\eta}\right)$.

Now assume that $|\alpha^*| \leq 1$. A straightforward calculation shows that the while loops will terminate after requesting at most $2 + \frac{1}{2}\log_2\left(\frac{L}{\tau}\right)$ pairwise comparisons. And

immediately after exiting the while loops we have $|\alpha_k^+ - \alpha_k^-| \leq 2$. It follows by the same arguments of above that if we want $|\alpha_{k+l}^+ - \alpha_{k+l}^-| \leq \eta/2$ it suffices to set $l = \log_2\left(\frac{4}{\eta}\right)$. This brings the total number of pairwise comparison requests to no more than $2\log_2\left(\frac{8L}{\tau\eta}\right)$. For sufficiently small η both cases are positive and the result follows from adding the two. □

This implies that if the function comparison oracle makes no errors and it is given an iterate x_k and direction d_k then $T_\ell\left(\sqrt{\frac{\epsilon\tau}{4dL^2}}\right) \leq 2\log_2\left(\frac{2048dL^2(f(x_k)-f(x_k+\alpha^* v_k))}{\tau^3\epsilon}\right)$ which brings the total number of pairwise comparisons requested to at most $\frac{8dL}{\tau}\log\left(\frac{f(x_0)-f(x^*)}{\epsilon/2}\right)\log_2\left(\frac{2048dL^2\max_k(f(x_k)-f(x_k+\alpha^* v_k))}{\tau^3\epsilon}\right)$.

7.5.3 Proof of Theorem 7.2

We now introduce a line search algorithm that is robust to a function comparison oracle that makes errors. Essentially, the algorithm consists of nothing more than repeatedly querying the same random pairwise comparison. This strategy applied to active learning is well known because of its simplicity and its ability to adapt to unknown noise conditions [129]. However, we mention that when used in this way, this sampling procedure is known to be sub-optimal so in practice, one may want to implement a more efficient approach like that of [127]. Consider the subroutine of Figure 7.3.

Lemma 7.8. *[129] For any $x, y \in \mathbb{R}^d$ with $\mathbb{P}\left(C_f(x,y) = sign\{f(y) - f(x)\}\right) = p$, then with probability at least $1-\delta$ the algorithm of Figure 7.3 correctly identifies the sign of $\mathbb{E}\left[C_f(x,y)\right]$ and requests no more than*

$$\frac{\log(2/\delta)}{4|1/2-p|^2}\log_2\left(\frac{\log(2/\delta)}{4|1/2-p|^2}\right)$$

Repeated querying subroutine
Input: $x, y \in \mathbb{R}^d$, $\delta > 0$
Initialize: $S = \emptyset$, $l = -1$
do
$l = l + 1$
$\Delta_l = \sqrt{\frac{(l+1)\log(2/\delta)}{2^l}}$
$S = S \cup \{2^l \text{ i.i.d. draws of } C_f(x, y)\}$
while $\left|\frac{1}{2}\sum_{e_i \in S} e_i\right| - \Delta_l < 0$
return sign $\left\{\sum_{e_i \in S} e_i\right\}$.

Figure 7.3: Subroutine that estimates $\mathbb{E}[C_f(x, y)]$ by repeatedly querying the random variable.

pairwise comparisons.

We mention that that Lemma 7.8 is an inferior result compared to Lemma 4.3 but it more than suffices for our purposes here. It would be convenient if we could simply apply the result of Lemma 7.8 to the algorithm of Figure 7.2. Unfortunately, if we do this there is no guarantee that $|f(y) - f(x)|$ is bounded below so for the case when $\kappa > 1$, it would be impossible to lower bound $|1/2 - p|$ in the lemma. To account for this, we will sample at four points per iteration as opposed to just two in the noiseless algorithm to ensure that we can always lower bound $|1/2 - p|$. We will see that the algorithm and analysis naturally adapts to when $\kappa = 1$ or $\kappa > 1$.

Consider the following modification to the algorithm of Figure 7.2. We discuss the sampling process that takes place in $[\alpha_k, \alpha_k^+]$ but it is understood that the same process is repeated symmetrically in $[\alpha_k^-, \alpha_k]$. We begin with the first two `while` loops. Instead of repeatedly sampling $C_f(x, x + \alpha_k^+ v)$ we will have two sampling procedures running in parallel that repeatedly compare α_k to α_k^+ and α_k to $2\alpha_k^+$. As soon as the repeated sampling procedure terminates for one of them we terminate the second sampling strategy

and proceed with what the noiseless algorithm would do with α_k^+ assigned to be the sampling location that finished first. Once we're out of the initial `while` loops, instead of comparing α_k to $\frac{1}{2}(\alpha_k + \alpha_k^+)$ repeatedly, we will repeatedly compare α_k to $\frac{1}{3}(\alpha_k + \alpha_k^+)$ and α_k to $\frac{2}{3}(\alpha_k + \alpha_k^+)$. Again, we will treat the location that finishes its sampling first as $\frac{1}{2}(\alpha_k + \alpha_k^+)$ in the noiseless algorithm.

