Learning and Decision Making from Diverse Forms of Information

Yichong Xu

June 2020
CMU-ML-20-108

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Artur Dubrawski, Co-Chair
Aarti Singh, Co-Chair
Sivaraman Balakrishnan
John Langford (Microsoft Research)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by: Air Force Research Laboratory award numbers FA87501220324, FA87501420244, FA87501720130 and FA87501720212; National Institutes of Health award numbers R01GM117622, R01HL141916, R01HL144692 and R01NR013912; National Science Foundation award numbers CCF1563918, CCF1763734 and IIS1320347; and Innovation Works award number 2015WCZ01733E1.

Keywords: interactive learning, pairwise comparison, multitask learning, preference learning, ranking

To my family, my girlfriend, and everyone I have ever met

Abstract

Classical machine learning posits that data are independently and identically distributed, in a single format usually the same as test data. In modern applications however, additional information in other formats might be available freely or at a lower cost. For example, in data crowdsourcing we can collect preferences over the data points instead of directly asking the labels of a single data point at a lower cost. In natural language understanding problems, we might have limited amount of data in the target domain, but can use a large amount of general domain data for free.

The main topic of this thesis is to study how to efficiently incorporate these diverse forms of information into the learning and decision making process. We study two representative paradigms in this thesis.

- Firstly, we study learning and decision making problems with direct labels and comparisons. In many applications such as clinical settings and material science, comparisons are much cheaper to obtain than direct labels. We show that comparisons can greatly reduce the problem complexity; using comparisons as input, our algorithm requires an exponentially smaller amount of labels to work than traditional label-only algorithms. Moreover, our total query complexity is similar to previous algorithms. We consider various learning problems in this settings, including classification, regression, multi-armed bandits, nonconvex optimization, and reinforcement learning.
- Secondly, we study multi-task learning and transfer learning to learn from different domains and tasks of data. In this case, our algorithm use the previous collected data from similar tasks or domains, which are essentially free to use. We propose simple yet effective ways to transfer the knowledge from other domains and tasks, and achieve state-of-the-art result on several natural language understanding benchmarks.

We illustrate both theoretical and practical insights in this thesis. Theoretically, we show performance guarantees of our algorithms as well as their statistical minimaxity through information-theoretic limits. On the practical side, we demonstrate promising experimental results on price estimation and natural language understanding tasks.

Acknowledgments

First and foremost, I would like to thank my PhD advisors Artur Dubrawski and Aarti Singh. It is no doubt that my advisors shape me as a researcher in every aspect, from my research taste, my topics, to how I write scientific papers. Not only did I learn about the knowledge, but also I learn from their methods and thoughts to tackle research problems. From another perspective, I think I should also thank them for paying me salary so that I will not go bankrupt during my PhD journey (laugh).

Thank Sivaraman Balakrishnan and John Langford as my thesis committee members. It was wonderful experience working with Siva during my PhD. He helped me a lot on my paper writing skills, and I learned a lot from his writings. I had many exciting discussions on reinforcement learning with John, and that inspires a lot of my interest. I would also like to thank both of them for their helpful comments to improve this thesis.

During my PhD, I had two exciting internships at Microsoft Research. I would like to thank Jianfeng Gao, Jingjing Liu, Xiaodong Liu and Yelong Shen, who mentored and helped me a lot during these internships. Natural language processing was a complete new area for me, and I was lucky to have them when I entered this area. I learned a ton about the methodology and think process for doing experiments on machine learning. Special thanks to Xiaodong, who helped me a lot on coding and tricks. My coding improved every time I went to an internship.

I would like to thank all my collaborators: Sivaraman Balakrishnan, Xi Chen, Artur Dubrawski, Jianfeng Gao, Aparna Joshi, Chunyuan Li, Jingjing Liu, Xiaodong Liu, Hariank Muthakana, Kyle Miller, Hoifung Poon, Nihar B. Shah, Yelong Shen, Xiaofei Shi, Pingzhong Tang, Yifeng Teng, Ruosong Wang, Yuexin Wu, Shenke Xiao, Tianjun Xiao, Kuiyuan Yang, Lin F. Yang, Yiming Yang, Hongyang Zhang, Jiaxing Zhang, Zheng Zhang and Han Zhao. It was a nice experience working with all of you. My special thank goes to Nihar, Shah, Pingzhong Tang, Jiaxing Zhang and Zheng Zhang. I worked a lot with Nihar after he came to CMU, and I learned a lot from his ways of tackling research problems. I would also like to thank Pingzhong, Jiaxing and Zheng for being my mentors when I was undergraduate and spawning my interest in scientific research.

I would like to thank all of my friends, from CMU, from Microsoft Research, from Tsinghua University, and from anywhere in the world. Friendship means a lot for someone like me, who came to an entirely new country for my PhD. I really appreciate all the time I spent with friends. I would like to all of them for the joy, the support, the excitement that they gave me. I would also thank Diane Stidle and other amazing administrative staff at CMU MLD for supporting the PhD students here.

Last but most importantly, I would like to thank my family and my girlfriend Zhixin Li for their unconditional love and support during my PhD. Doing a PhD is never an easy thing for me, and I would not have done it without their help. I thank them sincerely.

Overall, my PhD is a difficult but rewarding journey. Research is like a box of chocolates. You never know whether you are gonna get ground-breaking and inspiring results, or another piece of junk that is not meaningful anyway. PhD is a wonderful journey, but my life journey has just begun. All in all, I appreciate all my experience that brings me here, and I will also appreciate everywhere they bring me to in the future of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditionally, machine learning deal with data from a single distribution and in a fixed form, usually independently and identically distributed. In many practical applications, however, we can obtain data in diverse forms for a single target task, with varied costs and qualities. A natural question is that whether and how algorithms can benefit from using these diverse forms of information. In this thesis we aim to answer this question by analyzing how diverse information can be incorporated into learning algorithms.

Multi-formed data can come in various ways in practice, and we study two different paradigms in this thesis. We first consider multi-formed data from feedbacks; during data acquisition, workers can not only answer about the label of a specific sample, but also many other alternative questions to help understand the problem. A broad goal is to understand the usefulness of, and to design algorithms to exploit, such alternative feedbacks. We study a special version of such feedbacks, namely learning from pairwise comparisons, in this thesis.

We then consider multi-formed data from multiple domains. In many applications, the available data is limited. However, we can obtain data from other domains, effectively of a different distribution, at essentially no cost. For example, for understanding Wikipedia, we can also use other forms of text, such as news and webpages, to train our model. We study the application of multi-task learning to understand multi-domain data for natural language understanding (NLU) problems.

By demonstrating the results of learning from multiple feedbacks and multiple domains, we illustrate the effectiveness of using multiple-source data in learning and decision making problems. We give an overview of our results in the next section.

1.1 Overview of Results

We review our results in this section. In Section 1.1.1, we illustrate our results on learning from both labels and comparisons. In Section 1.1.2, we describe the dueling-choice framework for decision making, where comparisons (duels) are available in addition to traditional direct queries. Lastly in Section 1.1.3, we study the problem of learning from multiple domains on natural language understanding tasks.

1.1.1 Interactive Learning from Labels and Comparisons

Classical classification and regression algorithms is centered around using labeled observations $\{(X_1, y_1), ..., (X_n, y_n)\}$, where $X_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$ for binary classification, or $y_i \in \mathbb{R}$ for regression. In Chapter We consider an alternative way of interaction in this setup, where in addition to direct labels, we can also compare two points (X_i, X_j) in the data pool, and obtain the point with a larger y value.

In many applications of machine learning comparisons are easier to obtain than labels. For example, in crowdsourcing, workers are often able to provide ordinal feedback accurately with little effort [153, 164]. Similarly, in clinical settings, precise assessment of each individual patient's health status can be difficult, expensive and/or risky (e.g. it may require application of invasive sensors or diagnostic surgeries), but comparing relative statuses of two patients at a time may be relatively easy and accurate.

In Chapter 2, we consider the problem of active classification with both labels and comparisons, and characterize how the access to an easier comparison oracle helps in improving the label and total query complexity. We show that the comparison oracle reduces the learning problem to that of learning a threshold function. We then present an algorithm that interactively queries the label and comparison oracles and we characterize its query complexity under Tsybakov and adversarial noise conditions for the comparison and labeling oracles. This chapter is based on [187].

In Chapter 3, we consider linear and nonparametric regression with additional ordinal information. We consider ordinal feedback of varying qualities where we have either a perfect ordering of the samples, a noisy ordering of the samples or noisy pairwise comparisons between the samples. We provide a precise quantification of the usefulness of these types of ordinal feedback in both nonparametric and linear regression, showing that in many cases it is possible to accurately estimate an underlying function with a very small labeled set, effectively *escaping the curse of dimensionality*. We also present lower bounds, that establish fundamental limits for the task and show that our algorithms are optimal in a variety of settings. Finally, we present extensive experiments on new datasets that demonstrate the efficacy and practicality of our algorithms and investigate their robustness to various sources of noise and model misspecification. This chapter is based on [185, 188].

1.1.2 Decision Making with Comparisons

Many practical problems can be casted as bandit or optimization problems, where we aim to find target items in a given (finite or infinite) set. In the classical Multi-Armed Bandit(MAB) framework, we selectively query (or pull) an arm to get a random reward from its mean distribution in each iteration. We consider the same additional interaction with comparisons as in the previous section: In each iteration in addition to pulls, we can also compare (or duel) two arms to get the arm with a larger reward. We refer to this problem as Multi-Armed Bandits with Dueling Choices (MAB-DC) since MAB with comparisons is usually called dueling bandits in literature [200], and duels are optional in the algorithm process.

In Chapter 4, we consider regret minimization for MAB-DC under both discrete and continuous arm spaces. For the discrete K-armed MAB-DC, we propose the DF algorithm by combining Beat-the-Mean [199] and Successive Elimination[70]. We show that DF acehives a $O(\log T)$

regret rate where T is the number of direct queries, and the regret combines the benefit of both duels and pulls. For a continuous arm space, we give the COMP-GP-UCB algorithm based on GP-UCB [156], where instead of directly querying the point with the maximum Upper Confidence Bound (UCB), we perform constrained optimization and use comparisons to filter out suboptimal points. COMP-GP-UCB comes with theoretical guarantee of $O(\frac{\Phi}{\sqrt{T}})$ on simple regret where Φ is an improved information gain stemming from a comparison-based constraint set that restricts the space for optimum search. In contrast, in the plain direct query setting, Φ depends on the entire domain. We discuss theoretical aspects and show experimental results to demonstrate efficacy of our algorithms. This chapter extends the content of [189].

In Chapter 5, we consider the Thresholding Bandit Problem (TBP) with Dueling Choices. The Thresholding Bandit Problem (TBP) aims to find the set of arms with mean rewards greater than a given threshold. We provide an algorithm called Rank-Search (RS) for solving TBP-DC by alternating between ranking and binary search. We prove theoretical guarantees for RS, and also give lower bounds to show the optimality of it. Experiments show that RS can outperform baseline algorithms when duels can be obtained at a lower cost than direct assessments. This chapter is based on [192].

In Chapter 6, we consider reinforcment learning with comparisons, namely Preference-based Reinforcement Learning (PbRL). PbRL replaces reward values in traditional reinforcement learning by preferences to better elicit human opinion on the target objective, specially when reward metrics are hard to design or elicit. Despite promising results in applications, the theoretical understanding of PbRL is still preliminary. We present the first finite-time analysis for general PbRL problems. We first show that a unique optimal policy may not exist if preferences over trajectories are deterministic for PbRL. If preferences are stochastic, and the preference probability relates to the reward values, we present algorithms for PbRL both with or without a simulator that are able to identify the best policy up to accuracy epsilon with high probability. Our method explores the state space by navigating to every (remove every since states with very low reach probability are not visited?) under-explored states, and solves PbRL using a combination of dueling bandits and policy search. Experiments show the efficacy of our method on real-world problems. This chapter is based on [193].

1.1.3 Natural Language Understanding from Multiple Domains

We focus another paradigm of learning from multiple sources, where we have data from multiple domains, and would like to learn a joint model targeting the performance on a specific dataset. The domains can be different in their input/output formats, data distributions and label distributions. We focus on the problem of Natural Language Understanding (NLU) problems, since NLU data is typically collected from different sources using various corpora.

In Chapter 6, we consider the problem of multi-task learning for machine reading comprehension(MRC). We propose a framework to learn a joint MRC model that can be applied to a wide range of MRC tasks in different domains. Inspired by recent ideas of data selection in machine translation, we develop a novel sample re-weighting scheme to assign sample-specific weights to the loss. Empirical study shows that our approach can be applied to many existing MRC models. Combined with contextual representations from pre-trained language models (such as ELMo), we achieve new state-of-the-art results on a set of MRC benchmark datasets. This work is based on

[191].

In Chapter 7, we consider the problem of transfer multiple source of knowledge to NLU in the medical domain. We use a multi-source transfer learning approach to transfer the knowledge from MT-DNN [119] (general domain NLU) and SciBERT [33] (language model in the medical domain). For transfer learning fine-tuning, we use multi-task learning various tasks on general and medical domains to improve performance. The proposed methods are proved effective for natural language understanding in the medical domain, and we rank the 2nd on the QA task of the MEDIQA-2019 shared task[34]. This work is based on [190].

Part I

Interactive Learning from Labels and Comparisons

Chapter 2

Classification with Labels and Comparisons

Given high costs of obtaining labels for big datasets, interactive learning is gaining popularity in both practice and theory of machine learning. On the practical side, there has been an increasing interest in designing algorithms capable of engaging domain experts in two-way queries to facilitate more accurate and more effort-efficient learning systems (c.f. [124, 171]). On the theoretical side, study of interactive learning has led to significant advances such as exponential improvement of query complexity over passive learning under certain conditions (c.f. [14, 15, 26, 44, 86, 146]). While most of these approaches to interactive learning fix the form of an oracle, e.g., the labeling oracle, and explore the best way of querying, recent work allows for multiple diverse forms of oracles [25, 36, 64, 196]. The focus of this chapter is on this latter setting, also known as active dual supervision [11]. We investigate how to recover a hypothesis h that is a good approximator of the optimal classifier h^*, in terms of expected 0/1 error $\Pr_X[h(X) \neq h^*(X)]$, given limited access to labels on individual instances $X \in \mathcal{X}$ and pairwise comparisons about which one of two given instances is more likely to belong to the +1/-1 class.

Our study is motivated by important applications where comparisons are easier to obtain than labels, and the algorithm can leverage both types of oracles to improve label and total query complexity. For example, in material design, synthesizing materials for specific conditions requires expensive experimentation, but with an appropriate algorithm we can leverage expertize of material scientists, for whom it may be hard to accurately assess the resulting material properties, but who can quickly compare different input conditions and suggest which ones are more promising. Similarly, in clinical settings, precise assessment of each individual patient's health status can be difficult, expensive and/or risky (e.g. it may require application of invasive sensors or diagnostic surgeries), but comparing relative statuses of two patients at a time may be relatively easy and accurate. In both these scenarios we may have access to a modest amount of individually labeled data, but the bulk of more accessible training information is available via pairwise comparisons. There are many other examples where humans find it easier to perform pairwise comparisons rather than providing direct labels, including content search [77], image retrieval [171], ranking [90], etc.

Despite many successful applications of comparison oracles, many fundamental questions

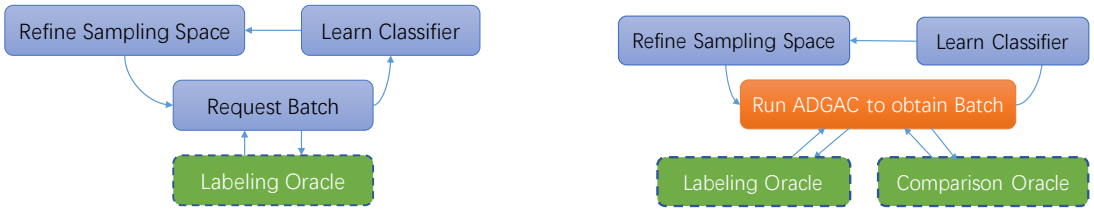


Figure 2.1: Explanation of work flow of ADGAC-based algorithms. **Left:** Procedure of typical active learning algorithms. **Right:** Procedure of our proposed ADGAC-based interactive learning algorithm which has access to both pairwise comparison and labeling oracles.

remain. One of them is how to design *noise-tolerant*, *cost-efficient* algorithms that can approximate the unknown target hypothesis to arbitrary accuracy while having access to pairwise comparisons. On one hand, while there is theoretical analysis on the pairwise comparisons concerning the task of learning to rank [7, 95], estimating ordinal measurement models [152] and learning combinatorial functions [28], much remains unknown how to extend these results to more generic hypothesis classes. On the other hand, although we have seen great progress on using single or multiple oracles with the same form of interaction [23, 64], classification using both comparison and labeling queries remains an interesting open problem. Independently of our work, Kane et al. [105] concurrently analyzed a similar setting of learning to classify using both label and comparison queries. However, their algorithms work only in the noise-free setting.

Our Contributions: Our work addresses the aforementioned issues by presenting a new algorithm, Active Data Generation with Adversarial Comparisons (ADGAC), which learns a classifier with both noisy labeling and noisy comparison oracles.

- We analyze ADGAC under Tsybakov (TNC) [165] and adversarial noise conditions for the labeling oracle, along with the adversarial noise condition for the comparison oracle. Our general framework can augment any active learning algorithm by replacing the batch sampling in these algorithms with ADGAC. Figure 2.1 presents the work flow of our framework.
- We propose A^2-ADGAC algorithm, which can learn an arbitrary hypothesis class. The label complexity of the algorithm is as small as learning a threshold function under both TNC and adversarial noise condition, independently of the structure of the hypothesis class. The *total query complexity* improves over previous best-known results under TNC which can only access the labeling oracle.
- We derive Margin-ADGAC to learn the class of halfspaces. This algorithm has the same label and total query complexity as A^2-ADGAC, but is computationally efficient.
- We present lower bounds on total query complexity for any algorithm that can access both labeling and comparison oracles, and a noise tolerance lower bound for our algorithms. These lower bounds demonstrate that our analysis is nearly optimal.

An important quantity governing the performance of our algorithms is the adversarial noise level of comparisons: denote by $\mathsf{Tol}_{\text{comp}}(\varepsilon, \delta, \mathcal{A})$ the adversarial noise tolerance level of comparisons that guarantees an algorithm $\mathcal{A}$ to achieve an error of ε, with probability at least $1 - \delta$. Table 2.1 compares our results with previous work in terms of label complexity, total query complexity,

Table 2.1: Comparison of various methods for learning of generic hypothesis class (Omitting $\log(1/\varepsilon)$ factors).

Label Noise	Work	# Label	# Query	$\mathsf{Tol}_{\text{comp}}$
Tsybakov (κ)	[84]	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} d\theta\right)$	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} d\theta\right)$	N/A
Tsybakov (κ)	**Ours**	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right)$	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} \theta + d\theta\right)$	$\mathcal{O}(\varepsilon^{2\kappa})$
Adversarial ($\nu = \mathcal{O}(\varepsilon)$)	[86]	$\tilde{\mathcal{O}}(d\theta)$	$\tilde{\mathcal{O}}(d\theta)$	N/A
Adversarial ($\nu = \mathcal{O}(\varepsilon)$)	**Ours**	$\tilde{\mathcal{O}}(1)$	$\tilde{\mathcal{O}}(d\theta)$	$\mathcal{O}(\varepsilon^2)$

Table 2.2: Comparison of various methods for learning of halfspaces (Omitting $\log(1/\varepsilon)$ factors).

Label Noise	Work	# Label	# Query	$\mathsf{Tol}_{\text{comp}}$	Efficient?
Massart	[27]	$\tilde{\mathcal{O}}(d)$	$\tilde{\mathcal{O}}(d)$	N/A	No
Massart	[14]	$\text{poly}(d)$	$\text{poly}(d)$	N/A	Yes
Massart	**Ours**	$\tilde{\mathcal{O}}(1)$	$\tilde{\mathcal{O}}(d)$	$\mathcal{O}(\varepsilon^2)$	Yes
Tsybakov (κ)	[86]	$\tilde{\mathcal{O}}(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} d\theta)$	$\tilde{\mathcal{O}}(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} d\theta)$	N/A	No
Tsybakov (κ)	**Ours**	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right)$	$\tilde{\mathcal{O}}\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-2} + d\right)$	$\mathcal{O}(\varepsilon^{2\kappa})$	Yes
Adversarial ($\nu = \mathcal{O}(\varepsilon)$)	[202]	$\tilde{\mathcal{O}}(d)$	$\tilde{\mathcal{O}}(d)$	N/A	No
Adversarial ($\nu = \mathcal{O}(\varepsilon)$)	[15]	$\tilde{\mathcal{O}}(d^2)$	$\tilde{\mathcal{O}}(d^2)$	N/A	Yes
Adversarial ($\nu = \mathcal{O}(\varepsilon)$)	**Ours**	$\tilde{\mathcal{O}}(1)$	$\tilde{\mathcal{O}}(d)$	$\mathcal{O}(\varepsilon^2)$	Yes

and $\mathsf{Tol}_{\text{comp}}$ for generic hypothesis class $\mathbb{C}$ with error ε. We see that our results significantly improve over prior work with the extra comparison oracle. Denote by d the VC-dimension of $\mathbb{C}$ and θ the disagreement coefficient. We also compare the results in Table 2.2 for learning halfspaces under isotropic log-concave distributions. In both cases, our algorithms enjoy small label complexity that is independent of θ and d. This is helpful when labels are very expensive to obtain. Our algorithms also enjoy better total query complexity under both TNC and adversarial noise condition for efficiently learning halfspaces.

2.1 Our Techniques

Intransitivity: The main challenge of learning with pairwise comparisons is that the comparisons might be *asymmetric* or *intransitive*. If we construct a classifier $h(x)$ by simply comparing x with a fixed instance $\hat{x}$ by comparison oracle, then the concept class of classifiers $\{h : h(x) = Z(x, \hat{x}), \hat{x} \in \mathcal{X}\}$ will have infinite VC dimension, so the complexity will be as high as infinite if we apply the traditional tools of VC theory. To resolve the issue, we conduct a group-based binary search in ADGAC. The intuition is that by dividing the dataset into several ranked groups $S_1, S_2, ...$, the majority of labels in each group can be stably decided if we sample enough examples from that group. Therefore, we are able to reduce the original problem in the high-dimensional

space to the problem of learning a "threshold" function in one-dimension space. Then some straightforward approaches such as binary search learns the thresholding function.

Combining with Active Learning Algorithms: If the labels follow Tsybakov noise (i.e., Condition 2), the most straightforward method to combine ADGAC with existing algorithms is to combine ADGAC with an algorithm that uses the label oracle only and works under TNC. However, we cannot save query complexity if we follow this method. To see this, notice that in each round we need roughly $n_i = \tilde{\mathcal{O}}\left(d\theta\left(\frac{1}{\varepsilon_i}\right)^{2\kappa-1}\right)$ samples and $m_i = \tilde{\mathcal{O}}\left(d\theta\left(\frac{1}{\varepsilon_i}\right)^{2\kappa-2}\right)$ labels; if we use ADGAC, we can obtain a labeling of n_i samples with at most $\varepsilon_i n_i \approx m_i$ errors with low label complexity. Suppose N is the set of labels that ADGAC makes error on. However, since the outside active learning algorithm works under TNC, we will need to query labels in N to make sure that the ADGAC labels follow TNC. That means our label complexity is still m_i, the same as the original algorithm. To avoid this problem, we combine ADGAC with algorithms under adversarial noise in all cases including TNC. This eliminates the need to query additional labels, and also reduces the query complexity.

Handling Independence: We mostly follow previous works on combining ADGAC with existing algorithms. However, since we now obtain labels from ADGAC instead of $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$, the labels are not independently sampled, and we need to adapt the proof to our case. We use different methods for A^2-ADGAC and Margin-ADGAC: For the former, we use results from PAC learning to bound the error on all n_i samples; for the latter, we decompose the error of any classifier h on labels generated by ADGAC into two parts: The first part is caused by the error of ADGAC itself, and second is by h on truthful labels. Using the above techniques enables us to circumvent the independence problem.

Lower Bounds: It is typically hard to provide a unified lower bound for multi-query learning framework, as several quantities are simultaneously involved in the analysis, e.g., the comparison complexity, the label complexity, the noise tolerance, etc. So traditional proof techniques for active learning, e.g., Le Cam's and Fano's bounds [44, 86], cannot be trivially applied to our setting. Instead, we prove lower bounds on one quantity by allowing arbitrary budgets of other quantities. Another non-trivial technique is in the proof of minimax bound for the adversarial noise level of comparison oracle (see Theorem 12): In the proof of upper bound, we divide the integral region w.r.t. the expectation into n segments, each of size $1/n$, and the expectation is thus the limit when $n \to \infty$. We upper bound the discrete approximation of the integral by a careful calibration of noise on each segment for a fixed n, and then let $n \to \infty$. The proof then leads to a general inequality (Lemma 21), and it might be of independent interest.

2.2 Related Works

It is well known that people are better at comparison than labeling [152, 158]. It has been widely used to tackle problems in classification [124], clustering [109] and ranking [5, 77].

Balcan et al. [28] studied using pairwise comparisons to learn submodular functions on sets. Another related problem is bipartite ranking [4], which exactly does the opposite of our problem: Given a group of binary labels, learn a ranking function that rank positive samples higher than negative ones.

Interactive learning has wide application in the field of computer vision and natural language processing (see e.g., [171]). There are also abundant literatures on interactive ways to improve unsupervised and semi-supervised learning [109]. However, there lacks a general statistical analysis of interactive learning for traditional classification tasks. Balcan and Hanneke [23] analyze class conditional queries (CCQ), where the user gives counterexamples to a given classification. Beygelzimer et al. [36] used a similar idea using search queries. However, their interactions requires a oracle that is usually stronger than the traditional labelers (i.e., we can simulate traditional active learning using such oracles), and is generally hard to deploy in practice. There turns out to be little general analysis on using a "weaker" interaction between human and computer. Balcan and Hanneke[23] studied an abstract query based notions from exact learning, but their analysis cannot handle queries that gives relation between samples (as comparisons do). Our work fits in this blank.

We compare our work to traditional label-based active learning [86], which has drawn a lot of attention in the society in recent years. Disagreement-based active learning has been shown to reach a near-optimal rate on classification problems [84]. Another line of research is margin-based active learning [14], which aims at computational efficiency of learning halfspaces, under the large-margin assumption.

2.3 Preliminaries

Notations: We study the problem of learning a classifier $h : \mathcal{X} \to \mathcal{Y} = \{-1, 1\}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the instance space and label space, respectively. Denote by $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$ the distribution over $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{P}_{\mathcal{X}}$ be the marginal distribution over $\mathcal{X}$. A hypothesis class $\mathbb{C}$ is a set of functions $h : \mathcal{X} \to \mathcal{Y}$. For any function h, define the error of h under distribution D over $\mathcal{X} \times \mathcal{Y}$ as $\mathsf{err}_D(h) = \Pr_{(X,Y)\sim D}[h(X) \neq Y]$. Let $\mathsf{err}(h) = \mathsf{err}_{P_{\mathcal{X}\mathcal{Y}}}(h)$. Suppose that $h^* \in \mathbb{C}$ satisfies $\mathsf{err}(h^*) = \inf_{h\in\mathbb{C}} \mathsf{err}(h)$. For simplicity, we assume that such an h^* exists in class $\mathbb{C}$.

We apply the concept of disagreement coefficient from Hanneke [84] for generic hypothesis class in this chapter. In particular, for any set $V \subseteq \mathbb{C}$, we denote by $\mathsf{DIS}(V) = \{x \in \mathcal{X} : \exists h_1, h_2 \in V, h_1(x) \neq h_2(x)\}$. The disagreement coefficient is defined as $\theta = \sup_{r>0} \frac{\Pr[\mathsf{DIS}(B(h^*,r))]}{r}$, where $B(h^*, r) = \{h \in \mathbb{C} : \Pr_{X\sim\mathcal{P}_{\mathcal{X}}}[h(X) \neq h^*(X)] \leq r\}$.

Problem Setup: We analyze two kinds of noise conditions for the labeling oracle, namely, adversarial noise condition and Tsybakov noise condition (TNC). We formally define them as follows.

Condition 1 (Adversarial Noise Condition for Labeling Oracle)**.** *Distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$ satisfies adversarial noise condition for labeling oracle with parameter $\nu \geq 0$, if $\nu = \Pr_{(X,Y)\sim\mathcal{P}_{\mathcal{X}\mathcal{Y}}}[Y \neq h^*(X)]$.*

Condition 2 (Tsybakov Noise Condition for Labeling Oracle)**.** *Distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$ satisfies Tsybakov noise condition for labeling oracle with parameters $\kappa \geq 1, \mu \geq 0$, if $\forall h : \mathcal{X} \to \{-1, 1\}, \mathsf{err}(h) - \mathsf{err}(h^*) \geq \mu \Pr_{X\sim\mathcal{P}_{\mathcal{X}}}[h(X) \neq h^*(X)]^\kappa$. Also, h^* is the Bayes optimal classifier, i.e., $h^*(x) = sign(\eta(x) - 1/2)$. [1] where $\eta(x) = \Pr[Y = 1|X = x]$. The special case of $\kappa = 1$ is*

[1]The assumption that h^* is Bayes optimal classifier can be relaxed if the approximation error of h^* can be quantified under assumptions on the decision boundary (c.f. [44]).

also called Massart noise condition.

In the classic active learning scenario, the algorithm has access to an unlabeled pool drawn from $\mathcal{P}_{\mathcal{X}}$. The algorithm can then query the labeling oracle for any instance from the pool. The goal is to find an $h \in \mathbb{C}$ such that the error $\Pr[h(X) \neq h^*(X)] \leq \varepsilon$[2]. The labeling oracle has access to the input $x \in \mathcal{X}$, and outputs $y \in \{-1, 1\}$ according to $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$. In our setting, however, an extra comparison oracle is available. This oracle takes as input a pair of instances $(x, x') \in \mathcal{X} \times \mathcal{X}$, and returns a variable $Z(x, x') \in \{-1, 1\}$, where $Z(x, x') = 1$ indicates that x is more likely to be positive, while $Z(x, x') = -1$ otherwise. In this chapter, we discuss an adversarial noise condition for the comparison oracle. We discuss about other conditions on comparisons at the end of this section.

Condition 3 (Adversarial Noise Condition for Comparison Oracle)**.** *Distribution $\mathcal{P}_{\mathcal{X}\mathcal{X}\mathcal{Z}}$ satisfies adversarial noise with parameter $\nu' \geq 0$, if $\nu' = \Pr[Z(X, X')(h^*(X) - h^*(X')) < 0]$.*

Note that we do not make any assumptions on the randomness of Z: $Z(X, X')$ can be either random or deterministic as long as the joint distribution $P_{\mathcal{X}\mathcal{X}\mathcal{Z}}$ satisfies Condition 3.

For an interactive learning algorithm $\mathcal{A}$, given error ε and failure probability δ, let $\mathsf{SC}_{\text{comp}}(\varepsilon, \delta, \mathcal{A})$ and $\mathsf{SC}_{\text{label}}(\varepsilon, \delta, \mathcal{A})$ be the comparison and label complexity, respectively. The query complexity of $\mathcal{A}$ is defined as the sum of label and comparison complexity. Similar to the definition of $\mathsf{Tol}_{\text{comp}}(\varepsilon, \delta, \mathcal{A})$, define $\mathsf{Tol}_{\text{label}}(\varepsilon, \delta, \mathcal{A})$ as the maximum ν such that algorithm $\mathcal{A}$ achieves an error of at most ε with probability $1 - \delta$. As a summary, $\mathcal{A}$ learns an h such that $\Pr[h(X) \neq h^*(X)] \leq \varepsilon$ with probability $1 - \delta$ using $\mathsf{SC}_{\text{comp}}(\varepsilon, \delta, \mathcal{A})$ comparisons and $\mathsf{SC}_{\text{label}}(\varepsilon, \delta, \mathcal{A})$ labels, if $\nu \leq \mathsf{Tol}_{\text{label}}(\varepsilon, \delta, \mathcal{A})$ and $\nu' \leq \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta, \mathcal{A})$. We omit the parameters of $\mathsf{SC}_{\text{comp}}, \mathsf{SC}_{\text{label}}, \mathsf{Tol}_{\text{comp}}, \mathsf{Tol}_{\text{label}}$ if they are clear from the context. We use $\mathcal{O}(\cdot)$ to express sample complexity and noise tolerance, and $\tilde{O}(\cdot)$ to ignore the $\log(\cdot)$ terms. Table 2.3 summarizes the main notations throughout the chapter.

Learning under TNC for comparisons. In this section we justify our choice of analyzing adversarial noise model for the comparison oracle. In fact, any algorithm using adversarial comparisons can be transformed into an algorithm using TNC comparisons, by treating learning comparison functions as a separate learning problem. Let $\mathbb{C}'$ be a hypothesis class consisting of comparison functions $f : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$. Suppose the optimal comparison function is $f^*(x, x') = \text{sign}(g^*(x) - g^*(x'))$, and Tsybakov noise condition holds for $((X, X'), Z)$ with some constant μ', κ'; i.e., for any $f \in \mathbb{C}'$ we have

$$\Pr[f(X, X') \neq Z] - \Pr[f^*(X, X') \neq Z] \geq \mu' \Pr[f(X, X') \neq f^*(X, X')]^{\kappa'}.$$

Also suppose $f^*(x, x') = \text{sign}(\Pr[Z = 1 | X = x, X' = x'] - 1/2)$. Assume $\mathbb{C}'$ has VC-dimension d' and disagreement coefficient θ', standard active learning requires

$$\Phi(\nu') = \tilde{\mathcal{O}}\left(\theta'\left(\frac{1}{\nu'}\right)^{2\kappa'-2}(d' \log(\theta') + \log(1/\delta)) \log\left(\frac{1}{\nu'}\right)\right)$$

samples to learn a comparison function of error ν' with probability $1 - \delta$. So an algorithm $\mathcal{A}$ for adversarial noise on comparisons can be automatically transformed into an algorithm $\mathcal{A}'$ for

[2]Note that we use the disagreement $\Pr[h(X) \neq h^*(X)]$ instead of the excess error $\mathsf{err}(h) - \mathsf{err}(h^*)$ in some of the other literatures. The two conditions can be linked by assuming a two-sided version of Tsybakov noise (see e.g., Audibert 2004).