If we perform this procedure every iteration, then at each iteration we are guaranteed to remove at least 1/3 the search space, as opposed to 1/2 in the noiseless case, so we realize that the number of iterations of the robust algorithm is within a constant factor of the number of iterations of the noiseless algorithm. However, unlike the noiseless case where at most two pairwise comparisons were requested at each iteration, we must now apply Lemma 7.8 to determine the number of pairwise comparisons that are requested per iteration.

Intuitively, the repeated sampling procedure requests the most pairwise comparisons when the distance between the two function evaluations being compared smallest. This corresponds to when the distance between probe points is smallest, i.e. when $\eta/2 \leq |\alpha_k - \alpha^*| \leq \eta$. By considering this worst case, we can bound the number of pairwise comparisons that are requested at any iteration. By strong convexity (τ) we find through a straightforward calculation that

$$\max\left\{|f(x+\alpha_k\, v) - f(x+\frac{2}{3}(\alpha_k+\alpha_k^+)\, v)|, |f(x+\alpha_k\, v) - f(x+\frac{1}{3}(\alpha_k+\alpha_k^+)\, v)|\right\} \geq \frac{\tau}{18}\eta^2$$

for all k. This implies $|1/2 - p| \geq \mu\left(\frac{\tau}{18}\eta^2\right)^{\kappa-1}$ so that on on any given call to the repeated querying subroutine, with probability at least $1-\delta$ the subroutine requests no more than $\widetilde{O}\left(\frac{\log(1/\delta)}{(\tau\eta^2)^{2(\kappa-1)}}\right)$ pairwise comparisons. However, because we want the total number of calls

to the subroutine to hold with probability $1-\delta$, not just one, we must union bound over 4 pairwise comparisons per iteration times the number of iterations per line search times the number of line searches. This brings the total number of calls to the repeated query subroutine to no more than $4 \times \frac{3}{2} \log_2 \left(\frac{256L \max_k \left(f(x_k) - f(x_k + \alpha_k^* v_k) \right)}{\tau^2 \eta^2} \right) \times \frac{4dL}{\tau} \log \left(\frac{f(x_0) - f(x^*)}{\eta^2 2dL^2/\tau} \right) = O\left(d\frac{L}{\tau} \log^2 \left(\frac{f(x_0) - f(x^*)}{d\eta^2} \right) \right)$. If we set $\eta = \left(\frac{\epsilon\tau}{4dL^2} \right)^{1/2}$ so that $\mathbb{E}\left[f(x_K) - f(x^*) \right] \leq \epsilon$ by Theorem 7.6, then the total number of requested pairwise comparisons does not exceed

$$\widetilde{O}\left(\frac{dL}{\tau} \left(\frac{d}{\epsilon} \right)^{2(\kappa-1)} \log^2 \left(\frac{f(x_0) - f(x^*)}{\epsilon} \right) \log(d/\delta) \right).$$

By finding a $T > 0$ that satisfies this bound for any ϵ we see that this is equivalent to a rate of $O\left(d \log(d/\delta) \left(\frac{d}{T} \right)^{\frac{1}{2(\kappa-1)}} \right)$ for $\kappa > 1$ and $O\left(\exp\left\{ -c\sqrt{\frac{T}{d \log(d/\delta)}} \right\} \right)$ for $\kappa = 1$, ignoring polylog factors.

7.6 Discussion

This paper presented lower bounds on the performance of derivative-free optimization for (i) an oracle that provides noisy function evaluations and (ii) an oracle that provides probably correct boolean comparisons between function evaluations. Our results were proven for the class of strongly convex functions but because this class is a subset of all, possibly non-convex functions, our lower bounds hold for much larger classes as well. Under both oracle models we showed that the expected error decays like $\Omega\left((d/T)^{1/2} \right)$. Furthermore, for the class of strongly convex functions with Lipschitz gradients, we proposed an algorithm that achieves a rate of $\widetilde{O}\left(d(d/T)^{1/2} \right)$ for both oracle models which shows that the lower bounds are tight with respect to the dependence on the number of

iterations T and no more than a factor of d off in terms of the dimension.

7.7 Bibliographical Remarks

The work presented in this chapter was based on the author's publication

- Kevin G. Jamieson, Robert D Nowak, and Ben Recht. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2012.

Following the publication of this work, the lower bound of $\sqrt{\frac{d}{T}}$ proved in this chapter was improved to $\sqrt{\frac{d^2}{T}}$ which is believed to be tight for strongly convex functions [130].

Bibliography

[1] Hidalgo CA Salesses P, Schechtner K. The collaborative image of the city: Mapping the inequality of urban perception. *PLoS ONE 8(7): e68400*, 2013.

[2] N. Stewart, G.D.A. Brown, and N. Chater. Absolute identification by relative judgment. *Psychological Review*, 112(4):881–911, 2005.

[3] T.H.A. Bijmolt and M. Wedel. The effects of alternative methods of collecting similarity data for multidimensional scaling. *International Journal of Research in Marketing*, 12(4):363–371, 1995.

[4] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[5] Nir Ailon. Active learning ranking from pairwise preferences with almost optimal query complexity. In *Advances in Neural Information Processing Systems*, pages 810–818, 2011.

[6] Kevin G Jamieson and Robert D Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2240–2248, 2011.

[7] Kevin G Jamieson and Robert D Nowak. Active ranking in practice: General ranking functions with sample complexity bounds. In *NIPS Workshop*, 2011.

[8] Kevin G Jamieson and Robert D Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Communication, Control, and Computing*

(Allerton), 2011 49th Annual Allerton Conference on, pages 1077–1084. IEEE, 2011.

[9] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014.

[10] Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–6. IEEE, 2014.

[11] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sebastien Bubeck. On finding the largest mean among many. *Signals, Systems and Computers (ASILOMAR)*, 2013.

[12] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *arXiv preprint arXiv:1502.07943*, 2015.

[13] Kevin Jamieson, Sumeet Katariya, Atul Deshpande, and Robert Nowak. Sparse dueling bandits. In *AISTATS*, 2015.

[14] Kevin G. Jamieson, Robert D Nowak, and Ben Recht. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2012.

[15] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1998.

[16] Scott Philips, James Pitton, and Les Atlas. Perceptual feature identification for active sonar echoes. In *OCEANS 2006*, 2006.

[17] B. McFee and G. Lanckriet. Partial order embedding with multiple kernels. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 721–728. ACM, 2009.

[18] I. Gormley and T. Murphy. A latent space model for rank data. *Statistical Network Analysis: Models, Issues, and New Directions*, pages 90–102, 2007.

[19] M.A.A. Cox and T.F. Cox. Multidimensional scaling. *Handbook of data visualization*, pages 315–347, 2008.

[20] J.F. Traub. *Information-based complexity*. John Wiley and Sons Ltd., 2003.

[21] C.H. Coombs. A theory of data. *Psychological review*, 67(3):143–159, 1960.

[22] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, 14(3):326–334, 1965.

[23] S. Dasgupta, A.T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. *The Journal of Machine Learning Research*, 10:281–299, 2009.

[24] S. Hanneke. *Theoretical foundations of active learning*. PhD thesis, Citeseer, 2009.

[25] Tibor Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proceedings of the eighth annual conference on Computational learning theory*, COLT '95, pages 108–117, New York, NY, USA, 1995. ACM.

[26] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4:933–969, 2003.

[27] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[28] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294. ACM, 2007.

[29] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 97–102. IET, 1999.

[30] T. Lu and C. Boutilier. Robust approximation and incremental elicitation in voting protocols. *IJCAI-11, Barcelona*, 2011.

[31] W. Chu and Z. Ghahramani. Extensions of gaussian processes for ranking: semi-supervised and active learning. *Learning to Rank*, page 29, 2005.

[32] Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle Tseng. Active learning for ranking through expected loss optimization. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2010.

[33] Jacob Abernethy, Theodoros Evgeniou, Olivier Toubia, and J-P Vert. Eliciting consumer preferences using robust adaptive choice questionnaires. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):145–155, 2008.

[34] J.F. Bennett and W.L. Hays. Multidimensional unfolding: Determining the dimensionality of ranked preference data. *Psychometrika*, 25(1):27–43, 1960.

[35] J.I. Marden. *Analyzing and modeling rank data*. Chapman & Hall/CRC, 1995.

[36] P. Diaconis and R.L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977.

[37] Similarity Learning. Aural Sonar dataset. [http://idl.ee.washington.edu/SimilarityLearning]. University of Washington Information Design Lab, 2011.

[38] R.D. Nowak. The geometry of generalized binary search. *Arxiv preprint arXiv:0910.4397*, 2009.

[39] Nir Ailon, Ron Begleiter, and Esther Ezra. Active learning using smooth relative regret approximations with applications. *The Journal of Machine Learning Research*, 15(1):885–920, 2014.