Table 2.3: Summary of notations.

Notation	Meaning
$\mathbb{C}$	Hypothesis class
$X, \mathcal{X}$	Instance & Instance space
$Y, \mathcal{Y}$	Label & Label space
$Z, \mathcal{Z}$	Comparison & Comparison space
d	VC dimension of $\mathbb{C}$
θ	Disagreement coefficient
h^*	Optimal classifier in $\mathbb{C}$
g^*	Optimal scoring function
κ	Tsybakov noise level (labeling)
ν	Adversarial noise level (labeling)
ν'	Adversarial noise level (comparison)
$\mathsf{err}_D(h)$	Error of h on distribution D
$\mathsf{SC}_{\text{label}}$	Label complexity
$\mathsf{SC}_{\text{comp}}$	Comparison complexity
$\mathsf{Tol}_{\text{label}}$	Noise tolerance (labeling)
$\mathsf{Tol}_{\text{comp}}$	Noise tolerance (comparison)

TNC on comparisons with $\mathsf{SC}_{\text{label}}(\mathcal{A}') = \mathsf{SC}_{\text{label}}(\mathcal{A})$ and $\mathsf{SC}_{\text{comp}}(\mathcal{A}) = \Phi(\mathsf{Tol}_{\text{comp}}(\mathcal{A}))$. So we only analyze adversarial noise for comparison in other parts of this chapter.

2.4 The ADGAC Algorithm

The hardness of learning from pairwise comparisons follows from the error of comparison oracle: the comparisons are *noisy*, and can be *asymmetric* and *intransitive*, meaning that the human might give contradicting preferences like $x_1 \preccurlyeq x_2 \preccurlyeq x_1$ or $x_1 \preccurlyeq x_2 \preccurlyeq x_3 \preccurlyeq x_1$ (here $\preccurlyeq$ is some preference). This makes traditional methods, e.g., defining a function class $\{h : h(x) = Z(x, \hat{x}), \hat{x} \in \mathcal{X}\}$, fail, because such a class may have infinite VC dimension.

In this section, we propose a novel algorithm, Active Data Generation with Adversarial Comparisons (ADGAC), to address this issue. Having access to both comparison and labeling oracles, ADGAC generates a labeled dataset by techniques inspired from group-based binary search. We show that ADGAC can be combined with any active learning procedure to obtain interactive algorithms that can utilize both labeling and comparison oracles. We provide theoretical guarantees for ADGAC.

2.4.1 Algorithm Description

To illustrate ADGAC, we start with a general active learning framework in Algorithm 1. Many active learning algorithms can be adapted to this framework, such as A^2 [26] and margin-based active algorithms [14, 15]. Here U represents the querying space/disagreement region of the

algorithm (i.e., we reject an instance x if $x \notin U$), and V represents a version space consisting of potential classifiers. For example, A^2 algorithm can be adapted to Algorithm 1 straightforwardly by keeping U as the sample space and V as the version space. More concretely, A^2 algorithm [26] for adversarial noise can be characterized by

$$U_0 = \mathcal{X},\ V_0 = \mathbb{C},\ f_V(U, V, W, i) = \{h : |W|\mathsf{err}_W(h) \leq n_i\varepsilon_i\},\ f_U(U, V, W, i) = \mathsf{DIS}(V),$$

where ε_i and n_i are parameters of the A^2 algorithm, and $\mathsf{DIS}(V) = \{x \in \mathcal{X} : \exists h_1, h_2 \in V, h_1(x) \neq h_2(x)\}$ is the disagreement region of V. Margin-based active learning [15] can also be fitted into Algorithm 1 by taking V as the halfspace that (approximately) minimizes the hinge loss, and U as the region within the margin of that halfspace.

Algorithm 1 Active Learning Framework

Input: ε, δ, a sequence of n_i, functions f_U, f_V.

1: Initialize $U \leftarrow U_0 \subseteq \mathcal{X}, V \leftarrow V_0 \subseteq \mathbb{C}$.
2: **for** $i = 1, 2, ..., \log(1/\varepsilon)$ **do**
3: Sample unlabeled dataset $\tilde{S}$ of size n_i. Let $S \leftarrow \{x : x \in \tilde{S}, x \in U\}$.
4: Request the labels of $x \in S$ and obtain $W \leftarrow \{(x_i, y_i) : x_i \in S\}$.
5: Update $V \leftarrow f_V(U, V, W, i)$, $U \leftarrow f_U(U, V, W, i)$.
6: **end for**

Output: Any classifier $\hat{h} \in V$.

To efficiently apply the comparison oracle, we propose to replace step 4 in Algorithm 1 with a subroutine, ADGAC, that has access to both comparison and labeling oracles. Subroutine 2 describes ADGAC. It takes as input a dataset S and a sampling number k. ADGAC first runs Quicksort algorithm on S using feedback from comparison oracle, which is of form $Z(x, x')$. Given that the comparison oracle $Z(\cdot, \cdot)$ might be asymmetric w.r.t. its two arguments, i.e., $Z(x, x')$ may not equal to $Z(x', x)$, for each pair (x_i, x_j), we randomly choose (x_i, x_j) or (x_j, x_i) as the input to $Z(\cdot, \cdot)$. After Quicksort, the algorithm divides the data into multiple groups of size $\alpha m = \varepsilon|\tilde{S}|$, and does group-based binary search by sampling k labels from each group and determining the label of each group by majority vote.

For active learning algorithm $\mathcal{A}$, let $\mathcal{A}$-ADGAC be the algorithm of replacing step 4 with ADGAC using parameters $(S_i, n_i, \varepsilon_i, k_i)$, where ε_i, k_i are chosen as additional parameters of the algorithm. We establish results for specific $\mathcal{A}$: A^2 and margin-based active learning in Sections 2.5 and 2.6, respectively.

2.4.2 Theoretical Analysis of ADGAC

Before we combine ADGAC with active learning algorithms, we provide theoretical results for ADGAC. By the algorithmic procedure, ADGAC reduces the problem of labeling the whole dataset S to binary searching a threshold on the sorted list S. One can show that the conflicting instances cannot be too many within each group S_i, and thus binary search performs well in our algorithm. We also use results in [7] to give an error estimate of Quicksort. We have the following result based on the above arguments.

Subroutine 2 Active Data Generation with Adversarial Comparison (ADGAC)

Input: Dataset S with $|S| = m$, n, ε, k.

$\alpha \leftarrow \frac{\varepsilon n}{2m}$.
Define preference relation on S according to Z. Run Quicksort on S to rank elements in an increasing order. Obtain a sorted list $S = (x_1, x_2, ..., x_m)$.
Divide S into groups of size αm: $S_i = \{x_{(i-1)\alpha m+1}, ..., x_{i\alpha m}\}, i = 1, 2, ..., 1/\alpha$.
$t_{\min} \leftarrow 1, t_{\max} \leftarrow 1/\alpha$.
while $t_{\min} < t_{\max}$ **do** ▷ Do binary search
    $t = (t_{\min} + t_{\max})/2$.
    Sample k points uniformly without replacement from S_t and obtain the labels $Y = \{y_1, ..., y_k\}$.
    If $\sum_{i=1}^{k} y_i \geq 0$, **then** $t_{\max} = t$; **else** $t_{\min} = t + 1$.
end while
For $t' > t$ and $x_i \in S_{t'}$, let $\hat{y}_i \leftarrow 1$.
For $t' < t$ and $x_i \in S_{t'}$, let $\hat{y}_i \leftarrow -1$.
For $x_i \in S_t$, let $\hat{y}_i$ be the majority of labeled points in S_t.

Output: Predicted labels $\hat{y}_1, \hat{y}_2, ..., \hat{y}_m$.

Theorem 4. *Suppose that Conditions 2 and 3 hold for $\kappa \geq 1, \nu' \geq 0$, and $n = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-1} \log(1/\delta)\right)$. Assume a set $\tilde{S}$ with $|\tilde{S}| = n$ is sampled i.i.d. from $\mathcal{P}_{\mathcal{X}}$ and $S \subseteq \tilde{S}$ is an arbitrary subset of $\tilde{S}$ with $|S| = m$. There exist absolute constants C_1, C_2, C_3 such that if we run Subroutine 2 with $\varepsilon < C_1$, $\nu' \leq C_2\varepsilon^{2\kappa}\delta$, $k = k^{(1)}(\varepsilon, \delta) := C_3 \log\left(\frac{\log(1/\varepsilon)}{\delta}\right)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}$, it will output a labeling of S such that $|\{x_i \in S : \hat{y}_i \neq h^*(x_i)\}| \leq \varepsilon n$, with probability at least $1 - \delta$. The expected number of comparisons required is $\mathcal{O}(m \log m)$, and the number of sample-label pairs required is $\mathsf{SC}_{\text{label}}(\varepsilon, \delta) = \tilde{\mathcal{O}}\left(\log\left(\frac{m}{\varepsilon n}\right) \log(1/\delta)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right)$.*

Similarly, we analyze ADGAC under adversarial noise condition w.r.t. labeling oracle with $\nu = \mathcal{O}(\varepsilon)$.

Theorem 5. *Suppose that Conditions 1 and 3 hold for $\nu, \nu' \geq 0$, and $n = \Omega\left(\frac{1}{\varepsilon}\log(1/\delta)\right)$. Assume a set $\tilde{S}$ with $|\tilde{S}| = n$ is sampled i.i.d. from $\mathcal{P}_{\mathcal{X}}$ and $S \subseteq \tilde{S}$ is an arbitrary subset of $\tilde{S}$ with $|S| = m$. There exist absolute constants C_1, C_2, C_3, C_4 such that if we run Subroutine 2 with $\varepsilon < C_1$, $\nu' \leq C_2\varepsilon^2\delta$, $k = k^{(2)}(\varepsilon, \delta) := C_3 \log\left(\frac{\log(1/\varepsilon)}{\delta}\right)$, and $\nu \leq C_4\varepsilon$, it will output a labeling of S such that $|\{x_i \in S : \hat{y}_i \neq h^*(x_i)\}| \leq \varepsilon n$, with probability at least $1 - \delta$. The expected number of comparisons required is $\mathcal{O}(m \log m)$, and the number of sample-label pairs required is $\mathsf{SC}_{\text{label}}(\varepsilon, \delta) = \mathcal{O}\left(\log\left(\frac{m}{\varepsilon n}\right) \log\left(\frac{\log(1/\varepsilon)}{\delta}\right)\right)$.*

Proof Sketch. We call a pair (x_i, x_j) an *inverse pair* if $Z(x_i, x_j) = -1, h^*(x_i) = 1, h^*(x_j) = -1$, and an *anti-sort pair* if $h^*(x_i) = 1, h^*(x_j) = -1$, and $i < j$. We show that the expectation of inverse pairs is $n(n-1)\varepsilon^*$. By the results in [7] the numbers of inverse pairs and anti-sort pairs have the same expectation, and the actual number of anti-sort pairs can be bounded by Markov's inequality. Then we show that the majority label of each group must be all -1 starting from beginning the list, and changes to all 1 at some point of the list. With a careful choice of k, we

may obtain the true majority with k labels under Tsybakov noise; we will thus end up in the turning point of the list. The error is then bounded by the size of groups. See appendix for the complete proof.

Theorems 4 and 5 show that ADGAC gives a labeling of dataset with arbitrary small error using label complexity *independent* of the data size. Moreover, ADGAC is computationally efficient since it only involves binary search. These nice properties of ADGAC lead to improved query complexity when we combine ADGAC with other active learning algorithms.

2.5 A^2-ADGAC: Learning of Generic Hypothesis Class

In this section, we combine ADGAC with A^2 algorithm to learn a generic hypothesis class. We use the framework in Algorithm 1: let A^2-ADGAC be the algorithm that replaces step 4 in Algorithm 1 with ADGAC of parameters $(S, n_i, \varepsilon_i, k_i)$, where n_i, ε_i, k_i are parameters to be specified later. Under TNC, we have the following result.

Theorem 6. *Suppose that Conditions 2 and 3 hold, and $h^*(x) = sign(\eta(x) - 1/2)$. There exist global constants C_1, C_2 such that if we run A^2-ADGAC with $\varepsilon < C_1, \delta$, $\nu' \leq \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta) = C_2\varepsilon^{2\kappa}\delta$, $\varepsilon_i = 2^{-(i+2)}, n_i = \Omega\left(\frac{1}{\varepsilon_i}\left(d\log(1/\varepsilon)\right) + \left(\frac{1}{\varepsilon_i}\right)^{2\kappa-1}\log(1/\delta)\right)$, $k_i = k^{(1)}\left(\varepsilon_i, \frac{\delta}{4\log(1/\varepsilon)}\right)$ with $k^{(1)}$ specified in Theorem 4, with probability at least $1-\delta$, the algorithm will return a classifier $\hat{h}$ with $\Pr[\hat{h}(X) \neq h^*(X)] \leq \varepsilon$ with comparison and label complexity*

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\theta\log^2\left(\frac{1}{\varepsilon}\right)\log(d\theta)\left(\left(d\log\left(\frac{1}{\varepsilon}\right)\right) + \left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\log(1/\delta)\right)\right),$$

$$\mathsf{SC}_{\text{label}} = \tilde{\mathcal{O}}\left(\log\left(\frac{1}{\varepsilon}\right)\log\left(\min\left\{\frac{1}{\varepsilon}, \theta\right\}\right)\log(1/\delta)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right).$$

The dependence on $\log^2(1/\varepsilon)$ in $\mathsf{SC}_{\text{comp}}$ can be reduced to $\log(1/\varepsilon)$ under Massart noise.

We can prove a similar result for adversarial noise condition.

Theorem 7. *Suppose that Conditions 1 and 3 hold. There exist global constants C_1, C_2, C_3 such that if we run A^2-ADGAC with $\varepsilon < C_1, \delta$, $\nu' \leq \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta) = C_2\varepsilon^2\delta, \nu \leq \mathsf{Tol}_{\text{label}}(\varepsilon, \delta) = C_3\varepsilon$, $\varepsilon_i = 2^{-(i+2)}, n_i = \tilde{\Omega}\left(\frac{1}{\varepsilon_i}d\log\left(\frac{1}{\varepsilon_i}\right)\log(1/\delta)\right), k_i = k^{(2)}\left(\varepsilon_i, \frac{\delta}{4\log(1/\varepsilon)}\right)$ with $k^{(2)}$ specified in Theorem 5, with probability at least $1-\delta$, the algorithm will return a classifier $\hat{h}$ with $\Pr[\hat{h}(X) \neq h^*(X)] \leq \varepsilon$ with comparison and label complexity*

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\theta d\log(\theta d)\log\left(\frac{1}{\varepsilon_i}\right)\log(1/\delta)\right),$$

$$\mathsf{SC}_{\text{label}} = \tilde{\mathcal{O}}\left(\log\left(\frac{1}{\varepsilon}\right)\log\left(\min\left\{\frac{1}{\varepsilon}, \theta\right\}\right)\log(1/\delta)\right).$$

Proof of Theorems 6 and 7 uses Theorem 4 and Theorem 5 with standard manipulations in VC theory. Theorems 6 and 7 show that having access to even a biased comparison function can

reduce the problem of learning a classifier in high-dimensional space to that of learning a threshold classifier in one-dimensional space as the label complexity matches that of actively learning a threshold classifier. Given the fact that comparisons are usually easier to obtain, A^2-ADGAC will save a lot in practice due to its small label complexity. More importantly, we improve the total query complexity under TNC by separating the dependence on d and ε; The query complexity is now the sum of the two terms instead of the product of them. This observation shows the power of pairwise comparisons for learning classifiers. Such small label/query complexity is impossible without access to a comparison oracle, since query complexity with only labeling oracle is at least $\Omega\left(d\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right)$ and $\Omega\left(d\log\left(\frac{1}{\varepsilon}\right)\right)$ under TNC and adversarial noise conditions, respectively [86]. Our results also matches the lower bound of learning with labeling and comparison oracles up to log factors (see Section 2.7).

We note that Theorems 6 and 7 require rather small $\mathsf{Tol}_{\text{comp}}$, equal to $\mathcal{O}(\varepsilon^{2\kappa}\delta)$ and $\mathcal{O}(\varepsilon^2\delta)$, respectively. We will show in Section 2.7.3 that it is necessary to require $\mathsf{Tol}_{\text{comp}} = \mathcal{O}(\varepsilon^2)$ in order to obtain a classifier of error ε, if we restrict the use of labeling oracle to only learning a threshold function. Such restriction is able to reach the near-optimal label complexity as specified in Theorems 6 and 7.

2.6 Margin-ADGAC: Learning of Halfspaces

In this section, we combine ADGAC with margin-based active learning [15] to efficiently learn the class of halfspaces. Before proceeding, we first mention a naive idea of utilizing comparisons: we can i.i.d. sample pairs (x_1, x_2) from $\mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X}$, and use $Z(x_1, x_2)$ as the label of $x_1 - x_2$, where Z is the feedback from comparison oracle. However, this method cannot work well in our setting without additional assumption on the noise condition for the labeling $Z(x_1, x_2)$.

Before proceeding, we assume that $\mathcal{P}_\mathcal{X}$ is isotropic log-concave on $\mathbb{R}^d$; i.e., $\mathcal{P}_\mathcal{X}$ has mean 0, covariance I and the logarithm of its density function is a concave function [14, 15]. The hypothesis class of halfspaces can be represented as $\mathbb{C} = \{h : h(x) = \text{sign}(w \cdot x), w \in \mathbb{R}^d\}$. Denote by $h^*(x) = \text{sign}(w^* \cdot x)$ for some $w^* \in \mathbb{R}^d$. Define $l_\tau(w, x, y) = \max(1 - y(w \cdot x)/\tau, 0)$ and $l_\tau(w, W) = \frac{1}{|W|}\sum_{(x,y)\in W} l_\tau(w, x, y)$ as the hinge loss. The expected hinge loss of w is $L_\tau(w, D) = \mathbb{E}_{x\sim D}[l_\tau(w, x, \text{sign}(w^* \cdot x))]$.

Margin-based active learning [15] is a concrete example of Algorithm 1 by taking V as (a singleton set of) the hinge loss minimizer, while taking U as the margin region around that minimizer. More concretely, take $U_0 = \mathcal{X}$ and $V_0 = \{w_0\}$ for some w_0 such that $\theta(w_0, w^*) \le \pi/2$. The algorithm works with constants $M \ge 2, \kappa < 1/2$ and a set of parameters r_i, τ_i, b_i, z_i that equal to $\Theta(M^{-i})$ (see proof in Appendix for formal definition of these parameters). V always contains a single hypothesis. Suppose $V = \{w_{i-1}\}$ in iteration $i-1$. Let v_i satisfies $l_{\tau_i}(v_i, W) \le \min_{v:\|v-w_{i-1}\|_2\le r_i, \|v\|_2\le 1} l_{\tau_i}(v, W) + \kappa/8$, where w_i is the content of V in iteration i. We also have $f_V(V, W, i) = \{w_i\} = \left\{\frac{v_i}{\|v_i\|_2}\right\}$ and $f_U(U, V, W, i) = \{x : |w_i \cdot x| \le b_i\}$.

Let Margin-ADGAC be the algorithm obtained by replacing the sampling step in margin-based active learning with ADGAC using parameters $(S, n_i, \varepsilon_i, k_i)$, where n_i, ε_i, k_i are additional parameters to be specified later. We have the following results under TNC and adversarial noise conditions, respectively.

Theorem 8. *Suppose that Conditions 2 and 3 hold, and* $h^*(x) = \textit{sign}(w^* \cdot x) = \textit{sign}(\eta(x) - 1/2)$. *There are settings of* $M, \kappa, r_i, \tau_i, b_i, \varepsilon_i, k_i$, *and constants* C_1, C_2 *such that for all* $\varepsilon \le C_1, \nu' \le \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta) = C_2 \varepsilon^{2\kappa} \delta$, *if we run Margin-ADGAC with* w_0 *such that* $\theta(w_0, w^*) \le \pi/2$, *and* $n_i = \tilde{O}\left(\frac{1}{\varepsilon_i} d \log^3(dk/\delta) + \left(\frac{1}{\varepsilon}\right)^{2\kappa-1} \log(1/\delta)\right)$, *it finds* $\hat{w}$ *such that* $\Pr[\textit{sign}(\hat{w} \cdot X) \ne \textit{sign}(w^* \cdot X)] \le \varepsilon$ *with probability at least* $1 - \delta$. *The comparison and label complexity are*

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\log^2(1/\varepsilon)\left(d \log^4(d/\delta) + \left(\frac{1}{\varepsilon}\right)^{2\kappa-2} \log(1/\delta)\right)\right),$$

$$\mathsf{SC}_{\text{label}} = \tilde{\mathcal{O}}\left(\log(1/\varepsilon) \log(1/\delta) \left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right).$$

The dependence on $\log^2(1/\varepsilon)$ *in* $\mathsf{SC}_{\text{comp}}$ *can be reduced to* $\log(1/\varepsilon)$ *under Massart noise.*

Theorem 9. *Suppose that Conditions 1 and 3 hold. There are settings of* $M, \kappa, r_i, \tau_i, b_i, \varepsilon_i, k_i$, *and constants* C_1, C_2, C_3 *such that for all* $\varepsilon \le C_1, \nu' \le \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta) = C_2 \varepsilon^{2\kappa} \delta, \nu \le \mathsf{Tol}_{\text{comp}}(\varepsilon, \delta) = C_3 \varepsilon$, *if we run Margin-ADGAC with* $n_i = \tilde{\mathcal{O}}\left(\frac{1}{\varepsilon_i} d \log^3(dk/\delta)\right)$ *and* w_0 *such that* $\theta(w_0, w^*) \le \pi/2$, *it finds* $\hat{w}$ *such that* $\Pr[\textit{sign}(\hat{w} \cdot X) \ne \textit{sign}(w^* \cdot X)] \le \varepsilon$ *with probability at least* $1 - \delta$. *The comparison and label complexity are*

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\log(1/\varepsilon)\left(d \log^4(d/\delta)\right)\right), \quad \mathsf{SC}_{\text{label}} = \tilde{\mathcal{O}}\left(\log(1/\varepsilon) \log(1/\delta)\right).$$

The proofs of Theorems 8 and 9 are different from the conventional analysis of margin-based active learning in two aspects: a) Since we use labels generated by ADGAC, which is not independently sampled from the distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$, we require new techniques that can deal with adaptive noises; b) We improve the results of [15] over the dependence of d by new Rademacher analysis.

Theorems 8 and 9 enjoy better label and query complexity than previous results (see Table 2.2). We mention that while Yan and Zhang [195] proposed a perceptron-like algorithm with label complexity as small as $\tilde{\mathcal{O}}(d \log(1/\varepsilon))$ under Massart and adversarial noise conditions, their algorithm works only under uniform distributions over the instance space. In contrast, our algorithm Margin-ADGAC works under broad log-concave distributions. The label and total query complexity of Margin-ADGAC improves over that of traditional active learning. The lower bounds in Section 2.7 show the optimality of our complexity.

2.7 Lower Bounds

In this section, we give lower bounds on learning using labeling and pairwise comparison. In Section 2.7.1, we give a lower bound on the optimal label complexity $\mathsf{SC}_{\text{label}}$. In Section 2.7.2 we use this result to give a lower bound on the total query complexity, i.e., the sum of comparison and label complexity. Our two methods match these lower bounds up to log factors. In Section 2.7.3, we additionally give an information-theoretic bound on $\mathsf{Tol}_{\text{comp}}$, which matches our algorithms in the case of Massart and adversarial noise.

Following from [86, 87], we assume that there is an underlying score function g^* such that $h^*(x) = \text{sign}(g^*(x))$. Note that g^* does not necessarily have relation with $\eta(x)$; We only require that $g^*(x)$ represents how likely a given x is positive. For instance, in digit recognition, $g^*(x)$ represents how an image looks like a 7 (or 9); In the clinical setting, $g^*(x)$ measures the health condition of a patient. Suppose that the distribution of $g^*(X)$ is continuous, i.e., the probability density function exists and for every $t \in \mathbb{R}$, $\Pr[g^*(X) = t] = 0$.

2.7.1 Lower Bound on Label Complexity

The definition of g^* naturally induces a comparison oracle Z with $Z(x, x') = \text{sign}(g^*(x) - g^*(x'))$. We note that this oracle is invariant to shifting w.r.t. g^*, i.e., g^* and $g^* + t$ lead to the same comparison oracle. As a result, we cannot distinguish g^* from $g^* + t$ without labels. In other words, pairwise comparisons do not help in improving label complexity when we are learning a threshold function on $\mathbb{R}$, where all instances are in the natural order. So the label complexity of any algorithm is lower bounded by that of learning a threshold classifier, and we formally prove this in the following theorem.

Theorem 10. *For any algorithm $\mathcal{A}$ that can access both labeling and comparison oracles, sufficiently small ε, δ, and any score function g that takes at least two values on $\mathcal{X}$, there exists a distribution $P_{\mathcal{XY}}$ satisfying Condition 2 such that the optimal function is in the form of $h^*(x) = \text{sign}(g(x) + t)$ for some $t \in \mathbb{R}$ and*

$$\mathsf{SC}_{\text{label}}(\varepsilon, \delta, \mathcal{A}) = \Omega\left((1/\varepsilon)^{2\kappa-2} \log(1/\delta)\right). \tag{2.1}$$

If $\mathcal{P}_{\mathcal{XY}}$ satisfies Condition 1 with $\nu = O(\varepsilon)$, $\mathsf{SC}_{\text{label}}$ satisfies Equation (2.1) with $\kappa = 1$.

The lower bound in Theorem 10 matches the label complexity of A^2-ADGAC and Margin-ADGAC up to a log factor. So our algorithm is near-optimal.

2.7.2 Lower Bound on Total Query Complexity

We use Theorem 10 to give lower bounds on the total query complexity of any algorithm which can access both comparison and labeling oracles.

Theorem 11. *For any algorithm $\mathcal{A}$ that can access both labeling and comparison oracles, and sufficiently small ε, δ, there exists a distribution $P_{\mathcal{XY}}$ satisfying Condition 2, such that*

$$\mathsf{SC}_{\text{comp}}(\varepsilon, \delta, \mathcal{A}) + \mathsf{SC}_{\text{label}}(\varepsilon, \delta, \mathcal{A}) = \Omega\left((1/\varepsilon)^{2\kappa-2} \log(1/\delta) + d \log(1/\varepsilon)\right). \tag{2.2}$$

If $\mathcal{P}_{\mathcal{XY}}$ satisfies Condition 1 with $\nu = O(\varepsilon)$, $\mathsf{SC}_{\text{comp}} + \mathsf{SC}_{\text{label}}$ satisfies Equation (2.2) with $\kappa = 1$.

The first term of Equation (2.2) follows from Theorem 10, whereas the second term follows from transforming a lower bound of active learning with access to only the labeling oracle. The lower bounds in Theorem 11 match the performance of A^2-ADGAC and Margin-ADGAC up to log factors.

2.7.3 Adversarial Noise Tolerance of Comparisons

Note that label queries are typically expensive in practice. Thus it is natural to ask the following question: what is the minimal requirement on ν', given that we are only allowed to have access to minimal label complexity as in Theorem 10? We study this problem in this section. More concretely, we study the requirement on ν' when we learn a threshold function using labels. Suppose that the comparison oracle gives feedback using a scoring function $\hat{g}$, i.e., $Z(x, x') = \text{sign}(\hat{g}(x) - \hat{g}(x'))$, and has error ν'. We give a sharp minimax bound on the risk of the optimal classifier in the form of $h(x) = \text{sign}(\hat{g}(x) - t)$ for some $t \in \mathbb{R}$ below.

Theorem 12. *Suppose that* $\min\{\Pr[h^*(X) = 1], \Pr[h^*(X) = -1]\} \geq \sqrt{\nu'}$ *and both* $\hat{g}(X)$ *and* $g^*(X)$ *have probability density functions. If* $\hat{g}(X)$ *induces an oracle with error* ν'*, then we have* $\min_t \max_{\hat{g}, g^*} \Pr[\textit{sign}(\hat{g}(X) - t) \neq h^*(X)] = \sqrt{\nu'}$.

The proof is technical and omitted. By Theorem 12, we see that the condition of $\nu' = \varepsilon^2$ is necessary if labels from g^* are only used to learn a threshold on $\hat{g}$. This matches our choice of ν' under Massart and adversarial noise conditions for labeling oracle (up to a factor of δ).

2.8 Conclusion

We presented a general algorithmic framework, ADGAC, for learning with both comparison and labeling oracles. We proposed two variants of the base algorithm, A^2-ADGAC and Margin-ADGAC, to facilitate low query complexity under Tsybakov and adversarial noise conditions. The performance of our algorithms matches lower bounds for learning with both oracles. Our analysis is relevant to a wide range of practical applications where it is easier, less expensive, and/or less risky to obtain pairwise comparisons than labels.

There are multiple directions for future works. One improvement over our work is to show complexity bounds for excess risk $\mathsf{err}(h) - \mathsf{err}(h^*)$ instead of $\Pr[h \neq h^*]$. Also, our bound on comparison complexity is in expectation due to limits of quicksort; deriving concentration inequalities on the comparison complexity would be helpful. Also, an adaptive algorithm that applies to different levels of noise w.r.t. labels and comparisons would be interesting; i.e., use labels when comparisons are noisy and use comparisons when labels are noisy. Other directions include using comparisons (or more broadly, rankings) for other ML tasks like regression or matrix completion.

2.9 Proofs

2.9.1 Proof of Theorem 4

Proof. We only prove the theorem for $\kappa > 1$, the case of $\kappa = 1$ holds with a similar proof. An equivalent condition (see e.g., page 341, [37]) for Condition 2 under $\kappa > 1$ is that there exists constant $\tilde{\mu} > 0$ such that for all $t > 0$ we have

$$\Pr(|\eta(x) - 1/2| < t) \leq \tilde{\mu} t^{1/(\kappa - 1)}. \tag{2.3}$$

We use Equation (2.3) instead of Condition 2 through out the proof.

To bound the error in labeling by ADGAC, we first bound the number of incorrectly sorted pairs due to noise/bias of the comparison oracle. We call (x_i, x_j) an *inverse pair* if $h^*(x_i) = 1, h^*(x_j) = -1, x_i \preccurlyeq x_j$ (the partial order is decided by randomly querying $Z(x_i, x_j)$ or $Z(x_j, x_i)$). Also, we call (x_i, x_j) an *anti-sort pair* if $h^*(x_i) = 1, h^*(x_j) = -1, i < j$ (after sorting by Quicksort). Let T be the set of all anti-sort pairs, T' be the set of all inverse pairs in S, and $\tilde{T}'$ be the set of all inverse pairs in $\tilde{S}$. We first bound $|T|$ using $|T'|$. Let s be the random bits supplied for Quicksort in its process, by Theorem 3 in [7] we have

$$\mathbb{E}_s[|T|] = |T'|.$$

Notice that sampling a pair of (X, X') is equivalent to sample a set $\tilde{S}$ of n points and then uniformly pick two different points in it. Also, number of inverse pairs in S is less than that in $\tilde{S}$. So we have

$$\mathbb{E}_S\left[\mathbb{E}_s[|T|]\right] = \mathbb{E}_S[|T'|] \leq \mathbb{E}_{\tilde{S}}[|\tilde{T}'|] = n(n-1)\nu' \leq n^2\nu'.$$

By Markov inequality we have

$$\Pr\left(|T| \geq \frac{2\nu'}{\delta}n^2\right) \leq \frac{\delta}{2}. \tag{2.4}$$

Suppose $|T| < \frac{2\nu'}{\delta}n^2$ (which holds with probability $> 1 - \delta/2$). We now proceed to bound the number of labeling errors made by ADGAC. First, notice that in Algorithm 2, we divide all samples into groups of size $\alpha m = \varepsilon n/2$. For every set S_i, let

$$\begin{aligned} q(S_i) &= \frac{1}{|S_i|}\min\left\{\sum_{x\in S_i} I(h^*(x) = 1), \sum_{x\in S_i} I(h^*(x) = -1)\right\} \\ &= \min\left\{\Pr_{X\sim S_i}(h^*(x) = -1), \Pr_{X\sim S_i}(h^*(x) = 1)\right\} \end{aligned}$$

where $X \sim S_i$ denote the empirical distribution that X is drawn uniformly at random from the finite collection of points in S_i. Let

$$\beta = \frac{2}{\varepsilon}\sqrt{\frac{\nu'}{\delta}} \leq C\varepsilon^{\kappa-1}$$

for some constant C. Suppose ε is small enough such that $\beta \leq 1/2$. Then we claim that there is at most 1 set S_i such that $q(S_i) \geq \beta$. Otherwise, suppose two such sets exist; let them be S_i and S_j. So there are at least $\alpha\beta m$ points $x \in S_i$ with $h^*(x) = -1$, and $\alpha\beta m$ points $x \in S_i$ with $h^*(x) = 1$; the same holds for S_j. These -1s and 1s would indicate at least

$$2\alpha^2\beta^2 m^2 = \frac{2\nu'}{\delta}n^2$$

anti-sort pairs, which violates our claim of $|T|$.

Since ADGAC uses group binary search, we first analyze some properties of the majority label of the Bayes optimal classifier within each group/set. For each set S_i, let $\mu(S_i) =$

sign$(\sum_{x\in S_i} h^*(x_i))$ be the majority Bayes optimal label. We can show that $\mu(S_i)$ is monotonic: that is, for every $i < j$ we have $\mu(S_i) \le \mu(S_j)$. To see this, suppose there exist two sets $S_i, S_j, i < j$ such that $\mu(S_i) = 1$ and $\mu(S_j) = -1$. That would indicate at least $\alpha^2 m^2/2 > \alpha^2\beta^2 m^2$ anti-sort pairs, which violates our assumption. So there must be a boundary l such that $\mu(S_i) = -1$ for $i < l$, and $\mu(S_i) = 1$ for $i \ge l$. We call S_l to be the *boundary set*. Now consider two cases:

- Case 1: there exists a set $S_{l'}$ such that $q(S_{l'}) \ge \beta$ (recall that from previous arguments, there is only one such set). If $l \ne l'$, the sets S_l and $S_{l'}$ generates at least $2(\alpha m/2)(\alpha\beta m) \ge 2\alpha^2\beta^2 m^2$ anti-sort pairs, which violates our assumption for $|T|$. So $l = l'$.
- Case 2: for all sets S_i, we have $q(S_i) < \beta$.

In both cases, we have $q(S_i) < \beta$ for all $i \ne l$.

Now we prove that the majority vote of the noisy labels agrees with the majority vote of the Bayes optimal classifier $\mu(S_i)$ for each set S_i that we visit, and hence we will find the boundary set S_l. Suppose $q(S_i) < \beta$. Take

$$t = \left(\frac{\varepsilon}{16\tilde{\mu}}\right)^{\kappa-1}.$$

For small enough ε, we have $t \le 1/2$, and $\Pr(x : |\eta(x) - 1/2| \le t) \le \varepsilon/16$. Let $U = \{x_i \in S : |\eta(x_i) - 1/2| \le t\}$. By relative form of Chernoff bound we have

$$\Pr\left(|U| > 3\log(4/\delta) + n\varepsilon/8\right) \le \exp\left(-\frac{3\log(4/\delta) + \varepsilon n/16}{3}\right) \le \frac{\delta}{4}.$$

Suppose $|U|/n \le \varepsilon/8$, so at most 1/4 of each S_i is in U.

For each set S_i, let $\bar{S}_i = \{x \in S_i : h^*(x) \ne \mu(S_i)\}$ and $S_i' = \{x \in S_i : |\eta(x) - 1/2| \le t\}$. So for each set such that $q(S_i) \le \beta$, we have

$$\begin{aligned}
\Pr(Y \ne \mu(S_i)|X \sim S_i) \le& \Pr(Y \ne \mu(S_i)|X \in S_i')\Pr(X \in S_i'|X \sim S_i) + \\
&\Pr(Y \ne \mu(S_i)|X \in \bar{S}_i)\Pr(X \in \bar{S}_i|X \sim S_i) + \\
&\Pr(Y \ne \mu(S_i)|X \notin S_i', X \notin \bar{S}_i)\Pr(X \notin S_i', X \notin \bar{S}_i|X \sim S_i) \\
\le& \left(\frac{1}{2} + t\right)\cdot\frac{1}{4} + 1\cdot\beta + \left(\frac{1}{2} - t\right)\cdot\left(\frac{3}{4} - \beta\right) \\
=& \frac{1}{2} - \frac{1}{2}t + \left(\frac{1}{2} + t\right)\beta.
\end{aligned}$$

Pick ν' small enough such that $\beta \le \frac{1}{4}t$:

$$\frac{2}{\varepsilon}\sqrt{\frac{\nu'}{\delta}} \le \frac{1}{4}\left(\frac{\varepsilon}{16\tilde{\mu}}\right)^{\kappa-1}.$$

This yields

$$\nu' \le \frac{\varepsilon^{2\kappa}\delta}{32(16\tilde{\mu})^{2\kappa-2}}.$$

Note that this also guarantees $\beta \leq 1/2$ above, since $t \leq \frac{1}{2}$. Now we have

$$\Pr\left[Y \neq \mu(S_i) | X \sim S_i\right] \leq \frac{1}{2} - \frac{1}{2}t + \frac{1}{4}t\left(t + 1/2\right) \leq \frac{1}{2} - \frac{1}{4}t.$$

In the algorithm, suppose we pick $X_1, X_2, ..., X_k \in S_i$ as the points for which to query the label and the labels are $Y_1, ..., Y_k$. Note that since $n = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{2\kappa-1}\log(1/\delta)\right)$, we have $|S_i| = \alpha n \geq k$ for every i, so we will not run out of samples for each set. By Hoeffding's inequality, we have

$$\Pr\left[\text{sign}\left(\sum_{j=1}^{k} Y_j\right) = \mu(S_i)\right] = \Pr\left[\frac{1}{k}\sum_{j=1}^{n} I(Y_j = \mu(S_i)) > \frac{1}{2}\right] \leq \exp\left(-\frac{1}{8}kt^2\right).$$

The choice of k yields that the majority vote of the noisy labels agrees with the majority vote $\mu(S_i)$ of the Bayes optimal classifier for each S_i with $q(S_i) \leq \beta$ we visit, with probability $\frac{\delta}{8\log(2/\varepsilon)}$.

Suppose the binary search output set S_t (i.e., the value of t at step 10). Now we analyze the errors we made in the final output. We consider the two cases:

- Case 1: If $q(S_l) \geq \beta$, then with probability $1 - \delta$, we have $t \in \{l-1, l, l+1\}$ since we might behave arbitrarily in set S_l. In this case, we have $q(S_l)|S_l| \geq \alpha\beta m$, and so $|\{x : x \in S_{t'}, t' < l, h^*(x) = 1\} \leq \alpha\beta m$, because otherwise we have $\alpha^2\beta^2m^2$ anti-sort pairs, which violates our assumption on $|T|$. Similarly, $|\{x : x \in S_{t'}, t' > l, h^*(x) = -1\}| \leq \alpha\beta m$. Counting also the possible errors made on S_l, the total number of errors is

$$|\{\hat{y}_i : \hat{y}_i \neq h^*(x_i)\}| \leq \alpha m + 2\alpha\beta m \leq \frac{\varepsilon n}{2}\left(1 + \frac{1}{2}t\right) \leq \varepsilon n.$$

- Case 2: If $q(S_i) < \beta$ for all i, then we have $t \in \{l-1, l\}$. Now note that we have $|\{x \in S_{l-1} : h^*(x) = -1\}| \geq \alpha m/2$, and so $|\{x \in S_{t'} : t' < l-1, h^*(x) = 1\}| \leq \alpha\beta m$ since otherwise at least $\alpha^2\beta m/2$ anti-sort pairs are present. So $|\{x \in S_{t'} : t' \leq l-1, h^*(x) = 1\}| \leq 2\alpha\beta m$ considering $q(S_{l-1}) < \beta$. Similarly, $|\{x \in S_{t'} : t' \geq l, h^*(x) = -1\}| \leq 2\alpha\beta m$. So the total number of errors is

$$|\{\hat{y}_i : \hat{y}_i \neq h^*(x_i)\}| \leq 4\alpha\beta m \leq \varepsilon n.$$

So we have at most εn error under both cases. Now we examine the total query complexity: The expected comparison complexity is $O(m \log m)$ even if we have noisy comparisons, see [7]. For label complexity, it takes $k = \tilde{\mathcal{O}}\left(\log(1/\delta)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right)$ queries for each set S_t, and we do this for $\mathcal{O}(\log(1/\alpha)) = \mathcal{O}\left(\log\left(\frac{2m}{\varepsilon n}\right)\right)$ times. So the total query complexity is

$$\tilde{\mathcal{O}}\left(\log\left(\frac{2m}{\varepsilon n}\right)\log(1/\delta)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right).$$

□

2.9.2 Proof of Theorem 5

Proof. The first part of proof is exactly the same as that of Theorem 4. We now bound $\Pr[Y \neq \mu(S_i)|X \sim S_i]$. Suppose $q(S_i) < \beta$. Let $V = \{x : \Pr[Y \neq h^*(X)|X = x] > 1/4\}$ and $U = \{x_i : \Pr[Y \neq h^*(X)|X = x_i] > 1/4\}$. We have $P(V) \leq 4\nu$. By a relative Chernoff bound, if $\nu \leq C_1\varepsilon$ for a small enough constant C_1 we have

$$\Pr[|U| \leq 8n\nu + 3\log(4/\delta)] \leq \exp\left(-\frac{3\log(4/\delta) + 4\nu n}{3}\right) \leq \delta/4.$$

So if $\nu \leq \frac{1}{64}\varepsilon$ we have $|U|/n \leq \varepsilon/8$ with probability $\delta/4$. In this case, at most 1/4 of each S_i is in U.

For each set S_i, let $\bar{S}_i = \{x \in S_i : h^*(x) \neq \mu(S_i)\}$ and $\tilde{S}_i = \{x \in S_i, x \in U\}$. So for each set such that $q(S_i) \leq \beta$, we have

$$\begin{aligned}
\Pr(Y \neq \mu(S_i)|X \sim S_i) \leq & \Pr(Y \neq \mu(S_i)|X \in \tilde{S}_i)\Pr(X \in \tilde{S}_i|X \sim S_i) + \\
& \Pr(Y \neq \mu(S_i)|X \in \bar{S}_i)\Pr(X \in \bar{S}_i|X \sim S_i) + \\
& \Pr(Y \neq \mu(S_i)|X \notin \tilde{S}_i, X \notin \bar{S}_i)\Pr(X \notin \tilde{S}_i, X \notin \bar{S}_i|X \sim S_i) \\
\leq & 1 \cdot \frac{1}{4} + 1 \cdot \beta + \left(\frac{3}{4} - \beta\right)\frac{1}{4} \\
= & \frac{7}{16} + \frac{3}{4}\beta.
\end{aligned}$$

So there exists constant C_2 such that if $\nu' \leq C_2\varepsilon^2\delta$ we have $\beta \leq \frac{1}{24}$, $\Pr(Y \neq \mu(S_i)|X \sim S_i)] \leq \frac{1}{2} - \frac{1}{32}$. Thus by Hoeffding's inequality, the choice of k yields that we recover $\mu(S_i)$ for each i we visit with probability $\frac{\delta}{8\log(2/\varepsilon)}$.

By similar analysis as the proof of Theorem 4, we can show that the number of errors (i.e., $|\{\hat{y}_i : \hat{y}_i \neq h^*(x_i)\}|$) is at most εn.

Now examine the total query complexity: It takes $k = \mathcal{O}\left(\log(\log(1/\varepsilon)/\delta)\right)$ queries for each set S_t, and we do this for $\mathcal{O}(\log(1/\alpha)) = \mathcal{O}(\log\left(\frac{2m}{\varepsilon n}\right))$ times. So the total query complexity is

$$\mathcal{O}\left(\log\left(\frac{2m}{\varepsilon n}\right)\log\left(\frac{\log(1/\varepsilon)}{\delta}\right)\right).$$

□

2.9.3 Proof for A^2-ADGAC

We use the following lemma adapted from [86]:

Lemma 13 ([86], Lemma 3.1)**.** *Suppose that $\mathcal{D} = \{x_1, x_2, ..., x_n\}$ is i.i.d. sampled from $\mathcal{P}_\mathcal{X}$, and $h^* \in \mathbb{C}$. There is a universal constant $c_0 \in (1, \infty)$ such that for any $\gamma \in (0, 1)$, and any $n \in \mathbb{N}$, letting*

$$U(n, \gamma) = c_0\frac{d\log(n/d) + \log(1/\gamma)}{n},$$

with probability at least $1-\gamma$, $\forall h \in \mathbb{C}$, *the following inequalities hold:*

$$\Pr_{X\sim\mathcal{P}_\mathcal{X}}[h(X)\neq h^*(X)] \leq \max\{2\Pr_{X\sim\mathcal{D}}[h(X)\neq h^*(X)], U(n,\gamma)\},$$
$$\Pr_{X\sim\mathcal{D}}[h(X)\neq h^*(X)] \leq \max\{2\Pr_{X\sim\mathcal{P}_\mathcal{X}}[h(X)\neq h^*(X)], U(n,\gamma)\}.$$

Here $X\sim\mathcal{D}$ *means* X *is uniformly sampled from finite set* $\mathcal{D}$.

Algorithm 3 A^2-ADGAC

Input: $n_i, \mathbb{C}, \varepsilon, \delta$, comparison oracle f.

1: Let $V \leftarrow \mathbb{C}$.
2: **for** $i = 1, 2, ..., \lceil\log(1/\varepsilon)\rceil$ **do**
3: Sample dataset $\tilde{S}$ of size n_i.
4: Let $S \leftarrow \{x \in \tilde{S} : x \in \mathsf{DIS}(V)\}$.
5: Run ADGAC (Subroutine 2) with $S, \tilde{S}, \varepsilon_i = 2^{-(i+2)}$, k_i and labeled dataset W.
6: $V = V \setminus \{h : |W|\mathsf{err}_W(h) \geq n_i\varepsilon_i\}$.
7: **end for**

Output: Any Classifier $\hat{h} \in V$.

Proof of Theorem 6. For a labeled dataset $W = \{(x_i, \hat{y}_i)\}_{i=1}^n$, let $\mathsf{err}_W(h) = \frac{1}{n}\sum_{i=1}^n I(h(x_i) \neq \hat{y}_i)$ be the empirical risk of h on W for any $h \in \mathbb{C}$ (remind that $\hat{y}_i$ are predictions of ADGAC). For a clearer explanation, we formalize the A^2-ADGAC algorithm in Algorithm 3. We use induction to prove that after iteration i we have $\Pr[h(X) \neq h^*(X)] \leq 4\varepsilon_i$ for all $h \in V$ after step 6 in Algorithm 3. This proposition holds for $i = 0$. Suppose it holds for $i-1$. By Theorem 4 and a union bound, with probability $1-\delta/4$, for every iteration i we have at most $n_i\varepsilon_i$ errors with respect to h^* after running ADGAC, i.e., $|W|\mathsf{err}_W(h^*) \leq n_i\varepsilon_i$. So h^* will not be eliminated from V in any iteration with probability $1-\delta/4$. On the other hand, notice that by Step 6 in Algorithm 3 all functions $h \in V$ satisfies $|W|\mathsf{err}_W(h) \leq n_i\varepsilon_i$, so by triangle inequality we have (notice that W is just the set S with labels)

$$\begin{aligned}|S|\Pr_{X\sim S}[h(X)\neq h^*(X)] &= |\{x\in S : h(x)\neq h^*(x)\}| \\ &\leq |\{(x,\hat{y})\in W : h(x)\neq\hat{y}\}| + |\{(x,\hat{y})\in W : h^*(x)\neq\hat{y}\}| \\ &\leq 2\varepsilon_i n_i.\end{aligned}$$

Also note that functions in V agrees on $\tilde{S}\setminus S$; so $|\tilde{S}|\Pr_{X\sim\tilde{S}}[h(X)\neq h^*(X)] \leq 2\varepsilon_i n_i$, and since $|\tilde{S}| = n_i$ we have $\Pr_{X\sim\tilde{S}}[h(X)\neq h^*(X)] \leq 2\varepsilon_i$. Now using Lemma 13 with $n = n_i$, we have $\Pr_{X\sim\mathcal{P}_\mathcal{X}}[h(x)\neq h^*(x)] \leq 4\varepsilon_i$ for every $h \in V$ by choosing n_i such that $U\left(n_i, \frac{\delta}{4\log(1/\varepsilon)}\right) \leq \varepsilon_i$. So at the end of the algorithm it outputs a classifier with $\Pr[\hat{h} \neq h^*] \leq \varepsilon$.

Now we examine the number of queries. By definition of disagreement coefficient, at round i we have $\mathsf{DIS}(V) \leq \theta\varepsilon_i$; thus using a relative Chernoff bound we know that with probability $1-\delta/4$ we have

$$m_i \coloneqq |S| \leq \log(12/\delta) + 2n_i\theta\varepsilon_i = \mathcal{O}\left(\theta\left((d\log(1/\varepsilon)) + \left(\frac{1}{\varepsilon_i}\right)^{2\kappa-2}\log(1/\delta)\right)\right).$$

It takes $O(m_i \log m_i)$ comparisons in expectation to rank the set, and there are $\log(1/\varepsilon)$ iterations. So the total comparison complexity is

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}]=\tilde{\mathcal{O}}\left(\theta \log\left(\frac{1}{\varepsilon}\right)\left(\log d\theta + (\kappa-1)\log\left(\frac{1}{\varepsilon}\right)\right)\left(\left(d\log\left(\frac{1}{\varepsilon}\right)\right)+\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\log(1/\delta)\right)\right).$$

This obtained the stated comparison complexity. The label complexity follows by multiplying the label complexity of ADGAC by $\log(1/\varepsilon)$. Note that in every step we have $\frac{m_i}{\varepsilon_i n_i} = O\left(\min\left\{\theta, \frac{1}{\varepsilon}\right\}\right)$. □

Proof of Theorem 7. With the same proof, A^2-ADGAC outputs a classifier with $\Pr[\hat{h} \neq h^*] \leq \varepsilon$ upon finishing. We know examine the number of queries. By definition of disagreement coefficient, at round i we have $\mathsf{DIS}(V) \leq \theta\varepsilon_i$; thus using a Chernoff bound we know that with probability $1-\delta/4$ we have

$$m_i \coloneqq |S| \leq \log(12/\delta) + 2n_i\theta\varepsilon_i = \mathcal{O}\left(\theta d \log\left(\frac{1}{\varepsilon_i}\right)\log\left(\frac{1}{\delta}\right)\right).$$

It takes $O(m_i \log m_i)$ comparisons in expectation to rank the set, and there are $\log(1/\varepsilon)$ iterations. So the total comparison complexity is

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\theta d \log(\theta d)\log\left(\frac{1}{\varepsilon_i}\right)\log\left(\frac{1}{\delta}\right)\right).$$

This obtained the stated comparison complexity. The label complexity follows by multiplying the label complexity of ADGAC by $\log(1/\varepsilon)$. Note that in every step we have $\frac{m_i}{\varepsilon_i n_i} = O\left(\min\left\{\theta, \frac{1}{\varepsilon}\right\}\right)$. □

2.9.4 Proof for Margin-ADGAC

We first prove Theorem 8, and Theorem 9 follows exactly the same proof with $\kappa = 1$ and using Theorem 5. For clearer explanation, we re-illustrate Margin-ADGAC in a form similar to that in [15] in Algorithm 4. The proof mostly follows that of [15]. We give a refined sample complexity via Rademacher complexity following the ideas in [195], and also change the proof according to the properties of ADGAC (note that we are not using independent samples by replace the sampling step with ADGAC).

To simplify notations, let $\mathsf{err}(w)$ be $\mathsf{err}(h_w(x)) = \mathsf{err}(\text{sign}(w \cdot x))$. Define $\Delta_D(w, w') = \Pr_{X\sim D}[\text{sign}(w \cdot X) \neq \text{sign}(w' \cdot X)]$. Also, let $\theta(w_1, w_2)$ be the angle between two vectors w_1, w_2. Let $D_{w,\gamma} = \{x : |w \cdot x| \leq \gamma\}$.

The key step is to prove the following theorem:

Theorem 14. *For $k \leq \log(1/\varepsilon)$, if $\Delta_{\mathcal{P}_\mathcal{X}}(w_{k-1}, w^*) \leq M^{-(k-1)}$, with probability $1 - \frac{\delta}{k+k^2}$, after round k of Margin-ADGAC we have $\Delta_{D_{w_{k-1},b_{k-1}}}(w_k, w^*) \leq \kappa$.*

To prove the theorem, we first list useful properties of isotropic log-concave distributions and fix the parameters we use for the algorithm. We use exactly the same parameters for r_i, τ_i, b_i, z_i as in [15], and we restate them here for completeness.

Algorithm 4 Margin-ADGAC: Efficiently learning halfspaces with comparison

Input: ε, δ, target errors ε_k, sample sizes n_k, sequences r_k, b_k, τ_k, precision value κ.

1: Draw n_1 unlabeled samples to S and run ADGAC with $\left(S, n_1, \varepsilon_0, \frac{\delta}{8\log(1/\varepsilon)}, k_1\left(\varepsilon_0, \frac{\delta}{8\log(1/\varepsilon)}\right)\right)$, and obtain a labeled dataset W.

2: **for** $k = 1, 2, ..., s = \lceil \log(4/\varepsilon) \rceil$ **do**

3: Find $v_k \in B(w_{k-1}, r_k)$ that approximately minimize training hinge loss over W, with $\|v_k\|_2 \leq 1$:

$$l_{\tau_k}(v_k, W) \leq \min_{w \in B(w_{k-1}, r_k) \cap B(0,1)} l_{\tau_k}(w, W) + \kappa/8.$$

4: $w_k \leftarrow \frac{v_k}{\|v_k\|_2}$.

5: Sample another dataset $\tilde{S}$ of n_k unlabeled samples.

6: $S = \{x \in S : |w_k \cdot x| \leq b_k\}$.

7: Run ADGAC with $\left(S, n_k, \varepsilon_k, \frac{\delta}{8\log(1/\varepsilon)}, k^{(1)}\left(\varepsilon_k, \frac{\delta}{8\log(1/\varepsilon)}\right)\right)$ and obtain labeled dataset W.

8: **end for**

Output: Return w_s.

Lemma 15 ([15, 24, 121])**.** *Suppose $X \sim \mathcal{P}_{\mathcal{X}}$ is a isotropic log-concave distribution in $\mathbb{R}^d$ with probability density function f. Then*

(a) There is an absolute constant c_1 such that, if $d = 1$, $f(x) > c_1$ for all $x \in [-1/9, 1/9]$.

(b) There is an absolute constant c_2 such that for any two unit vectors u and v in R^d we have $c_2\theta(u, v) \leq \Delta_{\mathcal{P}_{\mathcal{X}}}(u, v)$.

(c) There exists constant c_3 such that for any unit vector w and $\gamma > 0$, $\Pr[|w \cdot X| \leq \gamma] \leq c_3\gamma$.

(d) There is a constant c_4 such that for any unit vector u, all $0 < \gamma < 1$, for all a such that $\|u - a\|_2 \leq \gamma$ and $\|a\|_2 \leq 1$, $\mathbb{E}_{X \sim D_{u,\gamma}}[(a \cdot X)^2] \leq c_4(r^2 + \gamma^2)$.

(e) For any $c_5 > 0$, there is a constant $c_6 > 0$ such that the following holds: let u and v be two unit vectors in $\mathbb{R}^d$, and assume that $\theta(u, v) = \alpha < \pi/2$. Then $\Pr_{X \sim \mathcal{P}_{\mathcal{X}}}[$sign$(u \cdot X) \neq$ sign$(v \cdot X)$ and $|v \cdot X| \leq c_6\alpha] \leq c_5\alpha$.

Now we give the settings of parameters. Let $M = \max\{\frac{2}{c_2\pi}, 2\}$. Let c_1' be the value of c_6 in Lemma 15 corresponding to the case where c_5 is $\frac{c_2}{4M}$; let $b_k = c_1'M^{-k}$. Let $r_k = \min\{M^{-(k-1)}/c_2, \pi/2\}$ and $\kappa = \frac{1}{4c_1'M}$. Let $\tau_k = \frac{c_1 \min\{b_{k-1}, 1/9\}\kappa}{6}$, and $z_k^2 = r_k^2 + b_{k-1}^2$. Let $\varepsilon_k = \frac{c_3\tau_k^2 b_k \kappa^2}{256 c_4 z_k^2}$, and $n_k = O\left(\frac{1}{b_k} d \log^3\left(\frac{dk}{1/\delta}\right)\right)$. Also let $m_k = 2c_3 b_k n_k + \log(12k/\delta)$.

Then we prove the following lemma:

Lemma 16. *Suppose $|W| \geq m_k$. Let $c(W)$ be the set with truthful labels w.r.t. w^*, i.e., $c(W) = \{(x,$ sign$(w^* \cdot x)) : x \in W\}$. For any $w \in B(w_{k-1}, r_k)$, with probability $1 - \frac{\delta}{3(k+k^2)}$ we have*

$$|l(w, W) - l(w, c(W))| \leq \kappa/8.$$

Proof. Let $N = \{(x, \hat{y}) \in W : \hat{y} \neq \text{sign}(w^* \cdot x)\}$ be the set where ADGAC has x's label different

than $\text{sign}(w^* \cdot x)$ (remind that $\hat{y}$ is the prediction of ADGAC). We have

$$\begin{aligned} l(w, W) &= \frac{1}{|W|} \sum_{(x,\hat{y}) \in W} l_{\tau_k}(w, x, \hat{y}) \\ &= \frac{1}{|W|} \left(\sum_{(x,y) \notin N} l_{\tau_k}(w, x, \text{sign}(w^* \cdot x)) + \sum_{(x,y) \in N} l_{\tau_k}(w, x, -\text{sign}(w^* \cdot x)) \right). \end{aligned}$$

So

$$\begin{aligned} |l(w, W) - l(w, c(W))| &\leq \frac{1}{\tau_k |W|} \sum_{x \in N} 2(w \cdot x) \\ &\leq \frac{1}{\tau_k |W|} \sum_{x \in W} I(x \in N) 2(w \cdot x). \end{aligned} \tag{2.5}$$

We use the following lemma from [15]:

Lemma 17 (Lemma D.4, [15]). *For an absolute constant c, with probability $1 - \frac{\delta}{6(k+k^2)}$,*

$$\max_{x \in W} \|x\|_2 \leq c\sqrt{d} \log \left(\frac{|W|k}{\delta} \right).$$

Note that

$$|w \cdot x| \leq |w_{k-1} \cdot x| + |(w - w_{k-1}) \cdot x| \leq b_k + r_k \|x\|_2.$$

So with probability $1 - \frac{\delta}{6(k+k^2)}$, an event E_δ happens such that

$$\frac{|w \cdot x|}{\tau_k} \leq c' \sqrt{d} \log \left(\frac{|W|k}{\delta} \right)$$

for all $x \in W$, for some constant c'.

Notice that $\frac{|N|}{|W|} \leq \frac{\varepsilon_k n_k}{m_k}$. Let N' be a $\frac{\varepsilon_k n_k}{m_k}$ fraction of W with the largest values of $|w \cdot x|$. Let $\varphi(W) = \sum_{x \in N'} |w \cdot x|$. So by Equation (2.5) we have $|l(w, W) - l(w, c(W))| \leq \frac{2}{\tau_k |W|} \varphi(W)$. Now we have

$$\begin{aligned} \mathbb{E}[\varphi(W)] &= \mathbb{E}\left[\sum_{x \in W} \delta(x \in N') |w \cdot x| \right] \\ &\leq \sqrt{\frac{|N'|}{|W|}} \mathbb{E}\left[\sqrt{\sum_{x \in W} (w \cdot x)^2} \right] \\ &\leq \sqrt{\frac{\varepsilon_k n_k}{m_k}} \sqrt{\mathbb{E}\left[\sum_{x \in W} (w \cdot x)^2 \right]} \\ &\leq \sqrt{\frac{\varepsilon_k n_k}{m_k}} \sqrt{c_4} z_k |W| \leq \kappa \tau_k |W|/16. \end{aligned}$$

The first inequality is by Cauchy-Schwartz inequality; the second is by Jensen's inequality; the third inequality is by property (d) in Lemma 15; the last inequality is by the value of ε_i. If we condition $\mathcal{P}_{\mathcal{X}}$ on E_δ, the above expectation will be smaller since we bound $|w \cdot x|$ from above. Now by Mcdiarmid's inequality, $\frac{1}{|W|}\varphi(W)$ deviates by at most $\frac{c'\sqrt{d}\log\left(\frac{|W|k}{\delta}\right)}{|W|}$ when we change a single value of $w \cdot x$ for some $x \in W$. So by McDiarmid's inequality, using $|W| \geq m_k = \Omega(d\log^2(d/\delta))$, with probability $1 - \frac{\delta}{3(k+k^2)}$ we have

$$|l(w, W) - l(w, c(W))| \leq \mathbb{E}[\varphi(W)|E_\delta] + \kappa/16 \leq \kappa/8.$$

□

The other lemma is about bounding the difference between $l(w, c(W))$ and $E_W[l(w, c(W))]$. We improve the results in [15] using Rademacher complexity as below.

Lemma 18. *With probability* $1 - \frac{\delta}{6(k+k^2)}$ *we have*

$$|\mathbb{E}_W[l(w, x, \textit{sign}(w^* \cdot x))] - l(w, W)| \leq \kappa/16.$$

Proof. Note that every $x \in W$ is sampled independently from $D_{w_k, b_{k-1}}$. Following the same proof as in Lemma 16, an Event E_δ happens with probability $1 - \frac{\delta}{6(k+k^2)}$ that

$$\left|\frac{w \cdot x}{\tau_k}\right| \leq c'\sqrt{d}\log\left(\frac{|W|k}{\delta}\right)$$

for all $x \in W$, for some constant c'. This means $l_{\tau_k}(w, x, \text{sign}(w^* \cdot x))$ are also bounded in the same range under E_δ.

Define the function class $\mathcal{F} = \{x \to l_{\tau_k}(w, x, \text{sign}(w^* \cdot x)), \|w - w_k\| \leq r_k\}$. On event E_δ, all functions in $\mathcal{F}$ are bounded. Now we bound the Rademacher complexity $R_n(\mathcal{F})$. Actually, define $\mathcal{F}' = \{x \to \frac{1}{\tau_k} w \cdot x \cdot \text{sign}(w^* \cdot x), \|w - w_k\| \leq r_k\}$, we have $R_n(\mathcal{F}) \leq R_n(\mathcal{F}')$ by contraction

inequality of Rademacher complexity (since hinge loss is 1-Lipschitz). So

$$
\begin{aligned}
R_n(\mathcal{F}) &\le R_n(\mathcal{F}') \\
&= \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} \sum_{i=1}^{n} \sigma_i w \cdot x_i \cdot \text{sign}(w^* \cdot x_i) \\
&= \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} \sum_{i=1}^{n} \sigma_i (w \cdot x_i) \qquad (2.6) \\
&= \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sum_{i=1}^{n} \sigma_i (w_k \cdot x_i) + \\
&\frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} \sum_{i=1}^{n} \sigma_i (w - w_k) \cdot x_i \\
&= \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} \sum_{i=1}^{n} \sigma_i (w - w_k) \cdot x_i \\
&= \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} (w - w_k) \sum_{i=1}^{n} \sigma_i \cdot x_i \\
&\le \frac{1}{\tau_k n} E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \sup_{w:\|w-w_k\|\le r_k} \|w - w_k\|_2 \left\| \sum_{i=1}^{n} \sigma_i x_i \right\|_2 \\
&\le \frac{2r_k}{\tau_k n} \sqrt{E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \left\| \sum_{i=1}^{n} \sigma_i x_i \right\|_2^2} \qquad (2.7) \\
&\le \frac{2r_k}{\tau_k n} \sqrt{E_{x_1,\ldots,x_n \sim D_{w_k,b_{k-1}}} E_{\sigma_1,\ldots,\sigma_n} \left[\sum_{i=1}^{n} \|x_i\|_2^2 + \sum_{i,j} \sigma_i \sigma_j x_i \cdot x_j \right]} \qquad (2.8) \\
&\le \mathcal{O}\left(\frac{1}{n} \cdot \sqrt{n d \log^2\left(\frac{nk}{\delta}\right)} \right) \qquad (2.9) \\
&= \mathcal{O}\left(\sqrt{\frac{d \log^2\left(\frac{nk}{\delta}\right)}{n}} \right).
\end{aligned}
$$

Equation (2.6) is by the property that $\sigma_i \cdot \text{sign}(w^* \cdot x_i)$ has the same distribution as σ_i, and thus we can substitute $\sigma_i \cdot \text{sign}(w^* \cdot x_i)$ with a single variable; Equation (2.7) is by Jensen's inequality,

and Equation (2.9) is by the boundary condition on $\|x\|_2$. So by Rademacher's inequality we have

$$\begin{aligned}
|\mathbb{E}_W[l(w,x,\text{sign}(w^*\cdot x))] - l(w,W)| &\le R_{|W|}(\mathcal{F}) + \sqrt{\frac{\log(1/\delta)}{|W|}}C\sqrt{d}\log\left(\frac{|W|k}{\delta}\right) \\
&\le \mathcal{O}\left(\sqrt{\frac{d\log^2\left(\frac{|W|k}{\delta}\right)}{|W|}}\right) + \sqrt{\frac{\log(k/\delta)}{|W|}}C\sqrt{d}\log\left(\frac{|W|k}{\delta}\right) \\
&= \mathcal{O}\left(\sqrt{\frac{d\log(k/\delta)}{|W|}}\log\left(\frac{|W|k}{\delta}\right)\right).
\end{aligned}$$

The choice of $|W| = m_i = \Omega\left(d\log^3\left(\frac{dk}{1/\delta}\right)\right)$ makes the above quantity less than $\kappa/16$. □

Now we are ready to prove Theorem 14.

Proof of Theorem 14. With a probability of $1 - \frac{\delta}{k+k^2}$, suppose the conditions in Lemma 16 and 18 holds for $w = v_k$ and $w = w^*$. We have

$$\begin{aligned}
&\Delta_{D_{w_{k-1},b_{k-1}}}(w_k, w^*) \\
&= \Delta_{D_{w_{k-1},b_{k-1}}}(v_k, w^*) \\
&\le \mathbb{E}_{x\in D_{w_{k-1},b_{k-1}}}[l(v_k,x,\text{sign}(w^*\cdot x))] && \text{(Since hinge loss upper bounds 0-1 loss)} \\
&\le l(v_k, c(W)) + \kappa/16 && \text{(Using Lemma 18)} \\
&\le l(v_k, W) + \kappa/8 && \text{(Using Lemma 16)} \\
&\le l(w^*, W) + \kappa/4 && \text{(By the process of selecting } v_k\text{)} \\
&\le l(w^*, c(W)) + \kappa/4 + \kappa/16 && \text{(Using Lemma 16)} \\
&\le L(w^*) + \kappa/4 + \kappa/8 && \text{(Using Lemma 18)} \\
&\le \kappa. && \text{(Using Lemma 3.7 in [15])}
\end{aligned}$$

□

Now we can prove Theorem 8.