[40] Dominique Tschopp, Suhas Diggavi, Payam Delgosha, and Soheil Mohajer. Randomized algorithms for comparison-based search. In *Advances in Neural Information Processing Systems 24*, pages 2231–2239. 2011.

[41] I. Borg and P.J.F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, 2005.

[42] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized non-metric multidimensional scaling. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2007.

[43] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A.T. Kalai. Adaptively learning the crowd kernel. *Arxiv preprint arXiv:1105.1033*, 2011.

[44] B. McFee. Distance metric learning from pairwise proximities.

[45] R.M. Johnson. Pairwise nonmetric multidimensional scaling. *Psychometrika*, 38(1):11–18, 1973.

[46] R.N. Shepard. Metric structures in ordinal data. *Journal of Mathematical Psychology*, 3(2):287–315, 1966.

[47] Nathan Srebro, Noga Alon, and Tommi S Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Advances In Neural Information Processing Systems*, pages 1321–1328, 2004.

[48] Hugh E Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, pages 167–178, 1968.

[49] V. De Silva and J.B. Tenenbaum. Sparse multidimensional scaling using landmark points. *Dept. Math., Stanford University, Stanford, CA, Tech. Rep*, 2004.

[50] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[51] K. Jamieson and R. Nowak. Active ranking using pairwise comparisons. *Neural Information Processing Systems (NIPS)*, 2011.

[52] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

[53] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

[54] R. Bissett and B. Schneider. Spatial and conjoint models based on pairwise comparisons of dissimilarities and combined effects: Complete and incomplete designs. *Psychometrika*, 56(4):685–698, 1991.

[55] Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, et al. Best arm identification: A unified approach to fixed budget and fixed confidence. 2012.

[56] Edward Paulson. A sequential procedure for selecting the population with the largest mean from k normal populations. *The Annals of Mathematical Statistics*, 35(1):174–180, 1964.

[57] Robert E Bechhofer. A sequential multiple-decision procedure for selecting the best one of several normal populations with a common unknown variance, and its use with various experimental designs. *Biometrics*, 14(3):408–429, 1958.

[58] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *Computational Learning Theory*, pages 255–270. Springer, 2002.

[59] Shie Mannor and John N Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *The Journal of Machine Learning Research*, 5:623–648, 2004.

[60] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. Pac

subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 655–662, 2012.

[61] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[62] R. H. Farrell. Asymptotic behavior of expected sample size in certain one sided tests. *The Annals of Mathematical Statistics*, 35(1):pp. 36–72, 1964.

[63] DA Darling and Herbert Robbins. Iterated logarithm inequalities. In *Herbert Robbins Selected Papers*, pages 254–258. Springer, 1985.

[64] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[65] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. *COLT 2010-Proceedings*, 2010.

[66] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory (ALT)*, 2009.

[67] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best arm identification in multi-armed bandit models. *arXiv preprint arXiv:1407.4443*, 2014.

[68] Akshay Balsubramani. Sharp uniform martingale concentration bounds. *arXiv preprint arXiv:1405.2639*, 2014.

[69] Emilie Kaufmann and Shivaram Kalyanakrishnan. Information complexity in bandit subset selection. COLT, 2013.

[70] Yasin Abbasi-Yadkori, Csaba Szepesvári, and David Tax. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[71] Sébastien Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. *arXiv preprint arXiv:1205.3181*, 2012.

[72] Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[73] Jasper Snoek, Kevin Swersky, Richard Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

[74] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. pages 507–523, 2011.

[75] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. *NIPS*, 2011.

[76] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, 2012.

[77] Alekh Agarwal, Peter Bartlett, and John Duchi. Oracle inequalities for computationally adaptive model selection. *arXiv preprint arXiv:1208.0129*, 2012.

[78] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[79] Evan R Sparks, Ameet Talwalkar, Michael J. Franklin, Michael I. Jordan, and Tim Kraska. TuPAQ: An efficient planner for large-scale predictive analytic queries. *arXiv preprint arXiv:1502.00068*, 2015.

[80] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, pages 23–37. Springer, 2009.

[81] Vincent A Cicirello and Stephen F Smith. The max k-armed bandit: A new model of exploration applied to search heuristic selection. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1355. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[82] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[83] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *The Journal of Machine Learning Research*, 7:1079–1105, 2006.

[84] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, pages 423–439, 2014.

[85] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The

nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[86] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[87] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[88] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[89] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.

[90] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[91] Tim Kraska, Ameet Talwalkar, John Duchi, Rean Griffith, Michael Franklin, and Michael Jordan. MLbase: A Distributed Machine-learning System. In *CIDR*, 2013.

[92] M. Lichman. UCI machine learning repository, 2013.

[93] Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[94] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 287–293. AAAI Press, 2011.

[95] All our ideas. http://allourideas.org/. Accessed: 2015-04-3.