Proof of Theorem 8. By relative Chernoff bound and property (c) in Lemma 15, with probability $1 - \frac{\delta}{6(k+k^2)}$ we have $|W| \ge m_k = 2c_3 b_k n_k + \log(12k/\delta)$ in every iteration. Then the correctness of Margin-ADGAC follows the same way as in [15]. Now examine the number of queries: In each step we need to compare m_i instances, as well as fitting the minimum requirement of ADGAC. So the comparison complexity is

$$\mathbb{E}[\mathsf{SC}_{\text{comp}}] = \tilde{\mathcal{O}}\left(\log^2(1/\varepsilon)\left(d\log^4(d/\delta) + \left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\log(1/\delta)\right)\right).$$

The label complexity is again obtained by multiplying the label complexity in each iteration by $\log(1/\varepsilon)$. Note that $\frac{\varepsilon_k n_k}{m_k}$ is constant in each iteration. Therefore,

$$\mathsf{SC}_{\text{label}} = \tilde{\mathcal{O}}\left(\log(1/\varepsilon)\log(1/\delta)\left(\frac{1}{\varepsilon}\right)^{2\kappa-2}\right).$$

□

Proof of Theorem 9. The proof follows exactly the same process as that of Theorem 8 using $\kappa = 1$, and Theorem 5. □

2.9.5 Proof of Lower Bounds

2.9.6 Proof of Theorem 10

Proof. Suppose $g(x_1) = a$ and $g(x_0) = b$ for $x_1, x_2 \in \mathcal{X}, a < b$. Let $h_1(x) = \text{sign}(g(x) - a)$ and $h_2(x) = \text{sign}(g(x) - b)$. Note that using $Z(x_1, x_2) = 0$ incurs $\nu' = 0$ for both $h^* = h_1$ and $h^* = h_2$, and thus comparison cannot distinguish between h_1 and h_2. Suppose $\mathbb{C} = \{h_1, h_2\}$. Thus, any algorithm $\mathcal{A}$ using both comparison and labeling oracles can be transformed into an algorithm $\mathcal{A}'$ that uses labeling oracle only, by making the comparison oracle always return 0. Note that $\mathsf{SC}_{\text{label}}(\mathcal{A}) = \mathsf{SC}_{\text{label}}(\mathcal{A}')$, so we only need to lower bound $\mathsf{SC}_{\text{label}}(\mathcal{A}')$. In the following, we adapt the proof in [86] to give a lower bound. The main difference is that our goal is to reach a small $\Pr[h(X) \neq h^*(X)]$, whereas in [86] the goal is a small $\mathsf{err}(h) - \mathsf{err}(h^*)$.

Let $P(x_1) = 24\varepsilon, P(x_0) = 1 - 24\varepsilon$. Consider two distributions $\mathcal{P}_1, \mathcal{P}_2$ over $\mathcal{X} \times \mathcal{Y}$ with two different Bayes function $\eta_1(), \eta_2()$. Let $\gamma = \varepsilon^{\kappa-1}$ if $\kappa > 1$, or $\gamma = \frac{1}{48}$ if $\kappa = 1$. Let $\eta_1(x_0) = \eta_2(x_0) = 1, \eta_1(x_1) = \frac{1}{2} + \gamma, \eta_2(x_1) = \frac{1}{2} - \gamma$. It is easy to verify both $\mathcal{P}_1$ and $\mathcal{P}_2$ satisfy Tsybakov noise condition.

Choose the groundtruth distribution to be $\mathcal{P}_1$ or $\mathcal{P}_2$ both with probability $1/2$. By the same proof as Theorem 4.3 in [86], an event happens with probability at least δ that $\hat{h}(x_1) \neq h^*(x_1)$, and thus $\Pr[\hat{h}(X) \neq h^*(X)] \geq \varepsilon$, if at most $2\lfloor \frac{1-\gamma^2}{\gamma^2} \log\left(\frac{1}{8\delta(1-2\delta)}\right) \rfloor$ labels are queried. So we prove the theorem for TNC.

The proof for adversarial noise is the same as the above proof using $\kappa = 1$. □

2.9.7 Proof of Theorem 11

Proof of Theorem 11. The first term in Equation (2.2) follows directly from Theorem 10. For the second term, we consider the case where both labeling and comparison oracles are perfect with $\nu = \nu' = 0$. This is a special case for all Conditions 1, 2 and 3. Notice that in this case, a perfect comparison oracle can be constructed from a labeling oracle by $Z(x, x') = \text{sign}(Y(x) - Y(x')) = \text{sign}(h^*(x) - h^*(x'))$; thus, any algorithm $\mathcal{A}$ with access to both labeling and comparison oracles can be transformed into another algorithm $\mathcal{A}'$ that uses labeling oracle (by replacing the comparison oracle with one that queries labeling oracle instead). So we have

$$2\mathsf{SC}_{\text{comp}}(\mathcal{A}) + \mathsf{SC}_{\text{label}}(\mathcal{A}) = \mathsf{SC}_{\text{label}}(\mathcal{A}') = \Omega(d\log(1/\varepsilon)),$$

where $\Omega(d\log(1/\varepsilon))$ is the standard lower bound for realizable active learning (see e.g., [86]). □

2.9.8 Proof of Theorem 12

Define $R^B(\hat{g})$ to be the error of comparison oracle induced by $\hat{g}$, and also $\mathbb{C}_{\hat{g}} = \{h : h(x) = \text{sign}(\hat{g}(x) - t), t \in \mathbb{R}\}$. To prove Theorem 12, we first give a lower bound on the left hand side (Theorem 19) by giving a $\hat{g}$ that every $h \in \mathbb{C}_{\hat{g}}$ will have every at least $\sqrt{\nu'}$. Then we give an upper bound on it (Theorem 20) by finding a good estimator t. We find t by reducing $\Pr[\text{sign}(\hat{g}(X) - t) \neq h^*(X)]$ to the case when for every x, x' such that $\hat{g}(x) = \hat{g}(x')$ we also have $h^*(x) = h^*(x')$. We find such a good function f in this case by fixing the amount of error at each value of $\hat{g}(x)$, and carefully adjusting the noise levels.

Theorem 19. *Suppose* $\min\{\Pr[h^*(X) = 1], \Pr[h^*(X) = -1]\} \geq \sqrt{\nu'}$. *For any* g^* *such that* $g^*(X)$ *has a density function, there exists* $\hat{g}$ *which induces a comparison oracle with error* ν', *such that for every* $h \in \mathbb{C}_{\hat{g}}$, *we have* $\Pr[h(X) \neq h^*(X)] \geq \sqrt{\nu'}$.

Proof. Consider the distribution of $g^*(X)$. Pick a consecutive interval $I = [a, b]$ with $a < 0 < b$ such that $\Pr(g^*(X) \in [0, b]) = \Pr(g^*(X) \in [a, 0]) = \sqrt{\nu'}$. Pick some integer $n \in \mathbb{N}$. Suppose the cdf and pdf of random variable $T = g^*(X)$ is $F(t)$ and $p(t)$ respectively. Define

$$\hat{g}(x) = \begin{cases} a + (b-a)\frac{F(g^*(x)) - F(a)}{\sqrt{\nu'}}, & \text{if } x \in [a, 0], \\ a + (b-a)\frac{F(g^*(x)) - F(0)}{\sqrt{\nu'}}, & \text{if } x \in (0, b], \\ g^*(x), & \text{otherwise.} \end{cases}$$

The error of the comparison oracle induced by $\hat{g}$ can be represented as

$$R^B(\hat{g}) = 2 \int_{g^*(x) \in (0,b]} p(g^*(x)) \int_{g^*(x') \in [a,0)} p(g^*(x')) \cdot \delta(\hat{g}(x') > \hat{g}(x)) \, \mathrm{d}g^*(x) \mathrm{d}g^*(x')$$

Let $t = g^*(x)$ and $t' = g^*(x')$. Then $\hat{g}(x') > \hat{g}(x)$ if and only if

$$\begin{aligned} & F(t') - F(a) > F(t) - F(0), \\ \Leftrightarrow & F(t) - F(t') < \sqrt{\nu'}. \end{aligned}$$

For every $t \in [0, b]$, let $G(t)$ satisfy $F(t) - F(G(t)) = \sqrt{\nu'}$. Then

$$\begin{aligned}
R^B(\hat{g}) &= 2\int_{t=0}^{b} p(t) \int_{t'=a}^{0} p(t') \cdot \delta\left(F(t) - F(t') < \sqrt{\nu'}\right) \, \mathrm{d}t\mathrm{d}t' \\
&= 2\int_{t=0}^{b} p(t) \int_{t'=a}^{G(t)} p(t') \, \mathrm{d}t\mathrm{d}t' \\
&= 2\int_{t=0}^{b} p(t)(F(G(t)) - F(a))\mathrm{d}t \\
&= 2\int_{t=0}^{b} p(t)(F(t) - F(0))\mathrm{d}t \\
&= 2\int_{t=0}^{b} p(t) \int_{t'=0}^{t} p(t')\mathrm{d}t\mathrm{d}t' \\
&= 2\int_{t=0}^{b} \int_{t'=0}^{b} p(t)p(t')\delta(t' < t)\mathrm{d}t\mathrm{d}t' \\
&= \nu'.
\end{aligned}$$

Now examine any function in $\mathbb{C}_{\hat{g}}$. If we pick a threshold $t \notin [a, b]$, the error is at least $\sqrt{\nu'}$ since we incur error on either $\{x : g^*(x) \in [a, 0]\}$ or $\{x : g^*(x) \in [0, b]\}$. If we pick threshold $a + (b - a)t$ for $t \in [0, 1]$, we induce an error for any $g^*(x) \in [a, 0]$ with $\frac{F(g^*(x)) - F(a)}{\sqrt{v}} > t$, and any $g^*(x) \in (0, b]$ with $\frac{F(g^*(x)) - F(0)}{\sqrt{v}} < t$. A routine calculation shows the error is always $\sqrt{\nu'}$. □

Theorem 20. *Suppose that $\hat{g}$ induces a comparison oracle with error ν', and also distributions of $\hat{g}(X)$ and $g^*(X)$ are smooth in the sense that they both have a density function. There exists $h_t(x) \coloneqq \mathit{sign}(\hat{g}(x) - t) \in \mathbb{C}_{\hat{g}}$ such that the error of $h_t(x)$ with respect to $h^*(x)$ is at most $\sqrt{\nu'}$, i.e.,*

$$\Pr[h_t(X) \neq h^*(X)] = \Pr[(\hat{g}(X) - t)g^*(X) < 0] \leq \sqrt{\nu'}.$$

We first prove the inequality:

Lemma 21. *Suppose $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$ satisfies $x_i, y_i \in \mathbb{R}, x_i, y_i \geq 0$. If $\sum_{i=1}^n \sum_{j=i}^n x_i y_j \leq t$, we have*

$$\min_{k=0,1,\ldots,n} \{x_1 + \cdots + x_k + y_{k+1} + \cdots + y_n\} \leq \sqrt{\frac{2nt}{n+1}},$$

the equality holds when $x_1 = x_2 = \cdots = x_n = y_1 = \cdots = y_n = \sqrt{\frac{2t}{n(n+1)}}$.

Proof of Lemma 21. Let $f(k) = x_1 + \cdots + x_k + y_{k+1} + \cdots + y_n$. We first prove that when the maximum of $\min_{k=0,1,\ldots,n} f(k)$ is achieved, we must have $x_i = y_i$ for all i. If not, not losing generality suppose $x_l > y_l$. Now consider $x'_i = x_i$ for all $i \neq l, l+1$, and $x'_l = y_l, x'_{l+1} = x_{l+1} + x_l - y_l$ (omit the latter step if $l = n$). Let $f'(k)$ be the function of k computed based on x' and y. By $x_l > y_l$ we have $f(l) > f(l-1)$. Notice that only $f'(l) = f(l-1) < f(l)$ is reduced

and for all other $k \neq l$ we have $f(k) = f'(k)$, so the minimum remains the same. Now we have

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=i}^{n} x_i' y_j &= \sum_{j=1}^{n}\sum_{i=1}^{j} x_i' y_j = \sum_{j=1}^{n} y_j \sum_{i=1}^{j} x_i' \\ &= \sum_{j=1}^{l-1} y_j \sum_{i=1}^{j} x_i + y_l \sum_{i=1}^{l} x_i' + \sum_{j=l+1}^{n} y_j \sum_{i=1}^{j} x_i \\ &\leq \sum_{j=1}^{l-1} y_j \sum_{i=1}^{j} x_i + y_l \sum_{i=1}^{l} x_i + \sum_{j=l+1}^{n} y_j \sum_{i=1}^{j} x_i \\ &\leq t.\end{aligned}$$

So there exists a configuration that maximizes $\min_k f(k)$ with $x_i = y_i$ for all i. Now suppose $x_i = y_i$ for all i. The constraint becomes

$$\sum_{i=1}^{n}\sum_{j=i}^{n} x_i x_j \leq \varepsilon,$$

which is equivalent to

$$\left(\sum_{i=1}^{n} x_i\right)^2 + \sum_{i=1}^{n} x_i^2 \leq 2\varepsilon.$$

By Cauchy-Schwarz inequality we have

$$\sum_{i=1}^{n} x_i^2 \geq \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}.$$

So

$$x_1 + \cdots + x_k + y_{k+1} + \cdots + y_n = \sum_{i=1}^{n} x_i \leq \sqrt{\frac{2nt}{n+1}}.$$

It is easy to verify the equality condition. □

Proof of Theorem 20. Not losing generality, suppose $\hat{g}(x) \in [0, 1]$; such a assumption is justifiable since any increasing transformation of $\hat{g}$ does not change $R^B(\hat{g})$. So we only need to consider $\mathbb{C}_{\hat{g}} = \{h : h(x) = h_t(x) = \text{sign}(\hat{g}(x) - t), t \in [0, 1]\}$. Let $q(u)$ denote the distribution of $\hat{g}(X)$. Let $\xi(u) = q(u) \Pr(h^*(X) = 1 | \hat{g}(x) = u)$. So we have

$$\int_0^t \xi(u)du = \Pr(h^*(X) = 1, \hat{g}(X) < t).$$

So the error of h_t with respect to h^* can be expressed as

$$\begin{aligned}\Pr((\hat{g}(X) - t)g^*(X) < 0) &= \Pr(\hat{g}(X) > t, g^*(X) < 0) + \Pr(\hat{g}(X) < t, g^*(X) > 0) \\ &= \int_0^t \xi(u)du + \int_t^1 (q(u) - \xi(u))du.\end{aligned}$$

On the other hand, the comparison error can be expressed as

$$\begin{aligned} R^B(\hat{g}) &= 2\Pr(\hat{g}(X) > \hat{g}(X'), h^*(X) = -1, h^*(X) = 1) \\ &= \int_0^1 \xi(u) \int_u^1 (q(v) - \xi(v)) du dv. \end{aligned}$$

Now consider we do this on the grid with step size $1/n$ and let $n \to \infty$; the integral will be the limit value. So, let

$$S_n = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=i}^{n} \xi(i/n)(q(j/n) - \xi(j/n)).$$

So

$$\Pr(\hat{g}(X) > g(X'), h^*(X) = -1, h^*(X) = 1) = \lim_{n\to\infty} S_n.$$

Also, let

$$T_n^t = \frac{1}{n} \left(\sum_{i:i/n<t} \xi(i/n) + \sum_{i:i/n>=t} (q(i/n) - \xi(i/n)) \right),$$

so

$$\Pr((\hat{g}(X) - t)g(X) < 0) = \lim_{n\to\infty} T_n^t.$$

Now let $x_i = \frac{1}{n}\xi(i/n), y_i = \frac{1}{n}(q(i/n) - \xi(i/n))$ in Lemma 21, and we have

$$\min_t T_n^t \leq \sqrt{\frac{2nS_n}{n+1}}.$$

Note that $\lim_{n\to\infty} 2S_n = R^B(\hat{g}) \leq \nu'$ and let $n \to \infty$ on both side, we have

$$\min_t \Pr[(\hat{g}(X) - t)g(X) < 0] \leq \sqrt{\nu'}.$$

□

Chapter 3

Regression with Labels and Ordinal Information

Classical regression is centered around the development and analysis of methods that use labeled observations, $\{(X_1, y_1), \ldots, (X_n, y_n)\}$, where each $(X_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, in various tasks of estimation and inference. Nonparametric and high-dimensional methods are appealing in practice owing to their flexibility, and the relatively weak a-priori structural assumptions that they impose on the unknown regression function. However, the price we pay is that these methods typically require a large amount of labeled data to estimate complex target functions, scaling exponentially with the dimension for fully nonparametric methods and scaling linearly with the dimension for high-dimensional parametric methods – the so-called curse of dimensionality. This has motivated research on structural constraints – for instance, sparsity or manifold constraints – as well as research on active learning and semi-supervised learning where labeled samples are used judiciously.

We consider a complementary approach, motivated by applications in material science, crowdsourcing, and healthcare, where we are able to supplement a small labeled dataset with a potentially larger dataset of *ordinal* information. Such ordinal information is obtained either in the form of a (noisy) ranking of unlabeled points or in the form of (noisy) pairwise comparisons between function values at unlabeled points. Some illustrative applications of the methods we develop in this chapter include:

Example 1: Crowdsourcing. In crowdsourcing we rely on human labeling effort, and in many cases humans are able to provide more accurate ordinal feedback with substantially less effort (see for instance [150, 154, 164]). When crowdsourced workers are asked to give numerical price estimates, they typically have difficulty giving a precise answer, resulting in a high variance between worker responses. On the other hand, when presented two products or listings side by side, workers may be more accurate at comparing them. We conduct experiments on the task of price estimation in Section 3.5.4.

Example 2: Material Synthesis. In material synthesis, the broad goal is to design complex new materials and machine learning approaches are gaining popularity [72, 194]. Typically, given a setting of input parameters (temperature, pressure etc.) we are able to perform a synthesis

experiment and measure the quality of resulting synthesized material. Understanding this landscape of material quality is essentially a task of high-dimensional function estimation. Synthesis experiments can be costly and material scientists when presented with pairs of input parameters are often able to cheaply and reasonably accurately provide comparative assessments of quality.

Example 3: Patient Diagnosis. In clinical settings, precise assessment of each individual patient's health status can be difficult, expensive, and risky but comparing the relative status of two patients may be relatively easy and accurate.

In each of these settings, it is important to develop methods for function estimation that combine standard supervision with (potentially) cheaper and abundant ordinal or comparative supervision.

3.1 Our Contributions

We consider both linear and nonparametric regression with both direct (cardinal) and comparison (ordinal) information. In both cases, we consider the standard statistical learning setup, where the samples X are drawn i.i.d. from a distribution $\mathbb{P}_X$ on $\mathbb{R}^d$. The labels y are related to the features X as,

$$y = f(X) + \varepsilon,$$

where $f = \mathbb{E}[y|X]$ is the underlying regression function of interest, and ε is the mean-zero label noise. Our goal is to construct an estimator $\widehat{f}$ of f that has low risk or mean squared error (MSE),

$$R(\widehat{f}, f) = \mathbb{E}(\widehat{f}(X) - f(X))^2,$$

where the expectation is taken over the labeled and unlabeled training samples, as well as a new test point X. We also study the fundamental information-theoretic limits of estimation with classical and ordinal supervision by establishing lower (and upper) bounds on the minimax risk. Letting η denote various problem dependent parameters, which we introduce more formally in the sequel, the minimax risk:

$$\mathfrak{M}(m, n; \eta) = \inf_{\widehat{f}} \sup_{f \in \mathcal{F}_\eta} R(\widehat{f}, f), \tag{3.1}$$

provides an information-theoretic benchmark to assess the performance of an estimator.

First, focusing on nonparametric regression, we develop a novel Ranking-Regression (R^2) algorithm for nonparametric regression that can leverage ordinal information, in addition to direct labels. We make the following contributions:

- To establish the usefulness of ordinal information in nonparametric regression, in Section 3.3.2 we consider the idealized setting where we obtain a perfect ordering of the unlabeled set. We show that the risk of the R^2 algorithm can be bounded with high-probability as $\widetilde{O}(m^{-2/3} + n^{-2/d})$[1], where m denotes the number of labeled samples and n

[1]We use the standard big-O notation throughout this chapter, and use $\widetilde{O}$ when we suppress log-factors, and $\widehat{O}$ when we suppress log-log factors.

the number of ranked samples. To achieve an MSE of ε, the number of labeled samples required by R^2 is *independent* of the dimensionality of the input features. This result establishes that sufficient ordinal information of high quality can allow us to effectively circumvent the curse of dimensionality.

- In Sections 3.3.3 and 3.3.4 we analyze the R^2 algorithm when using either a noisy ranking of the samples or noisy pairwise comparisons between them. For noisy ranking, we show that the MSE is bounded by $\widetilde{O}(m^{-2/3} + \sqrt{\nu} + n^{-2/d})$, where ν is the Kendall-Tau distance between the true and noisy ranking. As a corollary, we combine R^2 with algorithms for ranking from pairwise comparisons [39] to obtain an MSE of $\widetilde{O}(m^{-2/3} + n^{-2/d})$ when $d \geq 4$, when the comparison noise is bounded.

Turning our attention to the setting of linear regression with ordinal information, we develop an algorithm that uses active (or adaptive) comparison queries in order to reduce both the label complexity and total query complexity.

- In Section 3.4 we develop and analyze an interactive learning algorithm that estimates a linear predictor using both labels and comparison queries. Given a budget of m label queries and n comparison queries, we show that MSE of our algorithm decays at the rate of $\widetilde{O}\left(1/m + \exp(-n/d)\right)$. Once again we see that when sufficiently many comparisons are available, the label complexity of our algorithm is *independent* of the dimension d.

To complement these results we also give information-theoretic lower bounds to characterize the fundamental limits of combining ordinal and standard supervision.

- For nonparametric regression, we show that the R^2 algorithm is optimal up to log factors both when it has access to a perfect ranking, as well as when the comparisons have bounded noise. For linear regression, we show that the rate of $O(1/m)$, and the total number of queries, are not improvable up to log factors.

On the empirical side we comprehensively evaluate the algorithms we propose, on simulated and real datasets.

- We use simulated data, and study the performance of our algorithms as we vary the noise in the labels and in the ranking.
- Second, we consider a practical application of predicting people's ages from photographs. For this dataset we obtain comparisons using people's apparent ages (as opposed to their true biological age).
- Finally, we curate a new dataset using crowdsourced data obtained through Amazon's Mechanical Turk. We provide workers AirBnB listings and attempt to estimate property asking prices. We obtain both direct and comparison queries for the listings, and also study the time taken by workers to provide these different types of feedback. We find that, our algorithms which combine direct and comparison queries are able to achieve significantly better accuracy than standard supervised regression methods, for a fixed time budget.

3.2 Related Works

As a classical way to reduce the labeling effort, active learning has been mostly focused on classification [85]. For regression, it is known that, in many natural settings, the ability to make

active queries does not lead to improved rates over passive baselines. For example, [47] shows that when the underlying model is linear, the ability to make active queries can only improve the rate of convergence by a constant factor, and leads to no improvement when the feature distribution is spherical Gaussian. In [178], the authors show a similar result that in nonparametric settings, active queries do not lead to a faster rate for smooth regression functions except when the regression function is piecewise smooth.

There is considerable work in supervised and unsupervised learning on incorporating additional types of feedback beyond labels. For instance, [138, 207] study the benefits of different types of "feature feedback" in clustering and supervised learning respectively. [62] consider learning with partial corrections, where the user provides corrective feedback to the algorithm when the algorithm makes an incorrect prediction.

There is also a vast literature on models and methods for analyzing pairwise comparison data, like the classical Bradley-Terry [38] and Thurstone [162] models. In this literature, the typical focus is on ranking or quality estimation for a fixed set of objects. In contrast, we focus on function estimation and the resulting models and methods are quite different. We build on the work on "noisy sorting" [39, 151] to extract a consensus ranking from noisy pairwise comparisons.

Given ordinal information of sufficient fidelity, the problem of nonparametric regression is related to the problem of regression with shape constraints, or more specifically isotonic regression [29, 203]. Accordingly, we leverage such algorithms in our work and we comment further on the connections in Section 3.3.2. Some salient differences between this literature and our work are that we design methods that work in a semisupervised setting, and further that our target is an unknown d-dimensional (smooth) regression function as opposed to a univariate shape-constrained function.

The rest of this chapter is organized as follows. In Section 3.3, we consider the problem of combining direct and comparison-based labels for nonparametric regression, providing upper and lower bounds for both noiseless and noisy ordinal models. In Section 3.4, we consider the problem of combining adaptively chosen direct and comparison-based labels for linear regression. In Section 3.5, we turn our attention to an empirical evaluation of our proposed methods on real and synthetic data. Finally, we conclude in Section 3.6 with a number of additional results and open directions. In the Appendix we present detailed proofs for various technical results and a few additional supplementary experimental results.

3.3 Nonparametric Regression with Ordinal Information

We now provide analysis for nonparametric regression. First, in Section 3.3.1 we establish the problem setup and notations. Then, we introduce the R^2 algorithm in Section 3.3.2 and analyze it under perfect rankings. Next, we analyze its performance for noisy rankings and comparisons in Sections 3.3.3 and 3.3.4.

3.3.1 Background and Problem Setup

We consider a nonparametric regression model with random design, i.e. we suppose first that we are given access to an unlabeled set $\mathcal{U} = \{X_1, \ldots, X_n\}$, where $X_i \in \mathcal{X} \subset [0,1]^d$, and X_i

are drawn i.i.d. from a distribution $\mathbb{P}_X$ and we assume that $\mathbb{P}_X$ has a density p which is upper and lower bounded as $0 < p_{\min} \leq p(x) \leq p_{\max}$ for $x \in \mathcal{X}$. Our goal is to estimate a function $f : \mathcal{X} \mapsto \mathbb{R}$, where following classical work [82, 166] we assume that f is bounded in $[-M, M]$ and belongs to a Hölder ball $\mathcal{F}_{s,L}$, with $0 < s \leq 1$:

$$\mathcal{F}_{s,L} = \{f : |f(x) - f(y)| \leq L\|x - y\|_2^s, \forall\, x, y \in \mathcal{X}\}\,.$$

For $s = 1$ this is the class of Lipschitz functions. We discuss the estimation of smoother functions (i.e. the case when $s > 1$) in Section 3.6. We obtain two forms of supervision:

1. **Classical supervision:** For a (uniformly) randomly chosen subset $\mathcal{L} \subseteq \mathcal{U}$ of size m (we assume throughout that $m \leq n$ and focus on settings where $m \ll n$) we make noisy observations of the form:

   $$y_i = f(X_i) + \epsilon_i, \;\; i \in \mathcal{L},$$

   where ϵ_i are i.i.d. Gaussian with $\mathbb{E}[\epsilon_i] = 0$, $\text{Var}[\epsilon_i] = 1$. We denote the indices of the labeled samples as $\{t_1, \ldots, t_m\} \subset \{1, \ldots, n\}$.

2. **Ordinal supervision:** For the given dataset $\{X_1, \ldots, X_n\}$ we let π denote the *true ordering*, i.e. π is a permutation of $\{1, \ldots, n\}$ such that for $i, j \in \{1, \ldots, n\}$, with $\pi(i) \leq \pi(j)$ we have that $f(X_i) \leq f(X_j)$. We assume access to one of the following types of ordinal supervision:

   (1) We assume that we are given access to a noisy ranking $\widehat{\pi}$, i.e. for a parameter $\nu \in [0, 1]$ we assume that the Kendall-Tau distance between $\widehat{\pi}$ and the true-ordering is upper-bounded as:

   $$\sum_{i,j \in [n]} \mathbb{I}[(\pi(i) - \pi(j))(\widehat{\pi}(i) - \widehat{\pi}(j)) < 0] \leq \nu n^2. \tag{3.2}$$

   (2) For each pair of samples (X_i, X_j), with $i < j$ we obtain a comparison Z_{ij} where for some constant $\lambda > 0$:

   $$\mathbb{P}(Z_{ij} = \mathbb{I}(f(X_i) > f(X_j))) \geq \frac{1}{2} + \lambda. \tag{3.3}$$

   As we discuss in Section 3.3.4, it is straightforward to extend our results to a setting where only a randomly chosen subset of all pairwise comparisons are observed.

Although classic supervised learning learns a regression function with labels only and without ordinal supervision, we note that learning cannot happen in the opposite way: That is, we cannot consistently estimate the underlying function with only ordinal supervision and without labels: the underlying function is only identifiable up to certain monotonic transformations.

As discussed in Section 3.1, our goal is to design an estimator $\widehat{f}$ of f that achieves the minimax mean squared error (3.1), when $f \in \mathcal{F}_{s,L}$. We conclude this section recalling a well-known fact: given access to only classical supervision the minimax risk $\mathfrak{M}(m; \eta) = \Theta(m^{-\frac{2s}{2s+d}})$, suffers from an exponential curse of dimensionality.

3.3.2 Nonparametric Regression with Perfect Ranking

To ascertain the value of ordinal information we first consider an idealized setting, where we are given a perfect ranking π of the unlabeled samples in $\mathcal{U}$. We present our Ranking-Regression (R^2) algorithm with performance guarantees in Section 3.3.2, and a lower bound for it in Section 3.3.2 which shows that R^2 is optimal up to log factors.

Upper bounds for the R^2 Algorithm

Algorithm 5 R^2: Ranking-Regression

Input: Unlabeled data $\mathcal{U} = \{X_1, \ldots, X_n\}$, a labeled set of size m and corresponding labels, i.e. samples $\{(X_{t_1}, y_{t_1}), \ldots, (X_{t_m}, y_{t_m})\}$, and a ranking $\widehat{\pi}$.

1: Order elements in $\mathcal{U}$ as $(X_{\widehat{\pi}(1)}, \ldots, X_{\widehat{\pi}(n)})$.

2: Run isotonic regression (see Equation (3.4)) on $\{y_{t_1}, \ldots, y_{t_m}\}$. Denote the estimated values by $\{\widehat{y}_{t_1}, \ldots, \widehat{y}_{t_m}\}$.

3: For $i = 1, 2, \ldots, n$, let $\widetilde{i} = t_k$, where $\widehat{\pi}(t_k)$ is the largest value such that $\widehat{\pi}(t_k) \leq \widehat{\pi}(i), k = 0, 1, \ldots, m$, and $\widetilde{i} = \star$ if no such t_k exists. Set

$$\widehat{y}_i = \begin{cases} \widehat{y}_{\widetilde{i}} & \text{if } \widetilde{i} \neq \star \\ 0 & \text{otherwise.} \end{cases}$$

Output: Function $\widehat{f} =$ NearestNeighbor$(\{(X_i, \widehat{y}_i)\}_{i=1}^n)$.

Our nonparametric regression estimator is described in Algorithm 5 and Figure 3.1. We first rank all the samples in $\mathcal{U}$ according to the (given or estimated) permutation $\widehat{\pi}$. We then run isotonic regression [29] on the labeled samples in $\mathcal{L}$ to de-noise them and borrow statistical strength. In more detail, we solve the following program to de-noise the labeled samples:

$$\begin{aligned} \min_{\{\widehat{y}_{\widehat{\pi}(t_1)}, \ldots, \widehat{y}_{\widehat{\pi}(t_m)}\}} & \sum_{k=1}^{m} (\widehat{y}_{\widehat{\pi}(t_k)} - y_{\widehat{\pi}(t_k)})^2 \\ \text{s.t.} \quad & \widehat{y}_{t_k} \leq \widehat{y}_{t_l} \;\; \forall \, (k, l) \;\; \text{such that} \;\; \widehat{\pi}(t_k) < \widehat{\pi}(t_l) \\ & -M \leq \{\hat{y}_{\widehat{\pi}(t_1)}, \ldots, \hat{y}_{\widehat{\pi}(t_m)}\} \leq M. \end{aligned} \tag{3.4}$$

We introduce the bounds $\{M, -M\}$ in the above program to ease our analysis. In our experiments, we simply set M to be a large positive value so that it has no influence on our estimator. We then leverage the ordinal information in $\widehat{\pi}$ to impute regression estimates for the unlabeled samples in $\mathcal{U}$, by assigning each unlabeled sample the value of the nearest (de-noised) labeled sample which has a smaller function value according to $\widehat{\pi}$. Finally, for a new test point, we use the imputed (or estimated) function value of the nearest neighbor in $\mathcal{U}$.

In the setting where we use a perfect ranking the following theorem characterizes the performance of R^2:

Theorem 22. *For constants $C_1, C_2 > 0$ the MSE of $\widehat{f}$ is bounded by*

$$\mathbb{E}(\widehat{f}(X) - f(X))^2 \leq C_1 m^{-2/3} \log^2 n \log m + C_2 n^{-2s/d}.$$

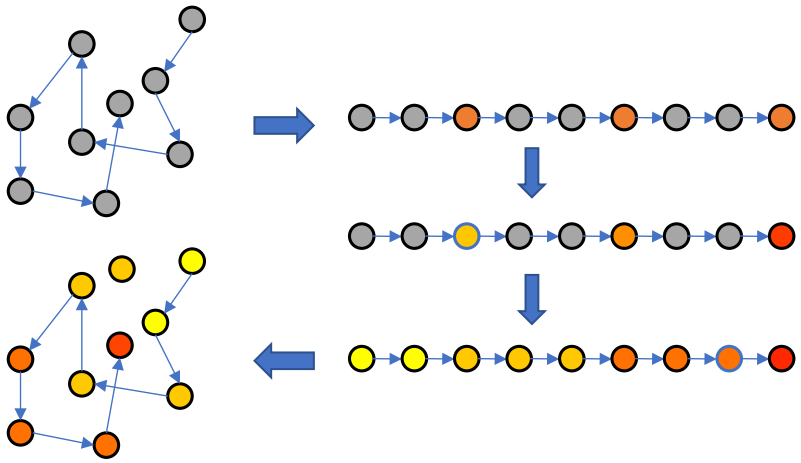


Figure 3.1: Top Left: A group of unlabeled points are ranked according to function values using ordinal information only. Top Right: We obtain function values of m randomly chosen samples. Middle Right: The values are adjusted using isotonic regression. Bottom Right: Function values of other unlabeled points are inferred. Bottom Left: For a new point, the estimated value is given by the nearest neighbor in $\mathcal{U}$.

Before we turn our attention to the proof of this result, we examine some consequences.

Remarks: (1) Theorem 22 shows a surprising dependency on the sizes of the labeled and unlabeled sets (m and n). The MSE of nonparametric regression using only the labeled samples is $\Theta(m^{-\frac{2s}{2s+d}})$ which is exponential in d and makes nonparametric regression impractical in high-dimensions. Focusing on the dependence on m, Theorem 22 improves the rate to $m^{-2/3}\text{polylog}(m,n)$, which is no longer exponential. By using enough ordinal information we can avoid the curse of dimensionality.

(2) On the other hand, the dependence on n (which dictates the amount of ordinal information needed) is still exponential. This illustrates that ordinal information is most beneficial when it is copious. We show in Section 3.3.2 that this is unimprovable in an information-theoretic sense.

(3) Somewhat surprisingly, we also observe that the dependence on n is faster than the $n^{-\frac{2s}{2s+d}}$ rate that would be obtained if all the samples were labeled. Intuitively, this is because of the noise in labels: Given the m labels along with the (perfect) ranking, the difference between two neighboring labels is typically very small (around $1/m$). Therefore, any unlabeled points in $\mathcal{U} \setminus \mathcal{L}$ will be restricted to an interval much narrower than the constant noise in direct labels. In the case where all points are labeled (i.e., $m = n$), the MSE is of order $n^{-2/3} + n^{-2s/d}$, also slightly better rate than when no ordinal information is available. On the other hand, the improvement is stronger when $m \ll n$.

(4) Finally, we also note in passing that the above theorem provides an upper bound on the minimax risk in Equation (3.1).

Proof Sketch We provide a brief outline and defer technical details to the Supplementary Material. For a randomly drawn point $X \in \mathcal{X}$, we denote by X_α the nearest neighbor of X in $\mathcal{U}$. We

decompose the MSE as

$$\mathbb{E}\left[(\widehat{f}(X) - f(X))^2\right] \leq 2\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] + 2\mathbb{E}\left[(f(X_\alpha) - f(X))^2\right]. \tag{3.5}$$

The second term corresponds roughly to the finite-sample bias induced by the discrepancy between the function value at X and the closest labeled sample. We use standard sample-spacing arguments (see [82]) to bound this term. This term contributes the $n^{-2s/d}$ rate to the final result. For the first term, we show a technical result in the Appendix (Lemma 34). Without loss of generality suppose $f\big(X_{t_1}\big) \leq \cdots f\big(X_{t_m}\big)$. By conditioning on a probable configuration of the points and enumerating over choices of the nearest neighbor we find that roughly (see Lemma 34 for a precise statement):

$$\begin{aligned}&\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] \leq \Big(\frac{\log^2 n \log m}{m}\Big)\times \\ &\mathbb{E}\Big(\sum_{k=1}^{m}\Big(\big(\widehat{f}(X_{t_k}) - f(X_{t_k})\big)^2 + \big(f\big(X_{t_{k+1}}\big) - f\big(X_{t_k}\big)\big)^2\Big)\Big).\end{aligned} \tag{3.6}$$

Intuitively, these terms are related to the estimation error arising in isotonic regression (first term) and a term that captures the variance of the function values (second term). When the function f is bounded, we show that the dominant term is the isotonic estimation error which is on the order of $m^{1/3}$. Putting these pieces together we obtain the theorem. ■

Lower bounds with Ordinal Data

To understand the fundamental limits on the usefulness of ordinal information, as well as to study the optimality of the R^2 algorithm we now turn our attention to establishing lower bounds on the minimax risk. In our lower bounds we choose $\mathbb{P}_{\mathcal{X}}$ to be uniform on $[0,1]^d$. Our estimators $\widehat{f}$ are functions of the labeled samples: $\{(X_{t_1}, y_{t_1}), \ldots, (X_{t_m}, y_{t_m})\}$, the set $\mathcal{U} = \{X_1, \ldots, X_n\}$ and the true ranking π. We have the following result:

Theorem 23. *For any estimator $\widehat{f}$ we have that for a universal constant $C > 0$,*

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}_{s,L}} \mathbb{E}\left[(f(X) - \widehat{f}(X))^2\right] \geq C(m^{-2/3} + n^{-2s/d}).$$

Comparing with the result in Theorem 22 we conclude that the R^2 algorithm is optimal up to log factors, when the ranking is noiseless.

Proof Sketch We establish each term in the lower bound separately. Intuitively, for the $n^{-2s/d}$ lower bound we consider the case when all the n points are labeled perfectly (in which case the ranking is redundant) and show that even in this setting the MSE of any estimator is at least $n^{-2s/d}$ due to the finite resolution of the sample.

To prove the $m^{-2/3}$ lower bound we construct a novel packing set of functions in the class $\mathcal{F}_{s,L}$, and use information-theoretic techniques (Fano's inequality) to establish the lower bound. The functions we construct are all increasing functions, and as a result the ranking π provides no

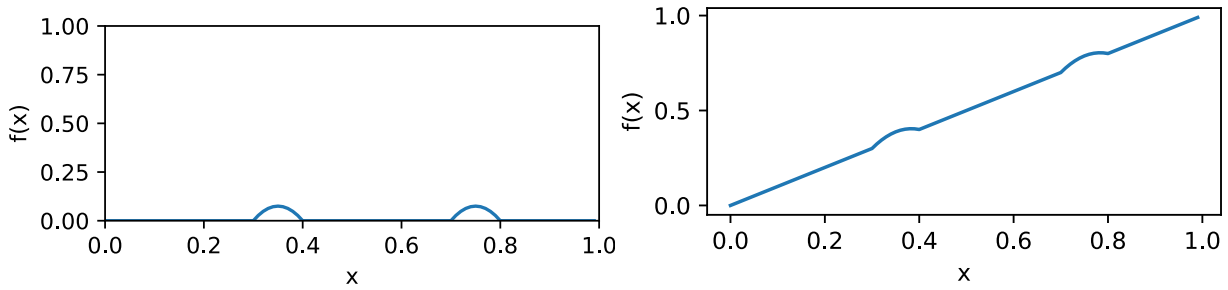


Figure 3.2: Original construction for nonparametric regression in 1-d (left), and our construction (right).

additional information for these functions, easing the analysis. Figure 3.2 contrasts the classical construction for lower bounds in nonparametric regression (where tiny bumps are introduced to a reference function) with our construction where we additionally ensure the perturbed functions are all increasing. To complete the proof, we provide bounds on the cardinality of the packing set we create, as well as bounds on the Kullback-Leibler divergence between the induced distributions on the labeled samples. We provide the technical details in Section 3.8.2.

■

3.3.3 Nonparametric Regression using Noisy Ranking

In this section, we study the setting where the ordinal information is noisy. We focus here on the setting where as in equation Equation (3.2) we obtain a ranking $\widehat{\pi}$ whose Kendall-Tau distance from the true ranking π is at most νn^2. We show that the R^2 algorithm is quite robust to ranking errors and achieves an MSE of $\widetilde{O}(m^{-2/3} + \sqrt{\nu} + n^{-2s/d})$. We establish a complementary lower bound of $\widetilde{O}(m^{-2/3} + \nu^2 + n^{-2s/d})$ in Section 3.3.3.

Upper Bounds for the R^2 Algorithm

We characterize the robustness of R^2 to ranking errors, i.e. when $\widehat{\pi}$ satisfies the condition in Equation (3.2), in the following theorem:

Theorem 24. *For constants $C_1, C_2 > 0$, the MSE of the R^2 estimate $\widehat{f}$ is bounded by*

$$\mathbb{E}[(\widehat{f}(X) - f(X))^2] \leq C_1 \left(\log^2 n \log m \left(m^{-2/3} + \sqrt{\nu}\right)\right) + C_2 n^{-2s/d}.$$

Remarks: (1) Once again we observe that in the regime where sufficient ordinal information is available, i.e. when n is large, the rate no longer has an exponential dependence on the dimension d.
(2) This result also shows that the R^2 algorithm is inherently robust to noise in the ranking, and the mean squared error degrades gracefully as a function of the noise parameter ν. We investigate the optimality of the $\sqrt{\nu}$-dependence in the next section.

We now turn our attention to the proof of this result.

Proof Sketch When using an estimated permutation $\widehat{\pi}$ the true function of interest f is no longer an increasing (isotonic) function with respect to $\widehat{\pi}$, and this results in a model-misspecification *bias*. The core technical novelty of our proof is in relating the upper bound on the error in $\widehat{\pi}$ to an upper bound on this bias. Concretely, in the Appendix we show the following lemma:

Lemma 25. *For any permutation $\widehat{\pi}$ satisfying the condition in Equation (3.2)*

$$\sum_{i=1}^{n}(f(X_{\pi^{-1}(i)}) - f(X_{\widehat{\pi}^{-1}(i)}))^2 \leq 8M^2\sqrt{2\nu}n.$$

Using this result we bound the minimal error of approximating an increasing sequence according to π by an increasing sequence according to the estimated (and misspecified) ranking $\widehat{\pi}$. We denote this error on m labeled points by Δ, and using Lemma 25 we show that in expectation (over the random choice of the labeled set)

$$\mathbb{E}[\Delta] \leq 8M^2\sqrt{2\nu}m.$$

With this technical result in place we follow the same decomposition and subsequent steps before we arrive at the expression in equation Equation (3.6). In this case, the first term for some constant $C > 0$ is bounded as:

$$\mathbb{E}\Big(\sum_{k=1}^{m}\big(\widehat{f}(X_{t_k}) - f(X_{t_k})\big)^2\Big) \leq 2\mathbb{E}[\Delta] + Cm^{1/3},$$

where the first term corresponds to the model-misspecification bias and the second corresponds to the usual isotonic regression rate. Putting these terms together in the decomposition in Equation (3.6) we obtain the theorem.

■

In settings where ν is large R^2 can be led astray by the ordinal information, and a standard nonparametric regressor can achieve the (possibly faster) $O\big(m^{-\frac{2s}{2s+d}}\big)$ rate by ignoring the ordinal information. In this case, a simple and standard cross-validation procedure can combine the benefits of both methods: we estimate the regression function twice, once using R^2 and once using k nearest neighbors, and choose the regression function that performs better on a held-out validation set. The following theorem shows guarantees for this method and an upper bound for the minimax risk (3.1):

Theorem 26. *Under the same assumptions as Theorem 24, there exists an estimator $\widehat{f}$ such that*

$$\mathbb{E}[(\widehat{f}(X) - f(X))^2] = \widetilde{O}\left(m^{-2/3} + \min\{\sqrt{\nu}, m^{-\frac{2s}{2s+d}}\} + n^{-2s/d}\right).$$

The main technical difficulty in analyzing the model-selection procedure we propose in this context is that a naïve analysis of the procedure, using classical tail bounds to control the deviation between the empirical risk and the population risk, results in an excess risk of $\widetilde{O}(1/\sqrt{m})$. However, this rate would overwhelm the $\widetilde{O}(m^{-2/3})$ bound that arises from isotonic regression. We instead follow a more careful analysis outlined in the work of [30] which exploits properties of the squared-loss to obtain an excess risk bound of $\widetilde{O}(1/m)$. We provide a detailed proof in the Appendix for convenience and note that in our setting, the y values are not assumed to be bounded and this necessitates some minor modifications to the original proof [30].

Lower bounds with Noisy Ordinal Data

In this section we turn our attention to lower bounds in the setting with noisy ordinal information. In particular, we construct a permutation $\widehat{\pi}$ such that for a pair (X_i, X_j) of points randomly chosen from $\mathbb{P}_{\mathcal{X}}$:

$$\mathbb{P}[(\pi(i) - \pi(j))(\widehat{\pi}(i) - \widehat{\pi}(j)) < 0] \leq \nu.$$

We analyze the minimax risk of an estimator which has access to this noisy permutation $\widehat{\pi}$, in addition to the labeled and unlabeled sets (as in Section 3.3.2).

Theorem 27. *There is a constant $C > 0$ such that for any estimator $\widehat{f}$ taking input $X_1, \ldots, X_n, y_1, \ldots, y_m$ and $\widehat{\pi}$,*

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}_{s,L}} \mathbb{E}\big(f(X) - \widehat{f}(X)\big)^2 \geq C(m^{-\frac{2}{3}} + \min\{\nu^2, m^{-\frac{2}{d+2}}\} + n^{-2s/d}).$$

Comparing this result with our result in Remark 3 following Theorem 24, our upper and lower bounds differ by the gap between $\sqrt{\nu}$ and ν^2, in the case of Lipschitz functions ($s = 1$).

Proof Sketch We focus on the dependence on ν, as the other parts are identical to Theorem 23. We construct a packing set of Lipschitz functions, and we subsequently construct a noisy comparison oracle $\widehat{\pi}$ which provides no additional information beyond the labeled samples. The construction of our packing set is inspired by the construction of standard lower bounds in nonparametric regression (see Figure 3.2), but we modify this construction to ensure that $\widehat{\pi}$ is uninformative. In the classical construction we divide $[0, 1]^d$ into u^d grid points, with $u = m^{1/(d+2)}$ and add a "bump" at a carefully chosen subset of the grid points. Here we instead divide $[0, t]^d$ into a grid with u^d points, and add an increasing function along the first dimension, where t is a parameter we choose in the sequel.

We now describe the ranking oracle which generates the permutation $\widehat{\pi}$: we simply rank sample points according to their first coordinate. This comparison oracle only makes an error when both x, x' lies in $[0, t]^d$, and both x_1, x_1' lie in the same grid segment $[tk/u, t(k + 1)/u]$ for some $k \in [u]$. So the Kendall-Tau error of the comparison oracle is $(t^d)^2 \times ((1/u)^2 \times u) = ut^{2d}$. We choose t such that this value is less than ν. Once again we complete the proof by lower bounding the cardinality of the packing-set for our stated choice of t, upper bounding the Kullback-Leibler divergence between the induced distributions and appealing to Fano's inequality. ■

3.3.4 Regression with Noisy Pairwise Comparisons

In this section we focus on the setting where the ordinal information is obtained in the form of noisy pairwise comparisons, following equation Equation (3.3). We investigate a natural strategy of aggregating the pairwise comparisons to form a consensus ranking $\widehat{\pi}$ and then applying the R^2 algorithm with this estimated ranking. We build on results from theoretical computer science, where such aggregation algorithms are studied for their connections to sorting with noisy comparators. In particular, Braverman and Mossel [39] study noisy sorting algorithms under the noise model described in Equation (3.3) and establish the following result:

Theorem 28 ([39]). *Let $\alpha > 0$. There exists a polynomial-time algorithm using noisy pairwise comparisons between n samples, that with probability $1 - n^{-\alpha}$, returns a ranking $\widehat{\pi}$ such that for a constant $c(\alpha, \lambda) > 0$ we have that:*

$$\sum_{i,j \in [n]} \mathbb{I}[(\pi(i) - \pi(j))(\widehat{\pi}(i) - \widehat{\pi}(j)) < 0] \leq c(\alpha, \lambda) n.$$

Furthermore, if allowed a sequential (active) choice of comparisons, the algorithm queries at most $O(n \log n)$ pairs of samples.

Combining this result with our result on the robustness of R^2 we obtain an algorithm for nonparametric regression with access to noisy pairwise comparisons with the following guarantee on its performance:

Corollary 29. *For constants $C_1, C_2 > 0$, R^2 with $\widehat{\pi}$ estimated as described above produces an estimator $\widehat{f}$ with MSE*

$$\mathbb{E}\big(\widehat{f}(X) - f(X)\big)^2 \leq C_1 m^{-2/3} \log^2 n \log m + C_2 \max\{n^{-2s/d}, n^{-1/2} \log^2 n \log m\}.$$

Remarks:

1. From a technical standpoint this result is an immediate corollary of Theorems 24 and 28, but the extension is important from a practical standpoint. The ranking error of $O(1/n)$ from the noisy sorting algorithm leads to an additional $\widetilde{O}(1/\sqrt{n})$ term in the MSE. This error is dominated by the $n^{-2s/d}$ term if $d \geq 4s$, and in this setting the result in Corollary 29 is also optimal up to log factors (following the lower bound in Section 3.3.2).

2. We also note that the analysis in [39] extends in a straightforward way to a setting where only a randomly chosen subset of the pairwise comparisons are obtained.

3.4 Linear Regression with Comparisons

In this section we investigate another popular setting for regression, that of fitting a linear predictor to data. We show that when we have enough comparisons, it is possible to estimate a linear function even when $m \ll d$, without making *any* sparsity or structural assumptions.

For linear regression we follow a different approach when compared to the nonparametric setting we have studied thus far. By exploiting the assumption of linearity, we see that each comparison now translates to a constraint on the parameters of the linear model, and as a consequence we are able to use comparisons to obtain a good initial estimate. However, the unknown linear regression parameters are not fully identified by comparison. For instance, we observe that the two regression vectors w and $2 \times w$ induce the same comparison results for any pairs (X_1, X_2). This motivates using direct measurements to estimate a global scaling of the unknown regression parameters, i.e the norm of regression vector w^*. Essentially by using comparisons instead of direct measurements to estimate the weights, we convert the regression problem to a *classification* problem, and therefore can leverage existing algorithms and analyses from the passive/active classification literature.

We present our assumptions and notation for the linear setup in the next subsection. Then we give the algorithm along with its analysis in Section 3.4.2. We also present information-theoretic lower bounds on the minimax rate in Section 3.4.3.

3.4.1 Background and Problem Setup

Following some of the previous literature on estimating a linear classifier [13, 14], we assume that the distribution $\mathbb{P}_X$ is isotropic and log-concave. In more detail, we assume that coordinates of X are independent, centered around 0, have covariance I_d; and that the log of the density function of X is concave. This assumption is satisfied by many standard distributions, for instance the uniform and Gaussian distributions [121]. We let $B(v, r)$ denote the ball of radius r around vector v. Our goal is to estimate a linear function $f(X) = \langle w^*, X\rangle$, and for convenience we denote:

$$r^* = \|w^*\|_2 \quad \text{and} \quad v^* = \frac{w^*}{\|w^*\|_2}.$$

Similar to the nonparametric case, we suppose that we have access to two kinds of supervision using labels and comparisons respectively. We represent these using the following two oracles:

- **Label Oracle:** We assume access to a label oracle $\mathcal{O}_l$, which takes a sample $X \in \mathbb{R}^d$ and outputs a label $y = \langle w^*, X\rangle + \varepsilon$, with $\mathbb{E}[\varepsilon] = 0$, $\text{Var}(\varepsilon) = \sigma^2$.
- **Comparison Oracle:** We also have access to a (potentially cheaper) comparison oracle $\mathcal{O}_c$. For each query, the oracle $\mathcal{O}_c$ receives a pair of samples $(X, X') \sim \mathbb{P}_X \times \mathbb{P}_X$, and returns a random variable $Z \in \{-1, +1\}$, where $Z = 1$ indicates that the oracle believes that $f(X) > f(X')$, and $Z = -1$ otherwise. We assume that the oracle has agnostic noise[2] ν, i.e.:

  $$\mathbb{P}(Z \neq \text{sign}(\langle w^*, X - X'\rangle)) \leq \nu.$$

  That is, for a randomly sampled triplet (X, X', Z) the oracle is wrong with probability at most ν. Note that the error of the oracle for a particular example $(X, X') = (x, x')$ can be arbitrary.

Given a set of unlabeled instances $\mathcal{U} = \{X_1, X_2, \ldots\}$ drawn from $\mathbb{P}_X$, we aim to estimate w^* by querying $\mathcal{O}_l$ and $\mathcal{O}_c$ with samples in $\mathcal{U}$, using a label and comparison budget of m and n respectively. Our algorithm can be either passive, active or semi-supervised; we denote the output of algorithm $\mathcal{A}$ by $\widehat{w} = \mathcal{A}(\mathcal{U}, \mathcal{O}_l, \mathcal{O}_c)$. For an algorithm $\mathcal{A}$, we study the minimax risk (3.1), which in this case can be written as

$$\mathfrak{M}(m, n) = \inf_{\mathcal{A}} \sup_{w^*} \mathbb{E}\left[\langle w^* - \widehat{w}, X\rangle^2\right]. \tag{3.7}$$

Our algorithm relies on a linear classification algorithm $\mathcal{A}^c$, which we assume to be a proper classification algorithm (i.e. the output of $\mathcal{A}^c$ is a linear classifier which we denote by $\widehat{v}$). We let $\mathcal{A}^c(\mathcal{U}, \mathcal{O}, m)$ denote the output (linear) classifier of $\mathcal{A}^c$ when it uses the unlabeled data pool $\mathcal{U}$,

[2] As we discuss in the sequel our model can also be adapted to the bounded noise model case of eqn. (3.3) using a different algorithm from active learning; see Section 6.4.2 for details.

the label oracle $\mathcal{O}$ and acquires m labels. $\mathcal{A}^c$ can be either passive or active; in the former case the m labels are randomly selected from $\mathcal{U}$, whereas in the latter case $\mathcal{A}^c$ decides which labels to query. We use $\varepsilon_{\mathcal{A}^c}(m, \delta)$ to denote (an upper bound on) the $0/1$ error of the algorithm $\mathcal{A}^c$ when using m labels, with $1 - \delta$ probability, i.e.:

$$\mathbb{P}[\mathsf{err}(\mathcal{A}^c(\mathcal{U}, m)) \leq \varepsilon_{\mathcal{A}^c}(m, \delta)] \geq 1 - \delta.$$

We note that by leveraging the log-concave assumption on P_X, it is straightforward to translate guarantees on the 0/1 error to corresponding guarantees on the ℓ_2 error $\|\widehat{v} - v^*\|_2$.

We conclude this section by noting that the classical minimax rate for ordinary least squares (OLS) is of order $O(d/m)$, where m is the number of label queries. This rate cannot be improved by active label queries (see for instance [47]).

3.4.2 Algorithm and Analysis

Our algorithm, Comparison Linear Regression (CLR), is described in Algorithm 6. We first use the comparisons to construct a classification problem with samples $(X - X')$ and oracle $\mathcal{O}_c$. Here we slightly overload the notation of $\mathcal{O}_c$ that $\mathcal{O}_c(X_i - X_j) = \mathcal{O}_c(X_i, X_j)$. Given these samples we use an active linear classification algorithm to estimate a normalized weight vector $\widehat{v}$. After classification, we use the estimated $\widehat{v}$ along with actual label queries to estimate the norm of the weight vector $\widehat{r}$. Combining these results we obtain our final estimate $\widehat{w} = \widehat{r} \cdot \widehat{v}$.

Algorithm 6 Comparison Linear Regression (CLR)

Input: comparison budget n, label budget m, unlabeled data pool $\mathcal{U}$, algorithm $\mathcal{A}^c$

1: Construct pairwise pool $\mathcal{U}' = \{X_1 - X_2, X_3 - X_4, \ldots, X_{n-1} - X_n\}$
2: Run $\mathcal{A}^c(\mathcal{U}', \mathcal{O}_c, n)$ and obtain a classifier with corresponding weight vector $\widehat{v}$
3: Query random samples $\{(X_i, y_i)\}_{i=1}^m$
4: Let $\widehat{r} = \dfrac{\sum_{i=1}^m \langle \widehat{v},\, X_i \rangle y_i}{\sum_{i=1}^m \langle \widehat{v},\, X_i \rangle^2}$.

Output: $\widehat{w} = \widehat{r} \cdot \widehat{v}$.

We have the following main result which relates the error of $\widehat{w}$ to the error of the classification algorithm $\mathcal{A}^c$.

Theorem 30. *There exists some constants C, M such that if $m > M$, the MSE of Algorithm 6 satisfies*

$$\mathbb{E}[\langle w^* - \widehat{w},\, X \rangle^2] \leq \widehat{O}\left(\frac{1}{m} + \log^2 m \cdot \varepsilon_{\mathcal{A}^c}\left(n, \frac{1}{m}\right) + \nu^2\right).$$

We defer a detailed proof to the Appendix and provide a concise proof sketch here.

Proof Sketch

We recall that, by leveraging properties of isotropic log-concave distributions, we can obtain an estimator $\widehat{v}$ such that $\|\widehat{v} - v^*\|_2 \leq \varepsilon = O\left(\varepsilon_{\mathcal{A}^c}(n, \delta)\right)$. Now let $T_i = \langle \widehat{v},\, X_i \rangle$, and we have

$$\widehat{r} = \frac{\sum_{i=1}^m T_i y_i}{\sum_{i=1}^m T_i^2} = r^* + \frac{\sum_{i=1}^m T_i r^* \langle v^* - \widehat{v},\, X_i \rangle + \varepsilon_i}{\sum_{i=1}^m T_i^2}.$$

And thus

$$\langle w^*, X\rangle - \langle \widehat{w}, X\rangle = r^*\langle v^* - \widehat{v}, X\rangle - \frac{\sum_{i=1}^m T_i r^* \langle v^* - \widehat{v}, X_i\rangle}{\sum_{i=1}^m T_i^2}\langle \widehat{v}, X\rangle + \frac{\sum_{i=1}^m T_i \varepsilon_i}{\sum_{i=1}^m T_i^2}\langle \widehat{v}, X\rangle.$$

The first term can be bounded using $\|\widehat{v} - v^*\|_2 \le \varepsilon$; for the latter two terms, using Hoeffding bounds we show that $\sum_{i=1}^m T_i^2 = O(m)$. Then by decomposing the sums in the latter two terms, we can bound the MSE. ■

Leveraging this result we can now use existing results to derive concrete corollaries for particular instantiations of the classification algorithm $\mathcal{A}^c$. For example, when $\nu = 0$ and we use passive learning, standard VC theory shows that the empirical risk minimizer has error $\varepsilon_{\text{ERM}} = O(d\log(1/\delta)/n)$. This leads to the following corollary:

Corollary 31. *Suppose that $\nu = 0$, and that we use the passive ERM classifier as $\mathcal{A}^c$ in Algorithm 6, then the output $\widehat{w}$ has MSE bounded as:*

$$\mathbb{E}[\langle w^* - \widehat{w}, X\rangle^2] \le \widehat{O}\left(\frac{1}{m} + \frac{d\log^3 m}{n}\right).$$

When $\nu > 0$, we can use other existing algorithms for either the passive/active case. We give a summary of existing results in Table 3.1, and note that (as above) each of these results can be translated in a straightforward way to a guarantee on the MSE when combining direct and comparison queries. We also note that previous results using isotropic log-concave distribution can be directly exploited in our setting since $X - X'$ follows isotropic log-concave distribution if X does [121]. Each of these results provide upper bounds on minimax risk Equation (3.7) under certain restrictions.

Table 3.1: Summary of existing results for passive/active classification for isotropic log-concave X distributions. C denotes some fixed constant; $\varepsilon = \varepsilon_{\mathcal{A}^c}(m, \delta)$. The work of [195] additionally requires that $X - X'$ follow a uniform distribution.

Algorithm	Oracle	Requirement	Rate of ε	Efficient?
ERM [85]	Passive	None	$\widetilde{O}\left(d\sqrt{\frac{1}{m}}\right)$	No
		$\nu = O(\varepsilon)$	$\widetilde{O}\left(\frac{d}{m}\right)$	
[14]	Passive	$\nu = O(\varepsilon)$	$\widetilde{O}\left(\left(\frac{d}{m}\right)^{1/3}\right)$	Yes
[13]	Active	$\nu = O(\varepsilon)$	$\exp\left(-\frac{Cm}{d+\log(m/\delta)}\right)$	Yes
[195]	Passive	$\nu = O\left(\frac{\varepsilon}{\log d + \log\log\frac{1}{\varepsilon}}\right)$	$\widetilde{O}\left(\frac{d}{m}\right)$	Yes
	Active		$\exp\left(-\frac{Cm}{d+\log(m/\delta)}\right)$	

3.4.3 Lower Bounds

Now we turn to information-theoretic lower bounds on the minimax risk (3.1). We consider *any* active estimator $\widehat{w}$ with access to the two oracles $\mathcal{O}_c, \mathcal{O}_l$, using n comparisons and m labels. In

this section, we show that the $1/m$ rate in Theorem 30 is optimal; we also show a lower bound in the active setting on the total number of queries in the appendix.

Theorem 32. *Suppose that X is uniform on $[-1,1]^d$, and $\varepsilon \sim \mathcal{N}(0,1)$. Then, for any (active) estimator $\widehat{w}$ with access to both label and comparison oracles, there is a universal constant $c > 0$ such that,*

$$\inf_{\widehat{w}} \sup_{w^*} \mathbb{E}\left[\langle w^* - \widehat{w},\, X\rangle^2\right] \geq \frac{c}{m}.$$

Theorem 32 shows that the $O(1/m)$ term in Theorem 30 is necessary. The proof uses classical information-theoretic techniques (Le Cam's method) applied to two increasing functions with $d = 1$, and is included in Appendix 3.8.7.

We note that we cannot establish lower bounds that depend solely on number of comparisons n, since we can of course achieve $O(d/m)$ MSE without using any comparisons. Consequently, we show a lower bound on the *total* number of queries in Appendix 3.8.8. This bound shows that when using the algorithm in the paper of [13] as $\mathcal{A}^c$ in CLR, the total number of queries is optimal up to log factors.

3.5 Experiment Results

To verify our theoretical results and test our algorithms in practice, we perform three sets of experiments. First, we use simulated data, where the noise in the labels and ranking can be controlled separately. Second, we consider an application of predicting people's ages from photographs, where we synthesize comparisons from data on people's apparent ages (as opposed to their true biological ages). Finally, we crowdsource data using Amazon's Mechanical Turk to obtain comparisons and direct measurements for the task of estimating rental property asking prices. We then evaluate various algorithms for predicting rental property prices, both in terms of their accuracy, as well as in terms of their time cost.

Baselines. In the nonparametric setting, we compare R^2 with k-NN algorithms in all experiments. We choose k-NN methods because they are near-optimal theoretically for Lipschitz functions, and are widely used in practice. Also, the R^2 method is a nonparametric method built on a nearest neighbor regressor. It may be possible to use the ranking-regression method in conjunction with other nonparametric regressors but we leave this for future work. We choose from a range of different constant values of k in our experiments.

In the linear regression setting, for our CLR algorithm, we consider both a passive and an active setting for comparison queries. For the passive comparison query setting, we simply use a Support Vector Machine (SVM) as $\mathcal{A}$ in Algorithm 6. For the active comparison query setting, we use an algorithm minimizing hinge loss as described in [13]. We compare CLR to the LASSO and to Linear Support Vector Regression (LinearSVR), where the relevant hyperparameters are chosen based on validation set performance. We choose LASSO and LinearSVR as they are the most prevalent linear regression methods. Unless otherwise noted, we repeat each experiment 20 times and report the average MSE[3].

[3]Our plots are best viewed in color.

Cost Ratio. Our algorithms aim at reducing the overall cost of estimating a regression function when comparisons can be more easily available than direct labels. In practice, the cost of obtaining comparisons can vary greatly depending on the task, and we consider two practical setups:

1. In many applications, both direct labels and comparisons can be obtained, but labels cost more than comparisons. Our price estimation task corresponds to this case. The cost, in this case, depends on the ratio between the cost of comparisons and labels. We suppose that comparisons cost 1, and that labels cost c for some constant $c > 1$ and that we have a total budget of C. We call c the *cost ratio*. Minimizing the risk of our algorithms requires minimizing $\mathfrak{M}(m, n; \eta)$ as defined in (3.1) subject to $cm + n \leq C$; for most cases, we need a small m and large n. In experiments with a finite cost ratio, we fix the number of direct measurements to be a small constant m and vary the number of comparisons that we use.
2. Direct labels might be substantially harder to acquire because of privacy issues or because of inherent restrictions in the data collection process, whereas comparisons are easier to obtain. Our age prediction task corresponds to this case, where it is conceivable that only some of the biological ages are available due to privacy issues. In this setting, the cost is dominated by the cost of the direct labels and we measure the cost of estimation by the number of labeled samples used.

3.5.1 Modifications to Our Algorithms

While R^2 and CLR are near optimal from a theoretical standpoint, we adopt the following techniques to improve their empirical performance:

R^2 with k-NN. Our analysis considers the case when we use 1-NN after isotonic regression. However, we empirically find that using more than 1 nearest neighbor can also improve the performance. So in our experiments, we use k-NN in the final step of R^2, where k is a small fixed constant. We note in passing that our theoretical results remain valid in this slightly more general setting.