[96] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[97] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb : An optimal exploration algorithm for multi-armed bandits. COLT, 2014.

[98] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 241–248, 2011.

[99] Masrour Zoghi, Shimon Whiteson, Remi Munos, and Maarten de Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. *arXiv preprint arXiv:1312.3393*, 2013.

[100] Tanguy Urvoy, Fabrice Clerot, Raphael Féraud, and Sami Naamane. Generic exploration and k-armed voting bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 91–99, 2013.

[101] Nir Ailon, Thorsten Joachims, and Zohar Karnin. Reducing dueling bandits to cardinal bandits. *arXiv preprint arXiv:1405.3396*, 2014.

[102] Shie Mannor and John N Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *The Journal of Machine Learning Research*, 5:623–648, 2004.

[103] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[104] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *CoRR*, abs/1306.2597, 2013.

[105] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013.

[106] T. Eitrich and B. Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of computational and applied mathematics*, 196(2):425–436, 2006.

[107] R. Oeuvray and M. Bierlaire. A new derivative-free algorithm for the medical image registration problem. *International Journal of Modelling and Simulation*, 27(2):115–124, 2007.

[108] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to derivative-free optimization*, volume 8. Society for Industrial Mathematics, 2009.

[109] Warren B. Powell and Ilya O. Ryzhov. *Optimal Learning*. John Wiley and Sons, 2012.

[110] Y. Nesterov. Random gradient-free minimization of convex functions. *CORE Discussion Papers*, 2011.

[111] N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *Arxiv preprint arXiv:0912.3995*, 2009.

[112] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[113] A. Agarwal, D.P. Foster, D. Hsu, S.M. Kakade, and A. Rakhlin. Stochastic convex optimization with bandit feedback. *Arxiv preprint arXiv:1107.1744*, 2011.

[114] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574, 2009.

[115] V. Protasov. Algorithms for approximate calculation of the minimum of a convex function from its values. *Mathematical Notes*, 59:69–74, 1996. 10.1007/BF02312467.

[116] M. Raginsky and A. Rakhlin. Information-based complexity, feedback, and dynamics in convex programming. *Information Theory, IEEE Transactions on*, (99):1–1, 2011.

[117] L.L. Thurstone. A law of comparative judgment. *Psychological Review; Psychological Review*, 34(4):273, 1927.

[118] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 2012.

[119] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *International Conference on Machine Learning (ICML)*, 2009.

[120] A.S. Nemirovsky and D.B. Yudin. Problem complexity and method efficiency in optimization. 1983.

[121] A. Agarwal, D.P. Foster, D. Hsu, S.M. Kakade, and A. Rakhlin. Stochastic convex optimization with bandit feedback. *Arxiv preprint arXiv:1107.1744*, 2011.

[122] A. Agarwal, P.L. Bartlett, P. Ravikumar, and M.J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *Information Theory, IEEE Transactions on*, (99):1–1, 2010.

[123] A.D. Flaxman, A.T. Kalai, and H.B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics, 2005.

[124] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *Conference on Learning Theory (COLT)*, 2010.

[125] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. 2012.

[126] A.B. Tsybakov. *Introduction to nonparametric estimation*. Springer Verlag, 2009.

[127] R.M. Castro and R.D. Nowak. Minimax bounds for active learning. *Information Theory, IEEE Transactions on*, 54(5):2339–2353, 2008.

[128] R.P. Brent. *Algorithms for minimization without derivatives*. Dover Pubns, 2002.

[129] M. Kääriäinen. Active learning in the non-realizable case. In *Algorithmic Learning Theory*, pages 63–77. Springer, 2006.

[130] Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory*, pages 3–24, 2013.

Appendix A

Chapter 2 Supplementary Materials

A.1 Computational complexity and implementation

The computational complexity of the algorithm in Figure 2.1 is determined by the complexity of testing whether a query is ambiguous or not and how many times we make this test. As written in Figure 2.1, the test would be performed $O(n^2)$ times. But if binary sort is used instead of the brute-force linear search this can be reduced to $n \log_2 n$ and, in fact, this is implemented in our simulations and the proofs of the main results. The complexity of each test is polynomial in the number of queries requested because each one is a linear constraint. Because our results show that no more than $O(d \log n)$ queries are requested, the overall complexity is no greater than $O(n \text{ poly}(d) \text{ poly}(\log n))$.