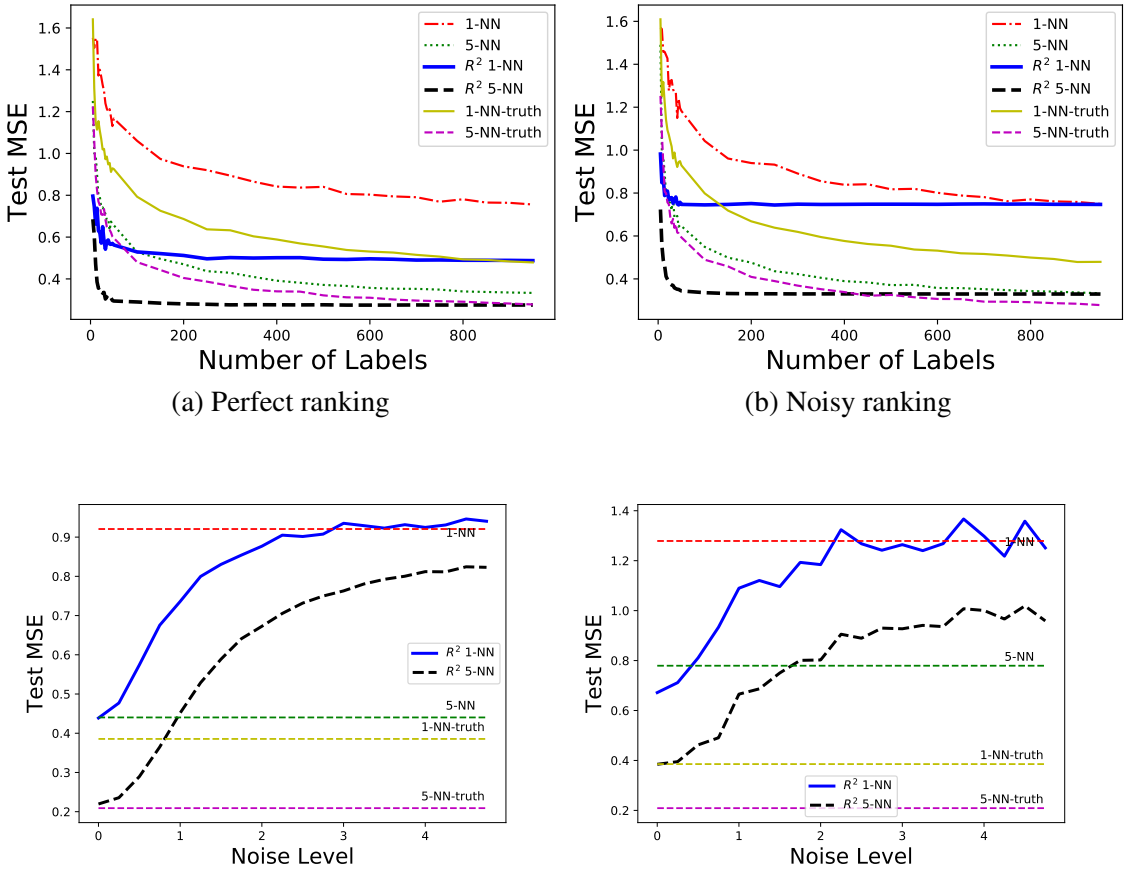
R^2 with comparisons. When R^2 uses passively collected comparisons, we would need $O(n^2)$ pairs to have a ranking with $O(1/n)$ error in the Kendall-Tau metric if we use the algorithm from [39]. We instead choose to take advantage of the feature space structure when we use R^2 with comparisons. Specifically, we build a nonparametric rankSVM [100] to score each sample using pairwise comparisons. We then rank samples according to their scores given by the rankSVM. We discuss another potential method, which uses nearest neighbors based on Borda counts, in Appendix 3.7.

CLR with feature augmentation. Using the directly labeled data only to estimate the norm of the weights corresponds to using linear regression with the direct labels with a *single* feature $\langle \widehat{v}, x \rangle$ from Algorithm 6. Empirically, we find that using all the features together with the estimated $\langle \widehat{v}, x \rangle$ results in better performance. Concretely, we use a linear SVR with input features $(x; \langle \widehat{v}, x \rangle)$, and use the output as our prediction.

3.5.2 Simulated Data

We generate different synthetic datasets for nonparametric and linear regression settings in order to verify our theory.

Simulated Data for R^2



(a) Perfect ranking

(b) Noisy ranking

(c) Effect of ranking noise, $m = 100$

(d) Effect of ranking noise, $m = 10$

Figure 3.3: Experiments on simulated data for R^2. 1-NN and 5-NN represent algorithms using noisy label data only; R^2 1-NN and R^2 5-NN use noisy labels as well as rankings; 1-NN-truth and 5-NN-truth use perfect label data only.

Data Generation. We generate simulated data following Härdle et al. [88]. We take $d = 8$, and sample X uniformly random from $[0, 1]^d$. Our target function is $f(x) = \sum_{i=1}^{d} f^{(i \bmod 4)}(x_i)$, where x_i is x's i-th dimension, and

$$f^{(1)}(x) = px - 1/2, \quad f^{(2)}(x) = px^3 - 1/3,$$
$$f^{(3)}(x) = -2\sin(-px), \quad f^{(4)}(x) = e^{-px} + e^{-1} - 1$$

with p sampled uniformly at random in $[0, 10]$. We generate a training and a test set each of size $n = 1000$ samples respectively. We rescale f so that it has 0 mean and unit variance over the

training set. This makes it easy to control the noise that we add relative to the function value. For training data, we generate the labels as $y_i = f(X_i) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 0.5^2)$, for the training data $\{X_1, \ldots, X_n\}$. At test time, we compute the MSE $\frac{1}{n}\sum_{i=1}^{n}(f(X_i^{\text{test}}) - \widehat{f}(X_i^{\text{test}}))^2$ for the test data $\{X_1^{\text{test}}, \ldots, X_n^{\text{test}}\}$.

Setup and Baselines. We test R^2 with either 1-NN or 5-NN for our simulated data, denoted as R^2 1-NN and R^2 5-NN respectively. We compare them with the following baselines: i) 1-NN and 5-NN using noisy labels (X, y) only. Since R^2 uses ordinal data in addition to labels, it should have lower MSE than 1-NN and 5-NN. ii) 1-NN and 5-NN using perfect labels $(X, f(X))$. Since these algorithms use perfect labels, when all sample points are labeled they serve as a benchmark for our algorithms.

R^2 with rankings. Our first set of experiments suppose that R^2 has access to the ranking over all 1000 training samples, while the k-NN baseline algorithms only have access to the labeled samples. We measure the cost as the number of labels in this case. Figure 3.3a compares R^2 with baselines when the ranking is perfect. R^2 1-NN and R^2 5-NN exhibited better performance than their counterparts using only labels, whether using noisy or perfect labels; in fact, we find that R^2 1-NN and R^2 5-NN perform nearly as well as 1-NN or 5-NN using all 1000 perfect labels, while only requiring around 50 labeled samples.

In Figure 3.3b, R^2 uses an input ranking obtained from noisy labels $\{y_1, ..., y_n\}$. In this case, the noise in the ranking is dependent on the label noise, since the ranking is directly derived from the noisy labels. This makes the obtained labels consistent with the ranking, and thus eliminates the need for isotonic regression in Algorithm 5. Nevertheless, we find that the ranking still provides useful information for the unlabeled samples. In this setting, R^2 outperformed its 1-NN and 5-NN counterparts using noisy labels. However, R^2 was outperformed by algorithms using perfect labels when $n = m$. As expected, R^2 and k-NN with noisy labels achieved identical MSE when $n = m$, as ranking noise is derived from noise in labels.

We consider the effect of *independent* ranking noise in Figure 3.3c. We fixed the number of labeled/ranked samples to 100/1000 and varied the noise level of ranking. For a noise level of σ, the ranking is generated from

$$y' = f(X) + \varepsilon' \tag{3.8}$$

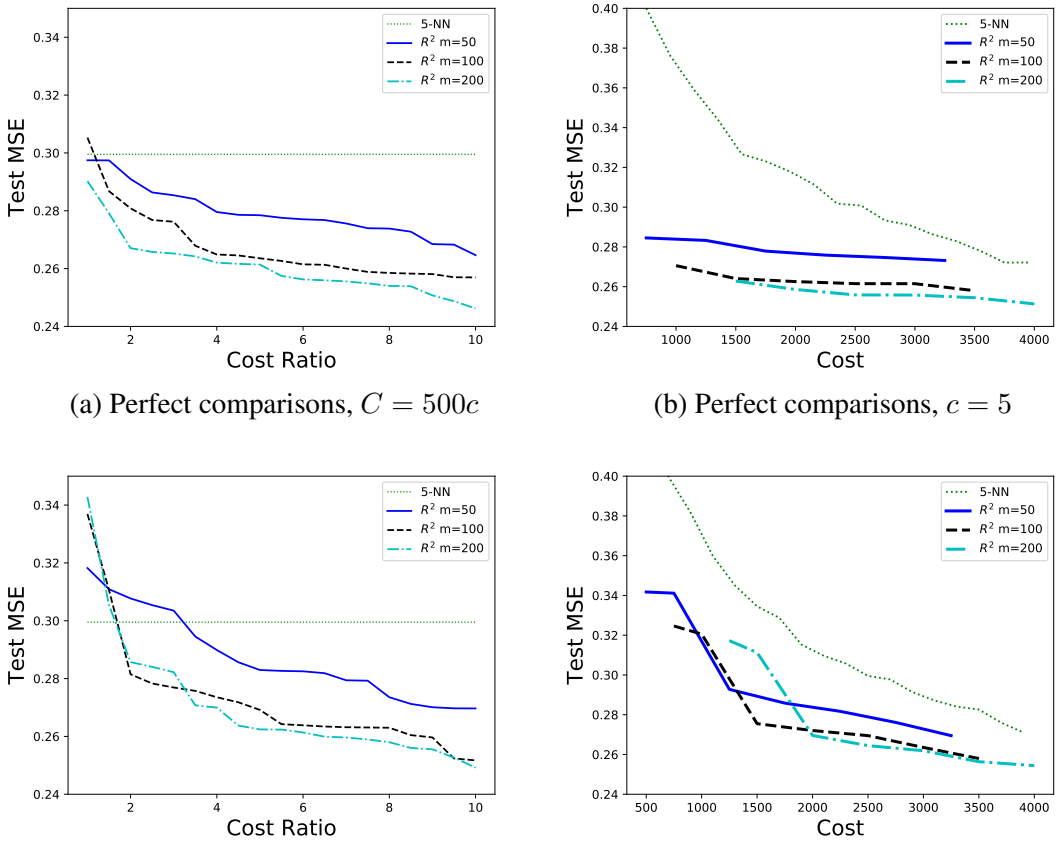
where $\varepsilon' \sim \mathcal{N}(0, \sigma^2)$. We also plot the performance of 1-NN and 5-NN using 100 noisy labels and 1,000 perfect labels for comparison. We varied σ from 0 to 5 and plotted the MSE. We repeat these experiments 50 times.

For both R^2 1-NN and 5-NN – despite the fact that they use noisy labels – their performance is close to the NN methods using noiseless labels. As σ' increases, both methods start to deteriorate, with R^2 5-NN hitting the naive 5-NN method at around $\sigma' = 1$ and R^2 1-NN at around $\sigma' = 2.5$. This shows that R^2 is robust to ranking noise of comparable magnitude to the label noise. We show in Figure 3.3d the curve when we use 10 labels and 1000 ranked samples, where a larger amount of ranking noise can be tolerated.

R^2 with comparisons. We also investigate the performance of R^2 when we have pairwise comparisons instead of a total ranking. We train a rankSVM with an RBF kernel with a bandwidth of 1, i.e. $k(x, x') = \exp(-\|x - x'\|_2^2)$, as described in Section 3.5.1. Our rankSVM has a ranking error of $\nu = 11.8\%$ on the training set and $\nu = 13.8\%$ on the validation set. For simplicity, we

only compare with 5-NN here since it gives best performance amongst the label-only algorithms. The results are depicted in Figure 3.4. When comparisons are perfect, we first investigate the effect of the cost ratio in Figure 3.4a. We fixed the budget to equal to $C = 500c$ (i.e., we would have 500 labels available if we only used labels), and each curve corresponds to a value of $m \in \{50, 100, 200\}$ and a varied n such that the total cost is $C = 500c$. We can see for almost all choices of m and cost ratio, R^2 provides a performance boost. In Figure 3.4b we fix $c = 5$, and vary the total budget C from 500 to 4,000. We find that R^2 outperforms the label-only algorithms in most setups.

In Figures 3.4c and 3.4d, we consider the same setup of experiments, but with comparisons generated from (3.8), where $\varepsilon' \sim \mathcal{N}(0, 0.5^2)$. Note that here the noise ε' is of the same magnitude but independent from the label noise. Although R^2 gave a less significant performance boost in this case, it still outperformed label-only algorithms when $c \geq 2$.



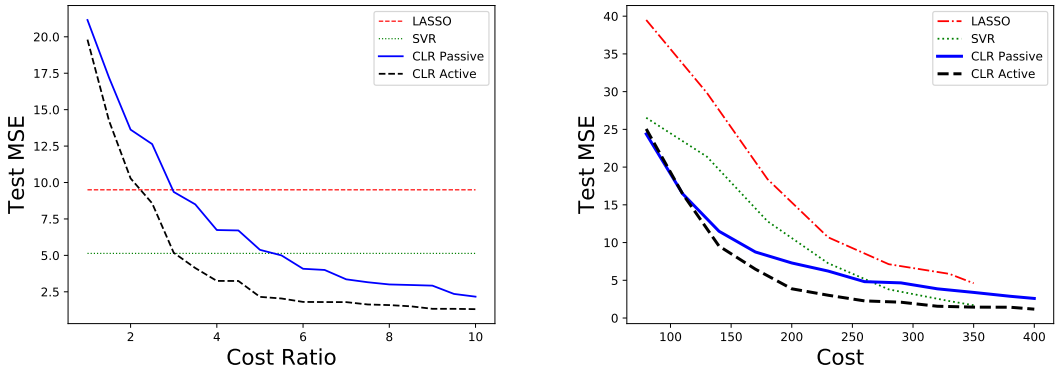
(a) Perfect comparisons, $C = 500c$

(b) Perfect comparisons, $c = 5$

(c) Noisy comparisons, $C = 500c$

(d) Noisy comparisons, $c = 5$

Figure 3.4: Experimental results on synthetic dataset for R^2 with comparisons. Each curve corresponds to a fixed m, and we vary n as the cost ratios or total budget change. Note that curves start at different locations because of different m values.



(a) Varying cost ratio c with $C = 50c$

(b) Varying budget C with $c = 5$

Figure 3.5: Experimental results on synthetic dataset for CLR.

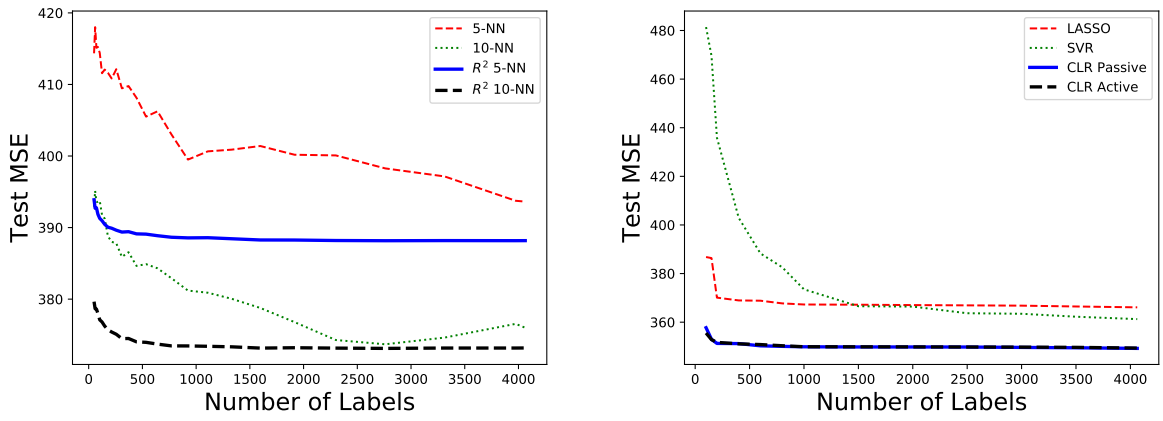
Simulated Data for CLR

For CLR we only consider the case with a cost ratio, because we find that a small number of comparisons already suffice to obtain a low error. We set $d = 50$ and generate both X and w^* from the standard normal distribution $\mathcal{N}(0, I_d)$. We generate noisy labels with distribution $\varepsilon \sim \mathcal{N}(0, 0.5^2)$. The comparison oracle generates response using the same noise model: $Z = \text{sign}((\langle w^*, x_1\rangle + \varepsilon_1) - (\langle w^*, x_2\rangle - \varepsilon_2))$ for input (x_1, x_2), with $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 0.5^2)$ independent of the label noise.

Model performances are compared in Figure 3.5. We first investigate the effect of cost ratio in Figure 3.5a, where we fixed the budget to $C = 50c$, i.e. if we only used labels we would have a budget of 50 labels. The passive comparison query version of CLR requires roughly $c > 5$ to outperform baselines, and the active comparison query version requires $c > 3$. We also experiment with a fixed cost ratio $c = 5$ and varying budget C in Figure 3.5b. The active version outperformed all baselines in most scenarios, whereas the passive version gave a performance boost when the budget was less than 250 (i.e. number of labels is restricted to less than 250 in label only setting). We note that the active and passive versions of CLR only differ in their collection of comparisons; both algorithms (along with the baselines) are given a random set of labeled samples, making it a fair competition.

3.5.3 Predicting Ages from Photographs

To further validate R^2 in practice, we consider the task of estimating people's ages from photographs. We use the APPA-REAL dataset [6], which contains 7,591 images, and each image is associated with a biological age and an apparent age. The biological age is the person's actual age, whereas the apparent ages are collected by asking crowdsourced workers to estimate their apparent ages. Estimates from (on average 38 different) labelers are averaged to obtain the apparent age. APPA-REAL also provides the standard deviation of the apparent age estimates. The images are divided into 4,063 train, 1,488 validation and 1,962 test samples, and we choose the best hyperparameters using the validation samples.



(a) R^2 (b) Linear Regression

Figure 3.6: Experimental results on age prediction.

Task. We consider the task of predicting biological age. Direct labels come from biological age, whereas the ranking is based on apparent ages. This is motivated by the collection process of many modern datasets: for example, we may have the truthful biological age only for a fraction of samples, but wish to collect more through crowdsourcing. In crowdsourcing, people give comparisons based on apparent age instead of biological age. As a consequence, in our experiments we assume additional access to a ranking that comes from the apparent ages. Since collecting crowdsourced data can be much easier than collecting the real biological ages, as discussed earlier, we define the cost as the number of direct labels used in this experiment.

Features and Models. We extract a 128-dimensional feature vector for each image using the last layer of FaceNet [147]. We rescale the features so that every $X \in [0,1]^d$ for R^2, or we centralize the feature to have zero mean and unit variance for CLR. We use 5-NN and 10-NN to compare with R^2 in this experiment. Utilizing extra ordinal information, R^2 has additional access to the ranking of apparent ages; since CLR does not use a ranking directly we provide it access to 4,063 comparisons (the same size as the training set) based on apparent ages.

Our results are depicted in Figure 3.6. The 10-NN version of R^2 gave the best overall performance amongst nonparametric methods. R^2 5-NN and R^2 10-NN both outperformed other algorithms when the number of labeled samples was less than 500. Interestingly, we observe that there is a gap between R^2 and its nearest neighbor counterparts even when $n = m$, i.e. the ordinal information continues to be useful even when all samples are labeled; this might be because the biological ages are "noisy" in the sense that they are also determined by factors not present in image (e.g., lifestyle). Similarly, for linear regression methods, our method also gets the lowest overall error for any budget of direct labels. In this case, we notice that active comparisons only have a small advantage over passive comparisons, since the comparison classifiers both converge with a sufficient number of comparisons.

3.5.4 Estimating AirBnB Listing Prices

In our third set of experiments, we consider the cost of both comparisons and direct labels. We use data of AirBnB listings and ask Amazon Mechanical Turk (AMT) workers to estimate their price. To measure the cost, we collect not only the labels and comparisons but also the time taken to answer the question. We use the time as an estimate of the cost.

Data Collection. Our data comes from Kaggle[4] and contains information about AirBnB postings in Seattle. We use 357 listings as the training set and 93 as the test set. We additionally pick 50 listings from the training set as validation data to select hyperparameters. We use the following features for our experiments:

(1) Host response rate, (2) Host acceptance rate, (3) Host listings count,
(4) Number of reviews per month, (5) Number of bathrooms, (6) Number of bedrooms,
(7) Number of beds, (8) Number of reviews, (9) Review scores rating,
(10) Number of people the house accommodates.

Each worker from AMT is asked to do either of the following two tasks: i) given the description of an AirBnB listing, estimate the per-night price on a regular day in 2018; ii) given the description of two AirBnB listings, select the listing that has a higher price. We collect 5 direct labels for each data point in the training set and 9 labels for each data point in the test set. For comparisons, we randomly draw 1,841 pairs from the training set and ask 2 workers to compare their prices.

Tasks. We consider two tasks, motivated by real-world applications.

1. In the first task, the goal is to predict the real listing price. This is motivated by the case where collecting the real price might be difficult to obtain or involves privacy issues. We assume that our training data, including both labels and comparisons, comes from AMT workers.

2. In the second task, the goal is to predict the user-estimated price. This can be of particular interest to AirBnB company and house-owners, for deciding the best price of the listing. We do not use the real prices in this case; we use the average of 10 worker estimated prices for each listing in the test set as the ground truth label, and the training data also comes from our AMT results.

Raw Data Analysis. Before we proceed to the regression task, we analyze the workers' performance for both tasks based on raw data in Table 3.2. Our first goal is to compare a pairwise comparison to an induced comparison, where the induced comparison is obtained by making two consecutive direct label queries and subtracting them. Similar to [150], we observe that comparisons are more accurate than the induced comparisons.

We first convert labels into pairwise comparisons by comparing individual direct labels: namely, for each obtained labeled sample pair $(x_1, y_1), (x_2, y_2)$ where y_1, y_2 are the *raw* labels from workers, we create a pairwise comparison that corresponds to comparing (x_1, x_2) with label being $\text{sign}(y_1 - y_2)$. We then compute the error rate of raw and label-induced comparisons for both Task 1 and 2. For Task 1, we directly compute the error rate w.r.t. the true listing price. For Task 2, we do not have the ground truth user prices; we instead follow the approach of [150] to compute the fraction of disagreement between comparisons. Namely, in either the raw or label-induced setting, for every pair of samples (x_i, x_j) we compute the majority of labels z_{ij}

[4] `https://www.kaggle.com/AirBnB/seattle/home`

based on all comparisons on (x_i, x_j). The disagreement on (x_i, x_j) is computed as the fraction of comparisons that disagrees with z_{ij}, and we compute the overall disagreement by averaging over all possible (x_i, x_j) pairs.

If an ordinal query is equivalent to two consecutive direct queries and subtracting the labels, we would expect a similar accuracy/disagreement for the two kinds of comparisons. However our results in Table 3.2 show that this is not the case: direct comparison queries have better accuracy for Task 1, as well as a lower disagreement within collected labels. This shows that a comparison query cannot be replaced by two consecutive direct queries. We do not observe a large difference in the average time to complete a query in Table 3.2; however the utility of comparisons in predicting price can be higher since they yield information about two labels. Further analysis of the raw data is given in Appendix 3.7.

Performance	Comparisons	Labels
Task 1 Error	**31.3%**	41.3%
Task 2 Disagreement	**16.4%**	29.5%
Average Time	64s	**63s**

Table 3.2: Performance of comparisons versus labels for both tasks.

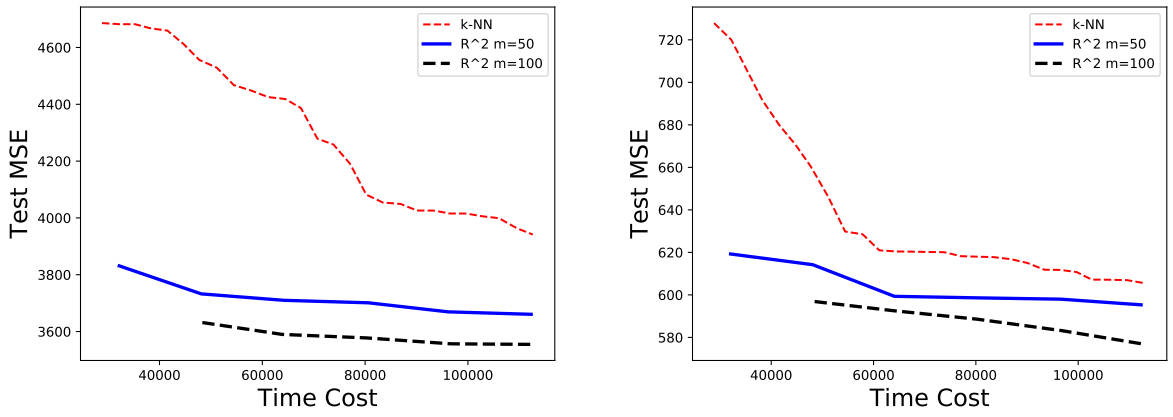
Results. We plot the experimental results in Figure 3.7. For nonparametric regression, R^2 had a significant performance boost over the best nearest neighbor regressor under the same total worker time, especially for Task 1. For Task 2, we observe a smaller improvement, but R^2 is still better than pure NN methods for all total time costs.

For linear regression, we find that the performance of CLR varies greatly with m (number of labels), whereas its performance does not vary as significantly with the number of comparisons. In fact, the errors of both CLR passive and active already plateau with a mere 50 comparisons, since the dimension of data is small ($d = 10$). So deviating from our previous experiments, in this setting, we vary the number of labels in Figure 3.7c and 3.7d. As in the nonparametric case, CLR also outperforms the baselines in both tasks. For Task 1, the active and passive versions of CLR perform similarly, whereas active queries lead to a moderate performance boost on Task 2. This is probably because the error on Task 2 is much lower than that on Task 1 (see Table 3.2), and active learning typically has an advantage over passive learning when the noise is not too high.

3.6 Conclusion

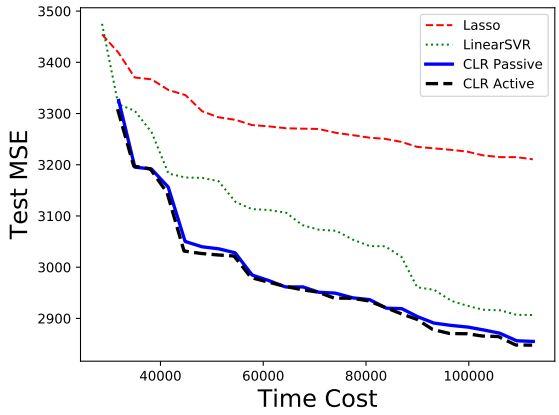
We design (near) minimax-optimal algorithms for nonparametric and linear regression using additional ordinal information. In settings where large amounts of ordinal information are available, we find that limited direct supervision suffices to obtain accurate estimates. We provide complementary minimax lower bounds, and illustrate our proposed algorithms on real and simulated datasets. Since ordinal information is typically easier to obtain than direct labels, one might expect in these favorable settings the R^2 algorithm to have lower effective cost than an algorithm based purely on direct supervision.

Several directions exist for future work. On nonparametric regression side, it remains to extend our results to the case where the Hölder exponent $s > 1$. In this setting the optimal rate

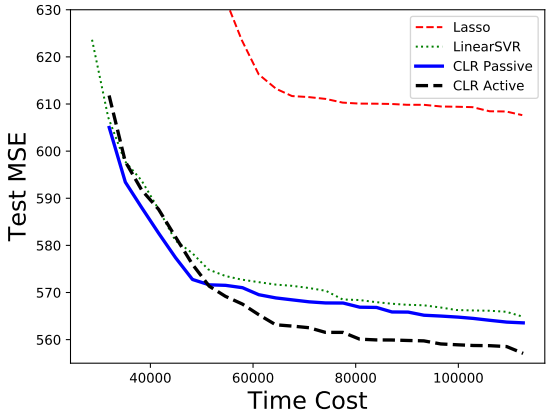


(a) Nonparametric, Task 1

(b) Nonparametric, Task 2



(c) Linear, Task 1



(d) Linear, Task 2

Figure 3.7: Results for AirBnB price estimation. In (a)(b), each curve has a fixed m with varied n; for (c)(d), each curve uses only 50 comparisons with a varied number of labels. For Figure (d), LASSO performs much worse than a LinearSVR and we only show part of the curve.

$O\left(m^{-\frac{-2s}{2s+d}}\right)$ can be faster than the convergence rate of isotonic regression, which can make our algorithm sub-optimal. It is also important to address the setting where both direct and ordinal supervision are actively acquired. For linear regression, an open problem is to consider the bounded noise model for comparisons. Our results can be easily extended to the bounded noise case using the algorithm in [85], however that algorithm is computationally inefficient. The best efficient active learning algorithm in this bounded noise setting [14] requires $m \geq O\left(d^{O(\frac{1}{(1-2\lambda)^4})}\right)$ comparisons, and a large gap remains between what can be achieved in the presence and absence of computational constraints.

Motivated by practical applications in crowdsourcing, we list some further extensions to our results:

Partial orders: In this paper, we focus on ordinal information either in the form of a total

ranking or pairwise comparisons. In practice, ordinal information might come in the form of partial orders, where we have several subsets of unlabeled data ranked, but the relation between these subsets is unknown. A straightforward extension of our results in the nonparametric case leads to the following result: if we have k (partial) orderings, each with $n_1, \ldots, n_k$ samples, and $m_1, \ldots, m_k$ samples in each ordering are labeled, we can show an upper bound on the MSE of $m_1^{-2/3} + \cdots + m_k^{-2/3} + (n_1 + \cdots + n_k)^{-2s/d}$. It would be interesting to study the optimal rate, as well as to consider other smaller partial orders.

Other models for ordinal information: Beyond the bounded noise model for comparison, we can consider other pairwise comparison models, like Plackett-Luce [122, 137] and Thurstone [162]. These parametric models can be quite restrictive and can lead to unnatural results that we can recover the function values even *without* querying any direct labels (see for example [150]). One might also consider pairwise comparisons with Tsybakov-like noise [165] which have been studied in the classification setting [187]; the main obstacle here is the lack of computationally-efficient algorithms that aggregate pairwise comparisons into a complete ranking under this noise model.

Other classes of functions: Several recent papers [31, 32, 45, 83] demonstrate the adaptivity (to "complexity" of the unknown parameter) of the MLE in shape-constrained problems. Understanding precise assumptions on the underlying smooth function which induces a low-complexity isotonic regression problem is interesting future work.

3.7 Additional Experimental Results

Relation between true and user estimated price. Figure 3.8 shows a scatter plot of the true listing prices of AirBnB data with respect to the user estimated prices. Although the true prices is linearly correlated with the user prices (with p-value $6e-20$), the user price is still very different from true price even if we take the average of 5 labelers. The average of all listings' true price is higher than the average of all user prices by 25 dollars, partially explaining the much higher error when we use user prices to estimate true prices.

An Alternative to using RankSVM. As an alternative to training RankSVM, we can also use nearest neighbors on Borda counts to take into account the structure of feature space: for each sample x, we use the score $s(x) = \frac{1}{k}\sum_{x' \in k\text{-NN}(x)} \text{Borda}(x')$, where k-NN(x) is the k-th nearest neighbor of x in the feature space, including x itself. The scores are then used to produce a ranking. When c is large, this method does provide an improvement over the label-only baselines, but generally does not perform as well as our rankSVM method. The results when cost ratio $c = 10$ and comparisons are perfect are depicted in Figure 3.9. We use $k = 25$ for deciding the nearest neighbors for Borda counts, and 5-NN as the final prediction step. While using R^2 with Borda counts do provide a gain over label-only methods, the improvement is less prominent than using rankSVM.

For estimating AirBnB price the results are shown in Figure 3.10. For task 1, NN of Borda counts introduces an improvement similar to (or less than) RankSVM, but for task 2 it is worse than nearest neighbors. We note that for task 1, the best number of nearest neighbors of Borda counts is 50, whereas for task 2 it is 5 (close to raw Borda counts). We suspect this is due to the larger noise in estimating true price, however a close examination for this observation remains as

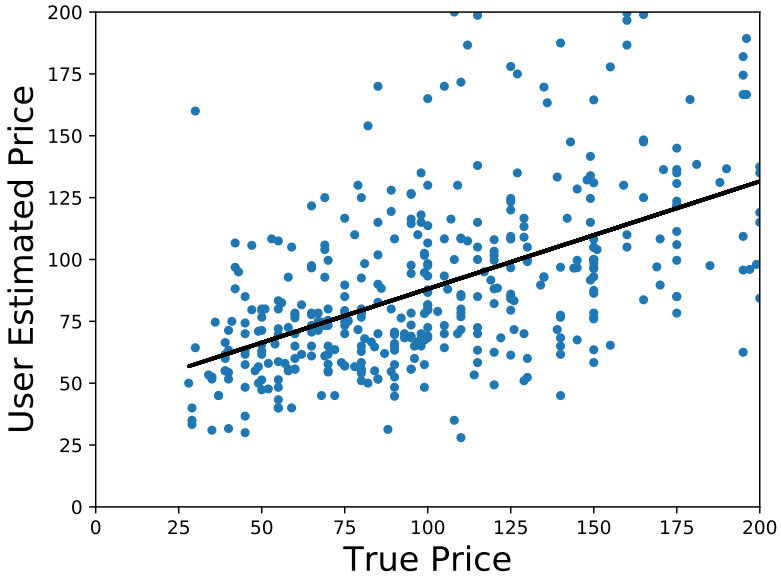


Figure 3.8: Scatter plot of true prices w.r.t average user estimated prices, along with the ordinary least square result. We only show prices smaller than 200 to make the relation clearer.

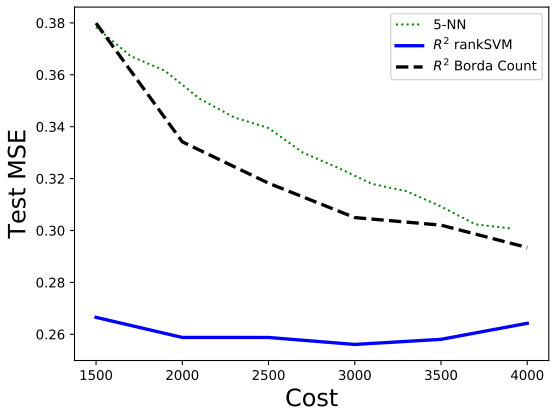
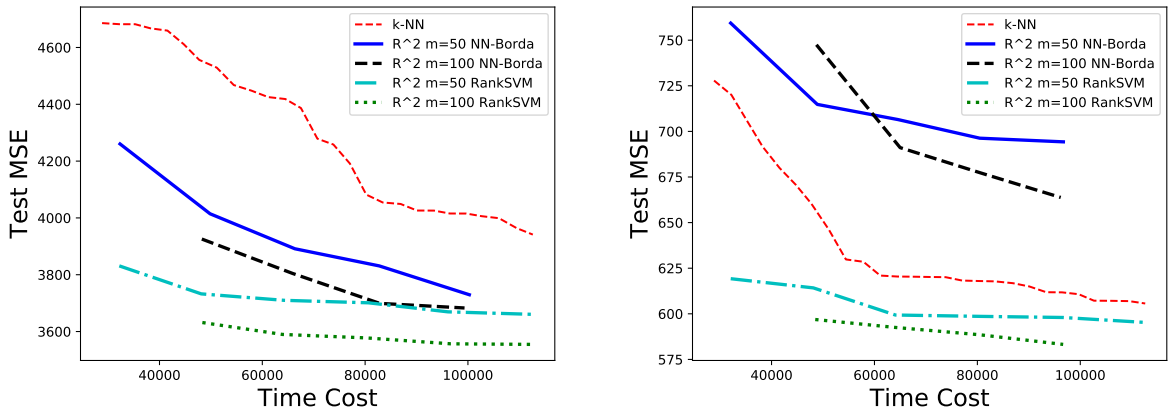


Figure 3.9: Experimental results on synthetic dataset for R^2 with comparisons, using nearest neighbor with Borda counts. Both R^2 algorithms uses 5-NN as the final prediction step.

future work.



(a) Task 1 (b) Task 2

Figure 3.10: Experiments on AirBnB price estimation for nonparametric methods, using nearest neighbors of Borda count. Figure (a) uses 50-NN for averaging Borda counts, while Figure (b) uses 5-NN.

3.8 Detailed Proofs

3.8.1 Proof of Theorem 22

Without loss of generality we assume throughout the proof that we re-arrange the samples so that the true ranking of the samples π is the identity permutation, i.e. that $f(X_1) \leq f(X_2) \leq \ldots \leq f(X_n)$. We let $C, c, C_1, c_1, \ldots$ denote universal positive constants. As is standard in nonparametric regression these constants may depend on the dimension d but we suppress this dependence.

For a random point $X \in \mathcal{X}$, let X_α be the nearest neighbor of X in the labeled set $\mathcal{L}$. We decompose the MSE as

$$\mathbb{E}\left[(\widehat{f}(X) - f(X))^2\right] \leq 2\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] + 2\mathbb{E}\left[(f(X_\alpha) - f(X))^2\right].$$

Under the assumptions of the theorem we have the following two results which provide bounds on the two terms in the above decomposition.

Lemma 33. *For a constant $C > 0$ we have that,*

$$\mathbb{E}\left[(f(X_\alpha) - f(X))^2\right] \leq Cn^{-2s/d}.$$

Lemma 34. *For any $0 < \delta \leq 1/2$ we have that there is a constant $C > 0$ such that:*

$$\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] \leq 4\delta M^2 + \frac{C\log(m/\delta)\log n \log(1/\delta)}{m}\left[\sum_{k=1}^{m}\left(\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right]\right) + \sum_{k=0}^{m}\left(\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2\right]\right)\right].$$

Taking these results as given we can now complete the proof of the theorem. We first note that the first term in the upper bound in Lemma 34 is simply the MSE in an isotonic regression problem, and using standard risk bounds for isotonic regression (see for instance, Theorem 2.2 in Zhang [203]) we obtain that for a constant $C > 0$:

$$\sum_{k=1}^{m} \mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right] \leq Cm^{2/3}.$$

Furthermore, since $f(X_{t_{m+1}}) - f(X_{t_0}) \leq 2M$, and the function values are increasing we obtain that:

$$\sum_{k=0}^{m} \left(\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2\right]\right) \leq 4M^2.$$

Now, choosing $\delta = \max\{n^{-2s/d}, 1/m\}$ we obtain:

$$\mathbb{E}\left[(\widehat{f}(X) - f(X))^2\right] \leq C_1 m^{-2/3} \log^2 n \log m + C_2 n^{-2s/d},$$

as desired. We now prove the two technical lemmas to complete the proof.

Proof of Lemma 33

The proof of this result is an almost immediate consequence of the following result from [82].

Lemma 35 ([82], Lemma 6.4 and Exercise 6.7)**.** *Suppose that there exist positive constants $p_{\min}$ and p_{max} such that $p_{\min} \leq p(x) \leq p_{\max}$. Then, there is a constant $c > 0$, such that*

$$\mathbb{E}[\|X_\alpha - X\|_2^2] \leq cn^{-2/d}.$$

Using this result and the Hölder condition we have

$$\begin{aligned}
\mathbb{E}\left[(f(X_\alpha) - f(X))^2\right] &\leq L\mathbb{E}\left[\|X_\alpha - X\|_2^{2s}\right] \\
&\overset{\text{(i)}}{\leq} L\left(\mathbb{E}\left[\|X_\alpha - X\|_2^2\right]\right)^s \\
&\leq cn^{-2s/d},
\end{aligned}$$

where (i) uses Jensen's inequality. We now turn our attention to the remaining technical lemma.

Proof of Lemma 34

We condition on a certain favorable configuration of the samples that holds with high-probability. For a sample $\{X_1, \ldots, X_n\}$ let us denote by

$$q_i := \mathbb{P}(X_\alpha = X_i),$$

where X_α is the nearest neighbor of X. Furthermore, for each k we recall that since we have re-arranged the samples so that π is the identity permutation we can measure the distance between adjacent labeled samples in the ranking by $t_k - t_{k+1}$. The following result shows that the labeled samples are roughly uniformly spaced (up to a logarithmic factor) in the ranked sequence, and that each point X_i is roughly equally likely (up to a logarithmic factor) to be the nearest neighbor of a randomly chosen point.

Lemma 36. *There is a constant $C > 0$ such that with probability at least $1 - \delta$ we have that the following two results hold:*

1. *We have that,*

$$\max_{1 \le j \le n} q_i \le \frac{Cd \log(1/\delta) \log n}{n}. \tag{3.9}$$

2. *Let us take $t_{m+1} := n + 1$, then*

$$\max_{k \in [m+1]} t_k - t_{k-1} \le \frac{Cn \log(m/\delta)}{m}. \tag{3.10}$$

Denote the event, which holds with probability at least $1 - \delta$ in the above Lemma by $\mathcal{E}_0$. By conditioning on $\mathcal{E}_0$ we obtain the following decomposition:

$$\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] \le \mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2 | \mathcal{E}_0\right] + \delta \cdot 4M^2$$

because both f and $\widehat{f}$ are bounded in $[-M, M]$. We condition all calculations below on $\mathcal{E}_0$. Now we have

$$\begin{aligned}
\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2 | \mathcal{E}_0\right] &= \sum_{i=1}^{n} \mathbb{P}[X_\alpha = X_i | \mathcal{E}_0] \mathbb{E}\left[(\widehat{f}(X_i) - f(X_i))^2 | \mathcal{E}_0\right] \\
&\le \sum_{i=1}^{n} \max_{1 \le j \le n} \mathbb{P}[X_\alpha = X_j | \mathcal{E}_0] \mathbb{E}\left[(\widehat{y}_i - f(X_i))^2 | \mathcal{E}_0\right] \\
&\le \frac{Cd \log(1/\delta) \log n}{n} \sum_{i=1}^{n} \mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right], \qquad (3.11)
\end{aligned}$$

where we recall that $\widehat{y}_{\widetilde{i}}$ (defined in Algorithm 5) denotes the de-noised (by isotonic regression) y value at the nearest labeled left-neighbor of the point X_i.

For convenience we define $f(X_{t_0}) = 0$ (equivalently as in Algorithm 5 we are assigning a 0 value to any point with no labeled point with a smaller function value according to the permutation

$\widehat{\pi}$). With this definition in place we have that,

$$
\begin{aligned}
&\sum_{i=1}^{n} \mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right] \\
\leq &\sum_{i=1}^{n} \left(2\mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_{\widetilde{i}}))^2 | \mathcal{E}_0\right] + 2\mathbb{E}\left[(f(X_i) - f(X_{\widetilde{i}}))^2 | \mathcal{E}_0\right]\right) \\
= &\sum_{i=1}^{n}\sum_{k=0}^{m} \mathbb{I}[\widetilde{i} = t_k] \left(2\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right] \mathbb{I}[k \neq 0] + 2\mathbb{E}\left[(f(X_i) - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) \\
\overset{\text{(i)}}{\leq} &\sum_{i=1}^{n}\sum_{k=0}^{m} \mathbb{I}[\widetilde{i} = t_k] \left(2\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right] \mathbb{I}[k \neq 0] + 2\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) \\
\overset{\text{(ii)}}{=} &\sum_{k=0}^{m} \left(2\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right] \mathbb{I}[k \neq 0] + 2\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) \sum_{i=1}^{n} \mathbb{I}[\widetilde{i} = t_k] \\
\overset{\text{(iii)}}{\leq} &\frac{Cn\log(m/\delta)}{m} \sum_{k=1}^{m} \left(2\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) + \frac{Cn\log(m/\delta)}{m} \sum_{k=0}^{m} \left(2\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) \\
\leq &\frac{2Cn\log(m/\delta)}{m} \left[\sum_{k=1}^{m} \left(\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right) + \sum_{k=0}^{m} \left(\mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2 | \mathcal{E}_0\right]\right)\right].
\end{aligned}
$$

The inequality (i) follows by noticing that if $\widetilde{i} = t_k$, $f(X_i) - f(X_{\widetilde{i}})$ is upper bounded by $f(X_{t_{k+1}}) - f(X_{t_k})$. We interchange the order of summations to obtain the inequality (ii). The inequality in (iii) uses Lemma 36. We note that,

$$
\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2 | \mathcal{E}_0\right] \leq \frac{\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right]}{\mathbb{P}(\mathcal{E}_0)} \leq 2\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right],
$$

since $\delta \leq 1/2$, and a similar manipulation for the second term yields that,

$$
\sum_{i=1}^{n} \mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right] \leq \frac{4Cn\log(m/\delta)}{m} \left[\sum_{k=1}^{m} \mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right] + \sum_{k=0}^{m} \mathbb{E}\left[(f(X_{t_{k+1}}) - f(X_{t_k}))^2\right]\right].
$$

Plugging this expression back in to Equation (3.11) we obtain the Lemma. Thus, to complete the proof it only remains to establish the result in Lemma 36.

Proof of Lemma 36

We prove each of the two results in turn.

Proof of Inequality Equation (3.9): As a preliminary, we need the following Vapnik-Cervonenkis result from [46]:

Lemma 37. *Suppose we draw a sample $\{X_1, \ldots, X_n\}$, from a distribution $\mathbb{P}$, then there exists a universal constant C' such that with probability $1 - \delta$, every ball B with probability:*

$$
\mathbb{P}(B) \geq \frac{C'\log(1/\delta)d\log n}{n},
$$

contains at least one of the sample points.

We now show that under this event we have

$$\max_i q_i \leq \frac{C' p_{\max} \log(1/\delta) d \log n}{p_{\min} n}.$$

Fix any point $X_i \in T$, and for a new point X, let $r = \|X_i - X\|_2$. If X_i is X's nearest neighbor in T, there is no point in the ball $B(X, r)$. Comparing this with the event in Lemma 37 we have

$$p_{\min} v_d r^d \leq \frac{C' \log(1/\delta) d \log n}{n},$$

where v_d is the volume of the unit ball in d dimension.

Hence we obtain an upper bound on r. Now since $p(x)$ is upper and lower bounded we can bound the largest q_i as

$$\max_i q_i \leq p_{\max} v_d r^d \leq \frac{C' p_{\max} \log(1/\delta) d \log n}{p_{\min} n}.$$

Thus we obtain the inequality Equation (3.9).

Proof of Inequality Equation (3.10): Recall that we define $t_{m+1} := n+1$. Notice that $t_1, \ldots, t_m$ are randomly chosen from $[n]$. So for each $k \in [m]$ we have

$$\mathbb{P}[t_k - t_{k-1} \geq t] \leq \frac{n-t+1}{n} \left(\frac{n-t}{n}\right)^{m-1} \leq \left(\frac{n-t}{n}\right)^{m-1},$$

since we must randomly choose t_k in $X_t, X_{t+1}, \ldots, X_n$, and choose the other $m-1$ samples in $X_1, \ldots, X_{t_k - t}, X_{t_k+1}, \ldots, X_n$. Similarly we also have

$$\mathbb{P}[t_{m+1} - t_m \geq t] \leq \left(\frac{n-t}{n}\right)^{m-1}.$$

So

$$\begin{aligned}\mathbb{P}[\max_{k \in [m+1]} t_k - t_{k-1} \geq t] &\leq \sum_{k=1}^{m+1} \mathbb{P}[\max_{k \in [m+1]} t_k - t_{k-1} \geq t] \\ &\leq (m+1) \left(\frac{n-t}{n}\right)^{m-1}.\end{aligned}$$

Setting this to be less than or equal to δ, we have

$$\frac{t}{n} \geq 1 - \left(\frac{\delta}{m+1}\right)^{\frac{1}{m-1}}.$$

Let $u = \log\left(1 - \left(\frac{\delta}{m+1}\right)^{\frac{1}{m-1}}\right) = -C\frac{\log(m/\delta)}{m}$, we have $1 - e^u = O(-u)$ since u is small and bounded. So it suffices for $t \geq C\frac{n \log(m/\delta)}{m}$ such that

$$\mathbb{P}[\max_{k \in [m+1]} t_k - t_{k-1} \geq t] \leq \delta.$$

3.8.2 Proof of Theorem 23

To prove the result we separately establish lower bounds on the size of the labeled set of samples m and the size of the ordered set of samples n. Concretely, we show the following pair of claims, for a positive constant $C > 0$,

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}_{s,L}} \mathbb{E}\left[(f(X) - \widehat{f}(X))^2\right] \geq C m^{-2/3} \tag{3.12}$$

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}_{s,L}} \mathbb{E}\left[(f(X) - \widehat{f}(X))^2\right] \geq C n^{-2s/d}. \tag{3.13}$$

We prove each of these claims in turn and note that together they establish Theorem 23.

Proof of Claim Equation (3.12): We establish the lower bound in this case by constructing a suitable packing set of functions, and using Fano's inequality. The main technical novelty, that allows us to deal with the setting where both direct and ordinal information is available, is that we construct functions that are all increasing functions of the first covariate x_1, and for these functions the ordinal measurements provide no additional information.

Without loss of generality we consider the case when $d = 1$, and note that the claim follows in general by simply extending our construction using functions for which $f(x) = f(x_1)$. We take the covariate distribution $\mathbb{P}_{\mathcal{X}}$ to be uniform on $[0, 1]$. For a kernel function K that is 1-Lipschitz on $\mathbb{R}$, bounded and supported on $[-1/2, 1/2]$, with

$$\int_{-1/2}^{1/2} K^2(x)dx > 0$$

we define:

$$\begin{aligned} u &= \lceil m^{\frac{1}{3}} \rceil, \quad h = 1/u, \\ x_k &= \frac{k - 1/2}{u}, \quad \phi_k(x) = \frac{Lh}{2} K\left(\frac{x - x_k}{h}\right), \quad \text{for} \quad k = \{1, 2, \ldots, u\}, \\ \Omega &= \{\omega : \omega \in \{0, 1\}^u\}. \end{aligned}$$

With these definitions in place we consider the following class of functions:

$$\mathcal{G} = \left\{ f_\omega : [0, 1] \mapsto \mathbb{R} : f_\omega(x) = \frac{Lx}{2} + \sum_{i=1}^{k} \omega_i \phi_i(x), \text{ for } x \in [0, 1] \right\}.$$

We note that the functions in $\mathcal{G}$ are L-Lipschitz, and thus satisfy the Hölder constraint (for any $0 < s \leq 1$).

We note that these functions are all increasing so the permutation π contains no additional information and can be obtained simply by sorting the samples (according to their first coordinate). Furthermore, in this case the unlabeled samples contribute no additional information as their distribution $\mathbb{P}_{\mathcal{X}}$ is known (since we take it to be uniform). Concretely, for any estimator in our setup which uses $\{(X_1, y_1), \ldots, (X_m, y_m), X_{m+1}, \ldots, X_n, \pi\}$ with

$$\sup_{f \in \mathcal{G}} \mathbb{E}\left[(f(X) - \widehat{f}(X))^2\right] < C m^{-2/3},$$

we can construct an equivalent estimator that uses only $\{(X_1, y_1), \ldots, (X_m, y_m)\}$. In particular, we can simply augment the sample by sampling $X_{m+1}, \ldots, X_n$ uniformly on $[0, 1]$ and generating π by ranking X in increasing order.

In light of this observation, in order to complete the proof of Claim Equation (3.12) it sufficies to show that $Cm^{-2/3}$ is a lower bound for estimating functions in $\mathcal{G}$ with access to only noisy labels. For any pair $\omega, \omega' \in \Omega$ we have that,

$$\begin{aligned}\mathbb{E}[(f_\omega(X) - f_{\omega'}(X))^2] &= \sum_{k=1}^{u} (\omega_k - \omega'_k)^2 \int \phi_k^2(x)dx \\ &= L^2 h^3 \|K\|_2^2 \rho(\omega, \omega'),\end{aligned}$$

where $\rho(\omega, \omega')$ denotes the hamming distance between x and x'.

Denote by P_0 the distribution induced by the function f_0 with $\omega = (0, \ldots, 0)$, with the covariate distribution being uniform on $[0, 1]$. We can upper bound the KL divergence between the distribution induced by any function in $\mathcal{G}$ and P_0 as:

$$\begin{aligned}\text{KL}(P_j^m, P_0^m) &= m \int_{\mathcal{X}} p(x) \int_{\mathbb{R}} p_j(y|x) \log \frac{p_0(y|x)}{p_j(y|x)} dydx \\ &= m \int_{\mathcal{X}} p(x) \sum_{i=1}^{u} \omega_i^j \phi_i^2(x)dx \\ &\leq mL^2 h^3 \|K\|_2^2 u.\end{aligned}$$

Now, the Gilbert-Varshamov bound [80, 168], ensures that if $u > 8$, then there is a subset $\Omega' \subseteq \Omega$ of cardinality $2^{u/8}$, such that the Hamming distance between each pair of elements $\omega, \omega' \in \Omega'$ is at least $u/8$. A straightforward application of Fano's inequality (see for instance Theorem 2.5 in [166]) shows that for small constants $c, c' > 0$, if:

$$mL^2 h^3 \|K\|_2^2 u \leq cu,$$

then the error of any estimator is lower bounded as:

$$\sup_{f \in \mathcal{G}} \mathbb{E}[(\widehat{f}(X) - f(X))^2] \geq c' L^2 h^3 \|K\|_2^2 u \geq c' m^{-2/3},$$

establishing the desired claim.

Proof of Claim Equation (3.13): We show this claim by reducing to the case where we have n points with noiseless evaluations, i.e., we observe $\{(X_1, f(X_1)), (X_2, f(X_2)), \ldots, (X_n, f(X_n))\}$. We notice that the ranking π provides no additional information when all the points are labeled without noise. Formally, if we have an estimator $\widehat{f}$ which when provided $\{(X_1, y_1), \ldots, (X_m, y_m), X_{m+1}, \ldots, X_n,$ obtains an error less than $cn^{-2s/d}$ (for some sufficiently small constant $c > 0$) then we can also use this estimator in setting where all n samples are labeled without noise, by generating π according to the noiseless labels, adding Gaussian noise to the labels of m points, and eliminating the remaining labels before using the estimator $\widehat{f}$ on this modified sample.

It remains to show that, $cn^{-2s/d}$ is a lower bound on the MSE of any estimator that receives n noiseless labels. To simplify our notation we will assume that $(2n)^{1/d}$ is an integer (if not we can

establish the lower bound for a larger sample-size for which this condition is true, and conclude the desired lower bound with an adjustment of various constants). For a given sample size n, we choose

$$h = (2n)^{-1/d},$$

and consider the grid with $2n$ cubes with side-length h. Denote the centers of the cubes as $\{x_1, \ldots, x_{2n}\}$. For a kernel function K supported on $[-1/2, 1/2]^d$, which is 1-Lipschitz on $\mathbb{R}^d$, bounded and satisfies:

$$\int_{[-1/2,1/2]^d} K^2(x)dx > 0$$

we consider a class of functions:

$$\mathcal{G} = \left\{ f_\omega : f_\omega(x) = Lh^s \sum_{i=1}^{2n} \omega_i K\left(\frac{x - x_i}{h}\right), \text{for } \omega \in \{0,1\}^{2n} \right\}.$$

We note that these functions are all in the desired Hölder class with exponent s. Given n samples (these may be arbitrarily distributed) $\{(X_1, f(X_1)), \ldots, (X_n, f(X_n))\}$, we notice that we are only able to identify at most n of the ω_i (while leaving at least n of the ω_i completely unconstrained) and thus any estimator $\widehat{f}$ must incur a large error on at least one of the functions f_ω consistent with the obtained samples. Formally, we have that

$$\sup_{f \in \mathcal{G}} \mathbb{E}(\widehat{f}(X) - f(X))^2 \geq \frac{nL^2 h^{2s+d} \|K\|_2^2}{4} \geq cn^{-2s/d},$$

as desired. This completes the proof of Claim Equation (3.13).

3.8.3 Proof of Theorem 24

Throughout this proof without loss of generality we re-arrange the samples so that the estimated permutation $\widehat{\pi}$ is the identity permutation. To simplify the notation further, we let $X_{(i)} = X_{\pi^{-1}(i)}$ be the i-th element according to true permutation π. This leads to $f(X_{(1)}) \leq f(X_{(2)}) \leq \cdots \leq f(X_{(n)})$.

We begin with a technical result that bounds the error of using $\widehat{\pi}$ instead of the true permutation π in the R^2 algorithm.

The first part of the proof is the same as that of Theorem 22. We have

$$\begin{aligned}
\mathbb{E}\left[(\widehat{f}(X) - f(X))^2\right] &\leq 2\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] + 2\mathbb{E}\left[(f(X_\alpha) - f(X))^2\right] \\
&\leq 2\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] + Cn^{-2s/d}.
\end{aligned}$$

And for event $\mathcal{E}_0$ we have (note that $\widetilde{i}$ is the nearest neighbor index defined in Algorithm 5)

$$\begin{aligned}
\mathbb{E}\left[(\widehat{f}(X) - f(X_\alpha))^2\right] &\leq C\frac{d\log(1/\delta)\log n}{n} \sum_{i=1}^{n} \mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right] + \delta \\
&\leq C\left(\frac{\log^2 n}{n} \sum_{i=1}^{n} \mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right] + n^{-2s/d}\right). \qquad (3.14)
\end{aligned}$$

The second inequality is obtained by letting $\delta = n^{-2s/d}$. To bound the sum of expectations above, we first prove a lemma bounding the difference between $X_1, \ldots, X_n$ and $X_{(1)}, \ldots, X_{(n)}$:

Lemma 25(Restated). Suppose the ranking $(X_1, \ldots, X_n)$ is of at most ν error with respect to the true permutation. Then

$$\sum_{i=1}^{n} (f(X_i) - f(X_{(i)})^2 \leq 8M^2\sqrt{2\nu}n.$$

Proof. Let $\theta_i = f(X_i)$ and $\theta_{(i)} = f(X_{(i)})$, and let $g(\theta_1, \ldots, \theta_n) = \sum_{i=1}^{n}(\theta_i - \theta_{(i)})^2$. Consider g as a function of θ_i; θ_i appears twice in g, one as $(\theta_i - \theta_{(i)})^2$, and the other as $(\theta_{\pi(i)} - \theta_{(\pi(i))})^2 = (\theta_{\pi(i)} - \theta_i)^2$. If $\pi(i) = i$, then θ_i does not influence value of g; otherwise, g is a quadratic function of θ_i, and it achieves maximum either when $\theta_i = M$ or $\theta_i = -M$. So when g achieves maximum it must be $\theta_i \in \{-M, M\}$. Now notice that $\theta_{(1)} \leq \cdots \leq \theta_{(n)}$, so the maximum is achieved when for some $0 \leq k \leq n$ that $\theta_{(i)} = -M$ for $i \leq k$, and $\theta_{(i)} = M$ for $i > k$.

Note that $\sum_{i=1}^{n}(\theta_i - \theta_{(i)})^2 = \sum_{i=1}^{n}(\theta_{\pi^{-1}(i)} - \theta_{(\pi^{-1}(i))})^2 = \sum_{i=1}^{n}(\theta_{(i)} - \theta_{(\pi^{-1}(i))})^2$. From the discussion above, in the maximum case $(\theta_{(i)} - \theta_{(\pi^{-1}(i))})^2 = 4M^2$ iff i and $\pi^{-1}(i)$ lies on different sides of k, and otherwise it is 0. To further bound the sum, we use the Spearman Footrule distance between π and $(1, 2, \ldots, n)$, which [66] shows that it can be bounded as

$$\sum_{i=1} |\pi(i) - i| \leq 2 \sum_{1 \leq i,j \leq n} I\left[(\pi(i) - \pi(j))(i - j) < 0\right].$$

And the RHS can be bounded by $2\nu n^2$ since the agnostic error of ranking is at most ν. We also have that

$$\sum_{i=1} |\pi(i) - i| = \sum_{i=1} |\pi(\pi^{-1}(i)) - \pi^{-1}(i)| = \sum_{i=1} |i - \pi^{-1}(i)|.$$

Let $U_1 = \{i : \pi^{-1}(i) \leq k, i > k\}$ and $U_2 = \{\pi^{-1}(i) > k, i \leq k\}$. So in the maximum case we have

$$\sum_{i=1}^{n} (\theta_{\pi(i)} - \theta_i)^2 = 4M^2(|U_1| + |U_2|).$$

Now notice that for $i \in U_1$, we have $|\pi^{-1}(i) - i| \geq i - k$; and for $i \in U_2$ we have$|\pi^{-1}(i) - i| \geq k - i + 1$. Considering the range of i we have

$$\sum_{j=1}^{|U_1|} j + \sum_{j=1}^{|U_2|} j \leq \sum_{i=1}^{n} |\pi^{-1}(i) - i| \leq 2\nu n^2.$$

So $|U_1| + |U_2| \leq 2\sqrt{2\nu}n$. And

$$\sum_{i=1}^{n} (f(X_i) - f(X_{(i)})^2 = \sum_{i=1}^{n} (\theta_{\pi(i)} - \theta_i)^2 \leq 4M^2(|U_1| + |U_2|) \leq 8M^2\sqrt{2\nu}n.$$

Thus we prove the lemma. □

Now back to the original proof. Under event $\mathcal{E}_0$ we have

$$\begin{aligned}
&\sum_{i=1}^{n} E[(\widehat{y}_i - f(X_i))^2|\mathcal{E}_0] \\
=&\sum_{i=1}^{n} E[(\widehat{y}_{\widetilde{i}} - f(X_i))^2|\mathcal{E}_0] \\
\leq&\sum_{i=1}^{n} \mathbb{E}\left[2(\widehat{y}_{\widetilde{i}} - f(X_{\widetilde{i}}))^2 + 2(f(X_{\widetilde{i}}) - f(X_i))^2|\mathcal{E}_0\right] \\
\leq&\frac{Cn\log(m/\delta)}{m}\sum_{k=1}^{m} \mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2|\mathcal{E}_0\right] + 2\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{\widetilde{i}}) - f(X_i))^2|\mathcal{E}_0\right]. \qquad (3.15)
\end{aligned}$$

We omit the condition $\mathcal{E}_0$ in discussion below. We bound the two terms separately. For the first term, we use the following theorem adapted from [203]:

Theorem 38 ([203], adapted from Theorem 2.3 and Remark 4.2)**.** *Suppose X_{t_k}, y_{t_k} are fixed for $k \in [m]$, and $f(X_{t_k})$ is arbitrary in order. Let*

$$S = \min_{u} \sum_{k=1}^{m} (u_k - f(X_{t_k}))^2,$$

where the minimum is taken over all sequence of $u \in \mathbb{R}^m$ that is non-decreasing. The risk of isotonic regression satisfies

$$\frac{1}{m^{2/3}M^{1/3}}\left(\mathbb{E}\left[\sum_{k=1}^{m}(\widehat{y}_{t_k} - f(X_{t_k}))^2\right] - S\right) \leq C$$

for some universal constant C.

So from Theorem 38 we know that

$$\mathbb{E}\left[(\widehat{y}_{t_k} - f(X_{t_k}))^2\right] \leq \mathbb{E}[S] + Cm^{1/3},$$

where the expectation in $E[S]$ is taken w.r.t. the randomness in t_k. From Lemma 25 we know that

$$\sum_{i=1}^{n}(f(X_i) - f(X_{(i)}))^2 \leq C\sqrt{\nu}n.$$

Note that since t_k is taken at random, each element X_i has equal probability $\frac{m}{n}$ to be picked; so

$$\mathbb{E}[S] \leq \mathbb{E}\left[\sum_{i=1}^{m}(f(X_{(t_k)}) - f(X_{t_k}))^2\right] \leq C\sqrt{\nu}m.$$

Now we bound the second term in (3.15). We have

$$
\begin{aligned}
&\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{\widetilde{i}}) - f(X_i))^2\right] \\
&\leq 3\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{\widetilde{i}}) - f(X_{\widetilde{(i)}}))^2\right] + 3\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{\widetilde{(i)}}) - f(X_{(i)}))^2\right] + 3\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{(i)}) - f(X_i))^2\right] \\
&\leq \frac{Cn\log m}{m}\sum_{k=1}^{m} \mathbb{E}\left[(f(X_{t_k}) - f(X_{(t_k)}))^2\right] + \frac{Cn\log m}{m}\sum_{k=1}^{m} \mathbb{E}\left[(f(X_{(t_{k+1})}) - f(X_{(t_k)}))^2\right] \\
&\qquad + 3\sum_{i=1}^{n} \mathbb{E}\left[(f(X_{(i)}) - f(X_i))^2\right] \\
&\leq \frac{Cn\log m}{m}\sqrt{\nu}m + \frac{Cn\log m}{m}\cdot 1 + C\sqrt{\nu}n \\
&= C\sqrt{\nu}n\log m.
\end{aligned}
$$

The first inequality is by noticing $(x+y+z)^2 \leq 3x^2+3y^2+3z^2$ for any number $x, y, z \in \mathbb{R}$; the second inequality is by grouping values of $\widetilde{i}$, and the choice of t_k; the third inequality comes from analysis of the first term on $\sum_{k=1}^{m} \mathbb{E}\left[(f(X_{t_k}) - f(X_{(t_k)}))^2\right]$, the fact that $f(X_{(t_m)}) - f(X_{(t_1)}) \leq 1$, and Lemma 25.

Summarizing the two terms we have

$$\mathbb{E}\left[(\widehat{y}_{\widetilde{i}} - f(X_i))^2 | \mathcal{E}_0\right] \leq C(\sqrt{\nu}n + m^{-2/3}n)\log m.$$

Take this back to (3.14) we prove the theorem.

3.8.4 Proof of Theorem 26

To simplify notation, we suppose that we have m labeled samples $T = \{(X_i, y_i)\}_{i=1}^{m}$ for training and another m labeled samples $V = \{(X_i, y_i)\}_{i=m+1}^{2m}$ for validation. We consider the following models:

1. R^2 using both the ordinal data and the labeled samples in T, and we denote by $\widehat{f}_0$,
2. k-NN regression using only the labeled samples in T for $k \in [m]$ which we denote by $\{\widehat{f}_1, \ldots, \widehat{f}_m\}$.