A.2 Proof of Corollary 2.4

Proof. For initial conditions given in Lemma 2.3, if $d \ll n-1$ a simple manipulation of (2.3) shows

$$\begin{aligned}
Q(n,d) &= 1 + \sum_{i=1}^{n-1} (n-i) Q(n-i, d-1) \\
&= 1 + \sum_{i=1}^{n-1} i\, Q(i, d-1) \\
&= 1 + \sum_{i=1}^{n-1} i \left[1 + \sum_{j=1}^{i-1} j\, Q(j, d-2) \right] \\
&= 1 + \Theta(n^2/2) + \sum_{i=1}^{n-1} \sum_{j=1}^{i-1} i\, j \left[1 + \sum_{k=1}^{j-1} k\, Q(k, d-3) \right] \\
&= 1 + \Theta(n^2/2) + \Theta(n^4/2/4) + \sum_{i=1}^{n-1} \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} i\, j\, k \left[1 + \sum_{l=1}^{k-1} l\, Q(l, d-4) \right] \\
&= 1 + \Theta(n^2/2) + \cdots + \Theta\left(\frac{n^{2d}}{2^d d!} \right).
\end{aligned}$$

From simulations, this is very tight for large values of n. If $d \geq n-1$ then $Q(n,d) = n!$ because any permutation of n objects can be embedded in $n-1$ dimensional space [21]. □

A.3 Construction of a d-cell with $n-1$ sides

Situations may arise in which $\Omega(n)$ queries must be requested to identify a ranking because the d-cell representing the ranking is bounded by $n-1$ hyperplanes (queries) and if they are not all requested, the ranking is ambiguous. We now show how to construct this pathological situation in $\mathbb{R}^2$. Let Θ be a collection of n points in $\mathbb{R}^2$ where each

$\theta \in \Theta$ satisfies $\theta_1^2 = \theta_2$ and $\theta_1 \in [0, 1]$ where θ_i denotes the ith dimension of θ ($i \in \{1, 2\}$). Then there exists a 2-cell in the hyperplane arrangement induced by the queries that has $n - 1$ sides. This follows because the slope of the parabola keeps increasing with θ_1 making at least one query associated with $(n - 1)$ θ's bisect the lower-left, unbounded 2-cell. This can be observed in Figure A.1. Obviously, a similar arrangement could be constructed for all $d \geq 2$.

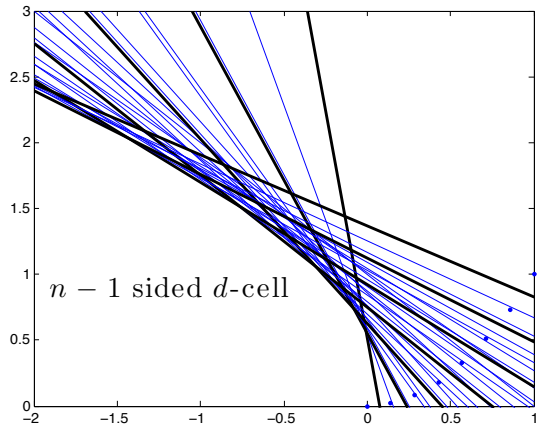


Figure A.1: The points Θ representing the objects are dots on the right, the lines are the queries, and the black, bold lines are the queries bounding the $n - 1$ sided 2-cell.

A.4 Proof of Lemma 2.10

Proof. Here we prove an upper bound on $P(k, d)$. $P(k, d)$ is equal to the number of d-cells in the partition induced by objects $1, \ldots, k$ that are intersected by a hyperplane corresponding to a pairwise comparison query between object $k + 1$ and object i, $i \in \{1, \ldots, k\}$. This new hyperplane is intersected by all the $\binom{k}{2}$ hyperplanes in the partition. These intersections partition the new hyperplane into a number of $(d - 1)$-cells. Because the $(k+1)$st object is in general position with respect to objects $1, \ldots, k$, the intersecting

hyperplanes will not intersect the hyperplane in any special or non-general way. That is to say, the number of $(d-1)$-cells this hyperplane is partitioned into is the same number that would occur if the hyperplane were intersected by $\binom{k}{2}$ hyperplanes in general position. Let $K = \binom{k}{2}$ for ease of notation. It follows then from [22, Theorem 3] that

$$\begin{aligned} P(k,d) = \sum_{i=0}^{d-1} \binom{K}{i} \leq \sum_{i=0}^{d-1} \frac{K^i}{i!} \leq \sum_{i=0}^{d-1} \frac{k^{2i}}{2^i i!} &= \frac{k^{2(d-1)}}{2^{d-1}(d-1)!} \left(1 + \sum_{i=1}^{d-1} \frac{(d-1)!}{(d-1-i)!} \left(\frac{2}{k^2}\right)^i\right) \\ &\leq \frac{k^{2(d-1)}}{2^{d-1}(d-1)!} \left(1 + \sum_{i=1}^{d-1} \left(\frac{2(d-1)}{k^2}\right)^i\right) \\ &= \frac{k^{2(d-1)}}{2^{d-1}(d-1)!} \left(\frac{1 - (2(d-1)/k^2)^d}{1 - 2(d-1)/k^2}\right). \end{aligned}$$

Thus, $\frac{2(d-1)}{k^2} \leq \epsilon < 1$ implies $P(k,d) < \frac{k^{2(d-1)}}{2^{d-1}(d-1)!} \frac{1}{1-\epsilon}$. □

Appendix B

Chapter 4 Supplementary Materials

B.1 Inverting expressions of the form $\log(\log(t))/t$

Lemma B.1. *for all positive a, b, t with $a/b \geq e$ we have $t \geq \frac{2\log(a/b)}{b} \implies b \geq \frac{\log(at)}{t}$.*

Proof Sketch. It can be shown that $\frac{\log(at)}{t}$ is monotonically decreasing for $t \geq \frac{2\log(a/b)}{b}$. It then suffices to show that $b \geq \frac{\log(at_0)}{t_0}$ for $t_0 = \frac{2\log(a/b)}{b}$ which is true whenever $a/b \geq e$. □

Lemma B.2. *Let $c > 0$, $t \geq 1$, $\varepsilon \in (0,1)$, and $\omega \in (0,1)$. Then*

$$\frac{1}{t}\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right) \geq c \Rightarrow t \leq \frac{1}{c}\log\left(\frac{2\log((1+\varepsilon)/(c\omega))}{\omega}\right). \tag{B.1}$$

Proof. It suffices to show set $c_0 = \frac{1}{t}\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)$ and show $\frac{1}{c_0}\log\left(\frac{2\log((1+\varepsilon)/(c_0\omega))}{\omega}\right) \geq t$. We begin with

$$\begin{aligned}\frac{1}{c_0}\log\left(\frac{2\log\left(\frac{1+\varepsilon}{c_0\omega}\right)}{\omega}\right) &= t\frac{\log\left(\frac{2\log((1+\varepsilon)t)-2\log(\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\omega)}{\omega}\right)}{\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)} \\ &= t\frac{\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)+\log\left(2-2\frac{\log(\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\omega)}{\log((1+\varepsilon)t)}\right)}{\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)}.\end{aligned}$$

The right hand side is greater than or equal to one if and only if the second term in the

numerator is greater than or equal to 0. And

$$\begin{aligned}\log\left(2-2\frac{\log(\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\omega)}{\log\left((1+\varepsilon)t\right)}\right)\geq 0 &\iff 1-2\frac{\log(\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\omega)}{\log\left((1+\varepsilon)t\right)}\geq 0\\ &\iff \sqrt{(1+\varepsilon)t}\geq\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\omega\\ &\Leftarrow \sqrt{y}\geq\log\left(\frac{\log(y)}{\omega}\right)\omega\quad\forall y>0.\end{aligned}$$

Note that $\omega\in(0,1)$ and $\sup_{\omega\in(0,1)}\omega\log\left(\frac{1}{\omega}\right)=e^{-1}$ so that

$$\begin{aligned}\log\left(\frac{\log(y)}{\omega}\right)\omega\leq&\log\left(\log(y)\right)+\log\left(\frac{1}{\omega}\right)\omega\\ \leq&\log\left(\log(y)\right)+e^{-1}<\sqrt{y}\end{aligned}$$

where the last inequality follows from noting that $\log\left(\log(y)\right)-\sqrt{y}+e^{-1}$ takes its maximum at $\widetilde{y}$ such that $2=\sqrt{\widetilde{y}}\log(\widetilde{y})$, which implies $2<\widetilde{y}<e$ which implies the result as $e^{-1}<1<\sqrt{2}$. □

Lemma B.3. *Let $c\in(0,1]$, $t\geq 1$, $s\geq 3$, $\varepsilon\in(0,1)$, and $\delta\in(0,e^{-e})$, $\omega\in(0,\delta]$. Then*

$$\frac{1}{t}\log\left(\frac{\log((1+\varepsilon)t)}{\omega}\right)\geq\frac{c}{s}\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right)\ \textit{and}\ \omega\leq\delta\Rightarrow t\leq\frac{s}{c}\frac{\log\left(2\log\left(\frac{1}{c\omega}\right)/\omega\right)}{\log(1/\delta)}.\tag{B.2}$$

Proof. We now use (B.1) with $c_0 = \frac{c}{s}\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right)$ to find that

$$
\begin{aligned}
t \le \frac{1}{c_0}\log\left(\frac{2\log\left(\frac{1+\varepsilon}{c_0\omega}\right)}{\omega}\right) &= \frac{s}{c}\frac{\log\left(\frac{2\log((1+\varepsilon)s)+\log\left(\frac{1}{\omega c\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right)}\right)}{\omega}\right)}{\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right)} \\
&= \frac{s}{c}\frac{\log\left(\log\left((1+\varepsilon)s\right)\right)+\log\left(\frac{2\log\left(\frac{e}{\omega c\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right)}\right)}{\omega\log((1+\varepsilon)s)}\right)}{\log\left(\log((1+\varepsilon)s)\right)+\log(1/\delta)} \\
&\le \frac{s}{c}\frac{\log\left(\log\left((1+\varepsilon)s\right)\right)+\log\left(2\log\left(\frac{1}{\omega c}\right)/\omega\right)}{\log\left(\log((1+\varepsilon)s)\right)+\log(1/\delta)} \\
&\le \frac{s}{c}\frac{\log\left(2\log\left(\frac{1}{\omega c}\right)/\omega\right)}{\log(1/\delta)}
\end{aligned}
$$

where the second to last line follows if $\log\left((1+\varepsilon)s\right) \ge 1$ and $\log\left(\frac{\log((1+\varepsilon)s)}{\delta}\right) \ge e$ which is satisfied by the assumption.and The last line follows because $\omega \le \delta$ since for any $x > 0$ and $a \ge b$, we have $\frac{x+a}{x+b} \ge \frac{a}{b}$. □

Appendix C

Chapter 7 Supplementary Materials

C.1 Bounds on (κ, μ, δ_0) for some distributions

In this section we relate the function evaluation oracle to the function comparison oracle for some common distributions. That is, if $E_f(x) = f(x) + w$ for some random variable w, we lower bound the probability $\eta(y, x) := \mathbb{P}(\text{sign}\{E_f(y) - E_f(x)\} = \text{sign}\{f(y) - f(x)\})$ in terms of the parameterization of (7.1).

Lemma C.1. *Let w be a Gaussian random variable with mean zero and variance σ^2. Then $\eta(y, x) \geq \frac{1}{2} + \min\left\{\frac{1}{\sqrt{2\pi e}}, \frac{1}{\sqrt{4\pi\sigma^2 e}}|f(y) - f(x)|\right\}$.*

Proof. Notice that $\eta(y, x) = \mathbb{P}(Z + |f(y) - f(x)|/\sqrt{2\sigma^2} \geq 0)$ where Z is a standard normal. The result follows by lower bounding the density of Z by $\frac{1}{\sqrt{2\pi e}}\mathbf{1}\{|Z| \leq 1\}$ and integrating where $\mathbf{1}\{\cdot\}$ is equal to one when its arguments are true and zero otherwise. □

We say w is a 2-sided gamma distributed random variable if its density is given by $\frac{\beta^\alpha}{2\Gamma(\alpha)}|x|^{\alpha-1}e^{-\beta|x|}$ for $x \in [-\infty, \infty]$ and $\alpha, \beta > 0$. Note that this distribution is unimodal only for $\alpha \in (0, 1]$ and is equal to a Laplace distribution for $\alpha = 1$. This distribution has variance $\sigma^2 = \alpha/\beta^2$.

Lemma C.2. *Let w be a 2-sided gamma distributed random variable with parameters $\alpha \in (0, 1]$ and $\beta > 0$. Then $\eta(y, x) \geq \frac{1}{2} + \min\left\{\frac{1}{4\alpha^2\Gamma(\alpha)^2}\left(\frac{\alpha}{e}\right)^{2\alpha}, \frac{(\beta/2e)^{2\alpha}}{4\alpha^2\Gamma(\alpha)^2}|f(y) - f(x)|^{2\alpha}\right\}$.*

Proof. Let $E_f(y) = f(y) + w$ and $E_f(x) = f(x) + w'$ where w and w' are i.i.d. 2-sided gamma distributed random variables. If we lower bound $e^{-\beta|x|}$ with $e^{-\alpha}\mathbf{1}\{|x| \leq \alpha/\beta\}$ and integrate we find that $\mathbb{P}(-t/2 \leq w \leq 0) \geq \min\left\{\frac{1}{2\alpha\Gamma(\alpha)}\left(\frac{\alpha}{e}\right)^{\alpha}, \frac{(\beta/e)^{\alpha}}{2\alpha\Gamma(\alpha)}(t/2)^{\alpha}\right\}$. And by the symmetry and independence of w and w' we have

$$\mathbb{P}(-t \leq w - w') \geq \frac{1}{2} + \mathbb{P}(-t/2 \leq w \leq 0)\mathbb{P}(-t/2 \leq w \leq 0).$$

□

While the bound in the lemma immediately above can be shown to be loose, these two lemmas are sufficient to show that the entire range of $\kappa \in (1, 2]$ is possible.