We select the best model according to performance on validation set. We further restrict all estimators to be bounded in $[-M, M]$; i.e., when $\widehat{f}_j(x) < -M$ for some x and j, we clip its value by setting $\widehat{f}_j(x) = -M$ and we analogously clip the function when it exceeds M. We note in passing that this only reduces the MSE since the true function f is bounded between $[-M, M]$. Throughout the remainder of the proof we condition on the training set T but suppress this in our notation. We define the empirical validation risk of a function $\widehat{f}$ to be:

$$\widehat{R}^V\left(\widehat{f}\right) = \frac{1}{m}\sum_{i=m+1}^{2m}\left(y_i - \widehat{f}(X_i)\right)^2,$$

and the population MSE of a function $\widehat{f}$ to be

$$\mathsf{err}(\widehat{f}) = \mathbb{E}\big[\big(\widehat{f}(X) - f(X)\big)^2\big],$$

where $\epsilon \sim N(0,1)$ denotes the noise in the direct measurements. Now let

$$\widehat{f}^* = \operatorname*{arg\,min}_{j=0,\ldots,m} \widehat{R}^V(\widehat{f}_j),$$

be the best model selected using cross validation and

$$f^* = \operatorname*{arg\,min}_{j=0,\ldots,m} \mathsf{err}(\widehat{f}_j),$$

be the estimate with lowest MSE in $\widehat{f}_0, \ldots, \widehat{f}_m$. Let us denote:

$$\mathcal{G} = \{\widehat{f}_0, \ldots, \widehat{f}_m\}.$$

Recall that f denotes the true unknown regression function. Then in the sequel we show the following result:

Lemma 39. *With probability at least $1-\delta$, for any $\widehat{f} \in \mathcal{G}$ we have that the following hold for some constant $C > 0$,*

$$\mathsf{err}(\widehat{f}) \leq 2\left[\widehat{R}^V(\widehat{f}) - \widehat{R}^V(f)\right] + \frac{C\log(m/\delta)}{m}, \tag{3.16}$$

$$\left[\widehat{R}^V(\widehat{f}) - \widehat{R}^V(f)\right] \leq 2\mathsf{err}(\widehat{f}) + \frac{C\log(m/\delta)}{m}. \tag{3.17}$$

Since $\widehat{R}^V(\widehat{f}^*) \leq \widehat{R}^V(f^*)$ we obtain using Equation (3.16) that with probability at least $1-\delta$,

$$\mathsf{err}(\widehat{f}^*) \leq 2\left[\widehat{R}^V(f^*) - \widehat{R}^V(f)\right] + \frac{C\log(m/\delta)}{m}.$$

Since $f^* \in \mathcal{G}$, we can use Equation (3.17) to obtain that with probability $1-\delta$,

$$\mathsf{err}(\widehat{f}^*) \leq 4\mathsf{err}(f^*) + \frac{2C\log(m/\delta)}{m}.$$

Since $\mathsf{err}(\widehat{f}^*)$ is a positive random variable, integrating this bound we obtain that,

$$\begin{aligned}\mathbb{E}[\mathsf{err}(\widehat{f}^*)] &= \int_0^\infty \mathbb{P}(\mathsf{err}(\widehat{f}^*) \geq t)dt, \\ &\leq 4\mathsf{err}(f^*) + \frac{6C\log(m)}{m}.\end{aligned}$$

So far we have implicitly conditioned throughout on the training set T. Taking an expectation over the training set yields:

$$\mathbb{E}[\mathsf{err}(\widehat{f}^*)] \leq 4\mathbb{E}[\mathsf{err}(f^*)] + \frac{6C\log(m)}{m}.$$

We now note that,

$$\mathbb{E}[\mathsf{err}(f^*)]] \leq \min_{j\in\{0,\ldots,m\}} \mathbb{E}[\mathsf{err}(\widehat{f}_j)].$$

Standard results on k-NN regression (for instance, a straightforward modification of Theorem 6.2 in [82] to deal with $0 < s \leq 1$ in the Hölder class) yield that for a constant $C > 0$,

$$\min_{j\in\{1,\ldots,m\}} \mathbb{E}[\mathsf{err}(\widehat{f}_j)] \leq Cm^{-2s/(2s+d)}.$$

Theorem 24 yields that,

$$\mathbb{E}[\mathsf{err}(\widehat{f}_0)] \leq C_1 \left(\log^2 n \log m \left(m^{-2/3} + \sqrt{\nu}\right)\right) + C_2 n^{-2s/d},$$

and putting these together we obtain that,

$$\mathbb{E}[\mathsf{err}(f^*)] \leq \widetilde{O}\left(m^{-2/3} + \min\{\sqrt{\nu}, m^{-\frac{2s}{2s+d}}\} + n^{-2s/d}\right),$$

and thus it only remains to prove Lemma 39 to complete the proof of the theorem.

Proof of Lemma 39

For a fixed classifier $\widehat{f}$ and for samples in the validation set $i \in \{m+1, \ldots, 2m\}$ we define the random variables:

$$Z_i = (y_i - \widehat{f}(X_i))^2 - (y_i - f(X_i))^2 = (\widehat{f}(X_i) - f(X_i))^2 + 2\epsilon_i(f(X_i) - \widehat{f}(X_i)),$$

and note that $\mathbb{E}[Z_i] = \mathsf{err}(\widehat{f})$. In order to obtain tail bounds on the average of the Z_i let us bound the absolute central moments of Z_i. Using the inequality that $(x+y)^k \leq 2^{k-1}(x^k + y^k)$, for $k > 2$ we obtain that,

$$\begin{aligned}\mathbb{E}|Z_i - \mathbb{E}[Z_i]|^k &= \mathbb{E}|(\widehat{f}(X_i) - f(X_i))^2 + 2\epsilon_i(f(X_i) - \widehat{f}(X_i)) - \mathsf{err}(\widehat{f})|^k \\ &\leq 2^{k-1}\mathbb{E}|(\widehat{f}(X_i) - f(X_i))^2 - \mathsf{err}(\widehat{f})|^k + 2^{k-1}\mathbb{E}|\epsilon_i(f(X_i) - \widehat{f}(X_i))|^k. \quad (3.18)\end{aligned}$$

We bound each of these terms in turn. Since $(\widehat{f}(X_i) - f(X_i))^2 \in [0, 4M^2]$, we obtain that,

$$\mathbb{E}|(\widehat{f}(X_i) - f(X_i))^2 - \mathsf{err}(\widehat{f})|^k \leq \mathrm{var}((\widehat{f}(X_i) - f(X_i))^2)(4M^2)^{k-2},$$

and using the fact that ϵ_i are Gaussian we obtain that,

$$\begin{aligned}\mathbb{E}|\epsilon_i(f(X_i) - \widehat{f}(X_i))|^k &\leq \mathbb{E}|\epsilon_i|^{k-2}\mathbb{E}(f(X_i) - \widehat{f}(X_i))^2(2M)^{k-2} \\ &\leq \mathrm{var}(\epsilon_i(f(X_i) - \widehat{f}(X_i)))k! \times (2M)^{k-2}.\end{aligned}$$

Since ϵ_i is independent of the other terms in Z_i we have that,

$$\mathrm{var}(Z_i) = \mathrm{var}(\epsilon_i(f(X_i) - \widehat{f}(X_i))) + \mathrm{var}((\widehat{f}(X_i) - f(X_i))^2).$$

Putting these pieces together with Equation (3.18) we obtain,

$$\begin{aligned}\mathbb{E}|Z_i - \mathbb{E}[Z_i]|^k &\leq 2^{k-1}\text{var}(Z_i)\left[k! \times (2M)^{k-2} + (4M^2)^{k-2}\right] \\ &\leq \frac{\text{var}(Z_i)}{2} k!(16M + 32M^2)^{k-2}.\end{aligned}$$

Let us denote $r := 16M + 32M^2$. It remains to bound the variance. We have that,

$$\text{var}(Z_i) \leq \mathbb{E}[Z_i^2] \leq 2\mathbb{E}((\widehat{f}(X_i) - f(X_i))^4) + 8\mathbb{E}(f(X_i) - \widehat{f}(X_i))^2,$$

and using the fact that the functions are bounded in $[-M, M]$ we obtain that,

$$\text{var}(Z_i) \leq (8M^2 + 8)\mathsf{err}(\widehat{f}). \tag{3.19}$$

Now, applying the inequality in Lemma 44, we obtain that for any $c < 1$ and for any $t \leq c/r$ that,

$$\mathsf{err}(\widehat{f}) \leq \frac{1}{m}\sum_{i=m+1}^{2m} Z_i + \frac{\log(1/\delta)}{mt} + \frac{8t(M^2+1)\mathsf{err}(\widehat{f})}{2(1-c)},$$

we choose $c = 1/2$ and $t = \min\{1/(2r), 1/(16(M^2+1))\}$, and rearrange to obtain that,

$$\mathsf{err}(\widehat{f}) \leq \frac{2}{m}\sum_{i=m+1}^{2m} Z_i + \frac{2\log(1/\delta)}{m}\max\{2r, 16(M^2+1)\}, \leq \frac{2}{m}\sum_{i=m+1}^{2m} Z_i + \frac{C\log(1/\delta)}{m},$$

and using a union bound over the $m+1$ functions $\widehat{f} \in \mathcal{G}$ we obtain Equation (3.16). Repeating this argument with the random variables $-Z_i$ we obtain Equation (3.17) completing the proof of the Lemma.

3.8.5 Proof of Theorem 27

We prove a slightly stronger result, and show Theorem 27 as a corollary.

Theorem 40. *Assume the same modeling assumptions for $X_1, \ldots, X_n, y_1, \ldots, y_m$ as in Theorem 23. Also permutation $\widehat{\pi}$ satisfies $\mathbb{P}[(f(X_i) - f(X_j))(\pi(i) - \pi(j)) < 0] \leq \nu$. Then for any estimator $\widehat{f}$ taking input $X_1, \ldots, X_n, y_1, \ldots, y_m$ and $\widehat{\pi}$, we have*

$$\inf_{\widehat{f}} \sup_{f \in \mathcal{F}_{s,L}} \mathbb{E}\big(f(X) - \widehat{f}(X)\big)^2 \geq C(m^{-\frac{2}{3}} + \min\{\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}}, 1\} m^{-\frac{2}{d+2}} + n^{-2s/d}).$$

Proof of Theorem 40 In this proof, we use x_i to represent i-th dimension of x, and upper script for different vectors $x^{(1)}, x^{(2)}, \ldots$. Let $u = \lceil m^{\frac{1}{2+d}} \rceil, h = 1/u$, and $t = \min\left\{\left(\nu m^{\frac{1}{2+d}}\right)^{\frac{1}{2d}}, 1\right\}$. Let $\Gamma = \{(\gamma_1, \ldots, \gamma_d), \gamma_i \in \{1, 2, \ldots, u\}\}$. Choose an arbitrary order on Γ to be $\Gamma = \{\gamma^{(1)}, \ldots, \gamma^{(u^d)}\}$. Let $x^{(k)} = \frac{\gamma^{(k)} - 1/2}{u}$, and $\phi_k(x) = \frac{L}{2} thK(\frac{x - tx^{(k)}}{th}), k = 1, 2, \ldots, u^d$, where K is a kernel function in d dimension supported on $[-1/2, 1/2]^d$, i.e., $\int K(x)dx$ and $\max_x K(x)$ are both

bounded, K is 1-Lipschitz. So $\phi_k(x)$ is supported on $[thx^{(k)} - 1/2th, thx^{(k)} + 1/2th]$. Let $\Omega = \{\omega = (\omega_1, \ldots, \omega_{u^d}), \omega_i \in \{0, 1\}\}$, and

$$\mathcal{E} = \left\{ f_\omega(x) = \frac{L}{2} x_1 + \sum_{i=1}^{k} \omega_i \phi_k(x), x \in [0,1]^d \right\}.$$

Functions in $\mathcal{E}$ are L-Lipschitz. The function value is linear in x_1 for $x \notin [0,t]^d$ in all functions in $\mathcal{E}$. Consider the comparison function $Z(x, x') = I(x_1 < x_1')$ that ranks x according to the first dimension. Since K is 1-Lipschitz, it only makes an error when both x, x' lies in $[0,t]^d$, and both x_1, x_1' lie in the same grid segment $[tk/u, t(k+1)/u]$ for some $k \in [u]$. So the error is at most $t^{2d}(1/u)^2 \cdot u \leq \nu$ for any function $f \in \mathcal{E}$. Thus, if there exists one estimator with $\sup_f \mathbb{E}[(f - \widehat{f})^2] < C \min\{\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}}, 1\} m^{-\frac{2}{d+2}}$, then we can obtain one estimator for functions in $\mathcal{E}$ by using $\widehat{f}$ on $\mathcal{E}$, and responding to all comparisons and rankings as $Z(x, x') = I(x_1 < x_1')$. So a lower bound on learning $\mathcal{E}$ is also a lower bound on learning any $f \in \mathcal{F}_{s,L}$ with ν-agnostic comparisons.

Now we show that $Ct^{d+2}h^2 = C \min\{\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}}, 1\} m^{-\frac{2}{d+2}}$ is a lower bound to approximate functions in $\mathcal{E}$. For all $\omega, \omega' \in \Omega$ we have

$$\begin{aligned}
\mathbb{E}[(f_\omega - f_{\omega'})^2]^{1/2} &= \left(\sum_{k=1}^{p^d} (\omega_k - \omega_k')^2 \int \phi_k^2(x) dx \right)^{1/2} \\
&= \left(\rho(\omega, \omega') L^2 t^{d+2} h^{d+2} \right)^{1/2} \\
&= L(th)^{\frac{d+2}{2}} \|K\|_2 \sqrt{\rho(\omega, \omega')},
\end{aligned}$$

where $\rho(\omega, \omega')$ denotes the Hamming distance between x and x'.

By the Varshamov-Gilbert lemma, we can have a $M = O(2^{u^d/8})$ subset $\Omega' = \{\omega^{(0)}, \omega^{(1)}, \ldots, \omega^{(M)}\}$ of Ω such that the distance between each element $\omega^{(i)}, \omega^{(j)}$ is at least $u^d/8$. So $d(\theta_i, \theta_j) \geq h^s t^{(d+2)/2}$. Now for P_j, P_0 (P_0 corresponds to f_ω when $\omega = (0, 0, \ldots, 0)$) we have

$$\begin{aligned}
KL(P_j, P_0) &= m \int_{\mathcal{X}} p(x) \int_{\dagger} p_j(y|x) \log \frac{p_0(y|x)}{p_j(y|x)} dy dx \\
&= m \int_{\mathcal{X}} p(x) \sum_{i=1}^{u^d} \omega_i^{(j)} \phi_{\omega^{(j)}}^2(x) \\
&\leq m \cdot C h^{d+2} t^{d+2} u^d = C u^d t^{d+2}.
\end{aligned}$$

We have $Cu^d t^{d+2} \leq cu^d \leq \alpha \log M$ (since $t \leq 1$), so again using Theorem 2.5 in [166] we obtain a lower bound of $d(\theta_i, \theta_j)^2 = Ch^2 t^{d+2} = \min\{\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}}, 1\} m^{-\frac{2}{d+2}}$. ■

Now we can prove Theorem 27.

Proof of Theorem 27. We only need to show

$$\min\{\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}}, 1\} m^{-\frac{2}{d+2}} \geq \min\{\nu^2, m^{-\frac{2}{d+2}}\}. \tag{3.20}$$

If $\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}} \geq 1$, we have $\nu^2 \geq m^{-\frac{2}{d+2}}$. In this case both sides of (3.20) equals $m^{-\frac{2}{d+2}}$. If $\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}} \leq 1$, we have $m \leq \nu^{-(d+2)}$, and thus LHS of (3.20) have term $\nu^{\frac{d+2}{2d}} m^{\frac{1}{2d}} m^{-\frac{2}{d+2}} \geq \nu^2$, which equals RHS. □

3.8.6 Proof of Theorem 30

Proof. We first list properties of log-concave distributions:

Theorem 41 ([13, 121])**.** *The following statements hold for an isotropic log-concave distribution $\mathbb{P}_X$:*

1. *Projections of $\mathbb{P}_X$ onto subspaces of $\mathbb{R}^d$ are isotropic log-concave.*
2. *$\mathbb{P}[\|X\|_2 \geq \alpha\sqrt{d}] \leq e^{1-\alpha}$.*
3. *There is an absolute constant C such that for any two unit vectors u and v in $\mathbb{R}^d$ we have $C\|v-u\|_2 \leq \mathbb{P}(\textit{sign}(u \cdot X) \neq \textit{sign}(v \cdot X))$.*

From property of $\mathcal{A}^c$ and point 3 in Theorem 41 we can get $\|\widehat{v} - v^*\|_2 \leq C\varepsilon_{\mathcal{A}^c}(n, \delta/4)$ using n comparisons, with probability $1 - \delta/4$. We use a shorthand notion $\varepsilon = C\varepsilon_{\mathcal{A}^c}(n, \delta/4)$ for this error. Now consider estimating r^*. The following discussion is conditioned on a fixed $\widehat{v}$ that satisfies $\|\widehat{v} - v^*\|_2 \leq \varepsilon$. For simplicity let $T_i = \langle \widehat{v}, X \rangle_i$. We have

$$\begin{aligned}
\widehat{r} &= \frac{\sum_{i=1}^m T_i y_i}{\sum_{i=1}^m T_i^2} \\
&= \frac{\sum_{i=1}^m T_i r^* \langle v^*, X_i \rangle + T_i \varepsilon_i}{\sum_{i=1}^m T_i^2} \\
&= r^* + \frac{\sum_{i=1}^m T_i r^* \langle v^* - \widehat{v}, X_i \rangle + T_i \varepsilon_i}{\sum_{i=1}^m T_i^2}.
\end{aligned}$$

Now we have

$$\begin{aligned}
\langle w^* - \widehat{w}, X \rangle &= r^* \langle v^*, X \rangle - \widehat{r} \langle \widehat{v}, X \rangle \\
&= r^* \langle v^* - \widehat{v}, X \rangle - \frac{\sum_{i=1}^m T_i r^* \langle v^* - \widehat{v}, X_i \rangle + T_i \varepsilon_i}{\sum_{i=1}^m T_i^2} \langle \widehat{v}, X \rangle.
\end{aligned}$$

So

$$\begin{aligned}
&\mathbb{E}\left[\langle w^* - \widehat{w}, X \rangle^2\right] \\
&\leq 3\mathbb{E}\left[(r^* \langle v^* - \widehat{v}, X \rangle)^2\right] + 3\mathbb{E}\left[\left(\frac{\sum_{i=1}^m T_i r^* \langle v^* - \widehat{v}, X_i \rangle}{\sum_{i=1}^m T_i^2} \langle \widehat{v}, X \rangle\right)^2\right] + 3\left[\left(\frac{\sum_{i=1}^m T_i \varepsilon_i}{\sum_{i=1}^m T_i^2} \langle \widehat{v}, X \rangle\right)^2\right]
\end{aligned} \tag{3.21}$$

The first term can be bounded by

$$(r^*)^2 \mathbb{E}[\langle \widehat{v} - v^*, X \rangle^2] = (r^*)^2 \|\widehat{v} - v^*\|_2^2 \leq (r^*)^2 \varepsilon^2.$$

For the latter two terms, we first bound the denominator $\sum_{i=1}^m T_i^2$ using Hoeffding's inequality. Firstly since $\|\widehat{v}\|_2 = 1$, from point 1 in Theorem 41, each T_i is also isotropic log-concave. Now using point 2 in Theorem 41 with $\alpha = 1 - \log(\delta/(4em))$ we get that with probability $1 - \delta/4$, $T_i \le \log(4em/\delta)$ for all $i \in \{1, 2, \ldots, m\}$. Let E_δ^T denote this event, and $\mathbb{P}'_X$ is the distribution of X such that $T_i \le \log(4em/\delta)$. Now using Hoeffding's inequality, under E_δ^T for any $t > 0$

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^m T_i^2 - \mathbb{E}_{\mathbb{P}'_X}[\langle \widehat{v},\, X\rangle^2]\right| \ge t\right] \le \exp\left(-\frac{2mt^2}{\log^2(4em/\delta)}\right).$$

Note that $\mathbb{E}_{\mathbb{P}'_X}[\langle \widehat{v},\, X\rangle^2] \le \mathbb{E}_{\mathbb{P}_X}[\langle \widehat{v},\, X\rangle^2] = 1$. Also we have

$$\begin{aligned}
1 = \mathbb{E}[T_i^2] &\le \mathbb{E}_{\mathbb{P}'_X}[T_i^2] + \int_{\log^2(4em/\delta)}^{+\infty} t\mathbb{P}[T_i^2 \ge t]dt \\
&\le \mathbb{E}_{\mathbb{P}'_X}[T_i^2] + \int_{\log^2(4em/\delta)}^{+\infty} te^{-\sqrt{t}+1}dt \\
&\le \mathbb{E}_{\mathbb{P}'_X}[T_i^2] + \frac{3\delta}{2m}
\end{aligned}$$

Let $t = \frac{1}{4}\mathbb{E}_{\mathbb{P}'_X}[\langle \widehat{v},\, X\rangle^2]$, we have

$$\sum_{i=1}^m T_i^2 \le \left[\frac{3m}{4}\mathbb{E}_{\mathbb{P}'_X}[\langle \widehat{v},\, X\rangle^2], \frac{5m}{4}\mathbb{E}_{\mathbb{P}'_X}[\langle \widehat{v},\, X\rangle^2]\right] \subseteq [m/2, 2m]. \tag{3.22}$$

with probability $1 - \delta/4$, when $m = \Omega(\log^3(1/\delta))$.

Let E_δ denote the event when $m/2 \le \sum_{i=1}^m T_i^2 \le 2m$ and T_i are bounded by $\log(4em/\delta)$ for all i . Condition on E_δ for the second term in (3.21) we have

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{\sum_{i=1}^m T_i r^*\langle v^* - \widehat{v},\, X_i\rangle}{\sum_{i=1}^m T_i^2}\langle \widehat{v},\, X\rangle\right)^2\right] &\le \frac{\mathbb{E}\left[\left(\sum_{i=1}^m T_i r^*\langle v^* - \widehat{v},\, X_i\rangle\right)^2\right]}{\frac{m^2}{4}}\mathbb{E}[(\langle \widehat{v},\, X\rangle)^2] \\
&= \frac{4\mathbb{E}\left[\left(\sum_{i=1}^m T_i r^*\langle v^* - \widehat{v},\, X_i\rangle\right)^2\right]}{m^2}
\end{aligned}$$

Now notice that $\frac{\widehat{v}-v^*}{\|\widehat{v}-v^*\|_2}X$ is also isotropic log-concave; using point 2 in Theorem 41 we have with probability $1 - \delta/4$, $(\widehat{v} - v^*)^T X_i \le \|\widehat{v} - v^*\|_2 \log(4em/\delta)$ for all $i \in \{1, 2, \ldots, m\}$. So

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i=1}^m T_i r^*\langle v^* - \widehat{v},\, X_i\rangle\right)^2\right] &\le (r^*)^2\varepsilon^2\log^2(4em/\delta)\mathbb{E}\left[\left(\sum_{i=1}^m |T_i|\right)^2\right] \\
&\le (r^*)^2\varepsilon^2\log^2(4em/\delta)\mathbb{E}\left[m\sum_{i=1}^m T_i^2\right] \\
&= (r^*)^2\varepsilon^2\log^2(4em/\delta)m^2
\end{aligned}$$

For the third term in (3.21), also conditioning on E_δ we have

$$\begin{aligned}\mathbb{E}\left[\left(\frac{\sum_{i=1}^m T_i\varepsilon_i}{\sum_{i=1}^m T_i^2}\langle\widehat{v},\, X\rangle\right)^2\right] &= \mathbb{E}\left[\left(\frac{\sum_{i=1}^m T_i\varepsilon_i}{\sum_{i=1}^m T_i^2}\right)^2\right]\mathbb{E}\left[\langle\widehat{v},\, X\rangle^2\right] \\ &\leq \frac{\mathbb{E}\left[\left(\sum_{i=1}^m T_i\varepsilon_i\right)^2\right]}{\frac{m^2}{4}} \\ &\leq \frac{4\mathbb{E}\left[\sum_{i=1}^m T_i^2\sigma^2\right]}{m^2} = \frac{4\sigma^2}{m}.\end{aligned}$$

Combining the three terms and considering all the conditioned events, we have

$$\begin{aligned}\mathbb{E}\left[(\langle w^*,\, X\rangle - \langle\widehat{w},\, X\rangle)^2\right] \leq& 4(r^*)^2\varepsilon^2 + (r^*)^2\varepsilon^2\log^2(4em/\delta) + \frac{4\sigma^2}{m} + C'\delta \\ \leq& O\left(\frac{1}{m} + \log^2(m/\delta)\varepsilon_{\mathcal{A}^c}(n,\delta/4) + \nu^2 + \delta\right)\end{aligned}$$

Taking $\delta = \frac{4}{m}$ obtain our desired result.

□

3.8.7 Proof of Theorem 32

Our proof ideas come from [44]. We use Le Cam's method, explained in the lemma below:

Lemma 42 (Theorem 2.2, [166])**.** *Suppose $\mathcal{P}$ is a set of distributions parametrized by $\theta \in \Theta$. $P_0, P_1 \in \mathcal{P}$ are two distributions, parametrized by θ_0, θ_1 respectively, and $KL(P_1, P_2) \leq \alpha \leq \infty$. Let d be a semi-distance on Θ, and $d(\theta_0, \theta_1) = 2a$. Then for any estimator $\widehat{\theta}$ we have*

$$\begin{aligned}\inf_{\widehat{\theta}}\sup_{\theta\in\Theta}\mathbb{P}[d(\widehat{\theta},\theta) \geq a] \geq& \inf_{\widehat{\theta}}\sup_{j\in\{0,1\}}\mathbb{P}[d(\widehat{\theta},\theta_j) \geq a] \\ \geq& \max\left(\frac{1}{4}\exp(-\alpha), -\frac{1-\sqrt{\alpha/2}}{2}\right)\end{aligned}$$

We consider two functions: $w_0^* = (\xi, 0, 0, \ldots, 0)^T$ and $w_1^* = (\frac{1}{\sqrt{m}}, 0, 0, \ldots, 0)^T$, where ξ is a very small constant. Note that for these two functions comparisons provide no information about the weights (comparisons can be carried out directly by comparing $x^{(1)}$, the first dimension of x). So differentiating w_0^* and w_1^* using two oracles is the same as that using only active labels. We have $d(w_0^*, w_1^*) = E\left[\left((w_0^* - w_1^*)^T X\right)^2\right] = \left(\frac{1}{\sqrt{m}} - \xi\right)^2$. For any estimator $\widehat{w}$, let $\{(X_i, y_i)\}_{i=1}^m$ be the set of samples and labels obtained by $\widehat{w}$. Note that X_{j+1} might depend on $\{(X_i, y_i)\}_{i=1}^j$.

Now for KL-divergence we have

$$\begin{aligned}
\mathrm{KL}(P_1, P_0) &= \mathbb{E}_{P_1}\left[\log \frac{P_1\left(\{(X_i, y_i)\}_{i=1}^m\right)}{P_0\left(\{(X_i, y_i)\}_{i=1}^m\right)}\right] \\
&= \mathbb{E}_{P_1}\left[\log \frac{\prod_{j=1}^m P_1(Y_j|X_j) P(X_j|\{(X_i, y_i)\}_{i=1}^j)}{\prod_{j=1}^m P_1(Y_j|X_j) P(X_j|\{(X_i, y_i)\}_{i=1}^j)}\right] \\
&= \mathbb{E}_{P_1}\left[\log \frac{\prod_{j=1}^m P_1(y_j|X_j)}{\prod_{j=1}^m P_0(y_j|X_j)}\right] \\
&= \sum_{i=1}^m \mathbb{E}_{P_1}\left[\mathbb{E}_{P_1}\left[\log \frac{\prod_{j=1}^m P_1(y_j|X_j)}{\prod_{j=1}^m P_0(y_j|X_j)}\middle| X_1, \ldots, X_m\right]\right] \\
&\leq n \max_x \mathbb{E}_{P_1}\left[\log \frac{\prod_{j=1}^m P_1(y_j|X_j)}{\prod_{j=1}^m P_0(y_j|X_j)}\middle| X_1 = x\right].
\end{aligned}$$

The third equality is because decision of X_j is independent of the underlying function giving previous samples. Note that given $X = x$, y is normally distributed; by basic properties of Gaussian we have

$$\mathbb{E}_{P_1}\left[\log \frac{\prod_{j=1}^m P_1(y_j|X_j)}{\prod_{j=1}^m P_0(y_j|X_j)}\middle| X_1 = x\right] = \frac{(\frac{1}{\sqrt{m}-\xi})^2}{2\sigma^2}.$$

Now by taking ξ sufficiently small we have for some constants C_1, C_2,

$$\mathrm{KL}(P_1, P_0) \leq C_1, d(\theta_0, \theta_1) \geq \frac{C_2}{m}.$$

Combining with Lemma 42 we obtain the lower bound.

3.8.8 Lower Bounds for Total Number of Queries under Active Case

Theorem 43. *For any (active) estimator $\widehat{w}$ with access to m labels and n comparisons, there exists a ground truth weight $\widetilde{w}$ and a global constant C, such that when $w^* = \widetilde{w}$ and $2n + m < d$,*

$$\mathbb{E}\left[\langle \widehat{w} - w^*, X\rangle^2\right] \geq C.$$

Theorem 43 shows a lower bound on the total number of queries in order to get low error. Combining with Theorem 32, in order to get a MSE of γ for some $\gamma < C$, we need to make at least $O(1/\gamma + d)$ queries (i.e., labels+comparisons). Note that for the upper bound in Theorem 30, we need $m + n = \widehat{O}(1/\gamma + d\log(d/\gamma))$ for Algorithm 6 to reach γ MSE, when using [13] as $\mathcal{A}^c$ (see Table 3.1). So Algorithm 6 is optimal in terms of total queries, up to log factors.

The proof of Theorem 43 is done by considering an estimator with access to $m + 2n$ *noiseless* labels $\{(x_i, w^* \cdot x_i)\}_{i=1}^{m+2n}$, which can be used to generate n comparisons and m labels. We sample w^* from a prior distribution in $B(0, 1)$, and show that the expectation of MSE in this case is at least a constant. Thus there exists a weight vector $\widetilde{w}$ that leads to constant error.

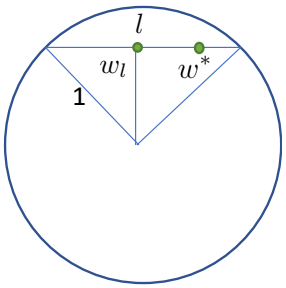


Figure 3.11: Graphic illustration about the sampling process in the proof of Theorem 43.

Proof. We just prove the theorem for $2n + m = d - 1$. Note that this case can be simulated by considering an estimator with access to $2n + m$ *truthful* samples; that is, $Y_i = w^* \cdot X_i$ for $i = 1, 2, \ldots, 2n + m$. In this way truthful comparisons can be simulated by query two labels. We now prove a lower bound for this new case with $2n + m = d - 1$ truthful samples.

We randomly sample w^* as below: first uniformly sample v^* on the surface of $B(0, 1)$, and then uniformly sample $r^* \in [0, 1]$. Let this distribution be $\mathcal{P}_{w^*}$. Since we only have $d - 1$ labels, for any set of samples $x_1, \ldots, x_{d-1}$ there exists a line $l \in B(0, 1)$ such that every $w \in l$ produces the same labels on all the samples. Not losing generality, suppose such l is unique (if not, we can augment $\widehat{w}$ such that it always queries until l is unique). Now for any active estimator $\widehat{w}$, let $X_1, \ldots, X_{d-1}$ denote the sequence of queried points when w^* is randomly picked as above. Now note that for every w,

$$\mathbb{P}[w^* = w | \{X_i, y_i\}_{i=1}^{d-1}, l] = \mathbb{P}[w^* = w | \{X_i, y_i\}_{i=1}^{d-1}] \propto \mathbb{P}[w^* = w] I(w \in l).$$

The first equality is because l is a function of $\{X_i, y_i\}_{i=1}^{d-1}$; the second statement is because all $w \in l$ produces the same dataset on $X_1, \ldots, X_{d-1}$, and every $w \notin l$ is impossible given the dataset. Notice that r^* is uniform on $[0, 1]$; so with probability at least a half, the resulting l has distance less than $1/2$ to the origin (since l contains w^*, and $\|w^*\|$ is uniform on $[0, 1]$). Denote by w_l the middle point of l (see Figure 3.11). For any such line l, the error is minimized by predicting the middle point of l: Actually we have

$$\mathbb{E}\left[\langle w^* - \widehat{w},\, X\rangle^2 | l \text{ has distance less than 1/2}\right] \geq \int_{u=0}^{|l|/2} u^2 dP(\|w^* - \widehat{w}\|_2 \geq u | w^* \in l) \quad (3.23)$$

$$\geq \int_{u=0}^{|l|/2} u^2 dP(\|w^* - w_l\|_2 \geq u | w^* \in l) \quad (3.24)$$

Note that the distribution of $w^* \in l$ is equivalent to that we sample from the circle containing l and centered at origin, and then condition on $w^* \in l$ (see Figure 3.11). Notice that this sampling process is the same as when $d = 2$; and with some routine calculation we can show that (3.24) is a constant C. So overall we have

$$\mathbb{E}\left[\langle w^* - \widehat{w},\, X\rangle^2\right] \geq \frac{1}{2} C,$$

where the expectation is taken over randomness of w^* and randomness of $\widehat{w}$. Now since the expectation is a constant, there must exists some w such that

$$\mathbb{E}\left[\langle w^* - \widehat{w},\, X\rangle^2 \middle|\, w^* = w\right] \geq \frac{1}{2}C,$$

which proves the theorem. □

3.9 Auxiliary Technical Results

We use some well-known technical results in our proofs and collect them here to improve readability. We use the Craig-Bernstein inequality [60]:

Lemma 44. *Suppose we have $\{X_1, \ldots, X_n\}$ be independent random variables and suppose that for $k \geq 2$, for some $r > 0$,*

$$\mathbb{E}[|X_i - \mathbb{E}[X_i]|^k] \leq \frac{\mathit{var}(X_i)}{2} k! r^{k-2}.$$

Then with probability at least $1 - \delta$, for any $c < 1$ and for any $t \leq c/r$ we have that:

$$\frac{1}{n}\sum_{i=1}^{n} (\mathbb{E}[X_i] - X_i) \leq \frac{\log(1/\delta)}{nt} + \frac{t\ \mathit{var}(X_i)}{2(1-c)}.$$

Part II

Decision Making with Dueling Choices

Chapter 4

Discrete and Continuous Multi-Armed Bandits with Dueling Choices

4.1 Introduction

The Multi-Armed Bandits (MAB) problem [144] is a popular framework that studies the tradeoff between exploration and exploitation inherent in applications like online advertising and finance. In its basic form, we have a arm space $\mathcal{X}$, and in each time step $t = 1, 2, ...$ we pull one arm $x_t \in \mathcal{X}$. The goal is to identify the arm with the best reward in $\mathcal{X}$, while also minimizing the cumulative regret in a total of T steps. Dueling bandit [200] is an important variant of MAB, where we take *duels* between a pair of arms $(x_t, x'_t) \in \mathcal{X}^2$, and x_t wins with probability $\Pr[x_t \succ x'_t] = 1/2 + \varepsilon(x_t, x'_t)$.

Dueling bandit is a natural fit for various applications including information retrieval and crowdsourcing; in these cases duels (comparisons) between arms come at a much smaller cost than pulling individual arms. However, in many of these applications, it is not enough to reply only on comparisons: For example, in information retrieval, the user preferences might not be transitive due to user bahavior, and a unique Condorcet winner might not exist. This has motivated research on suitable notions of a winner, including the Copeland winner [206], the von Neumann winner [69] and tournament solutions [142]. We propose a complementary approach for dueling bandits, where we can *either* pull one arm or duel two arms in each round. We call our setting Multi-Armed Bandits with Dueling Choices (MAB-DC), we note that different than dueling bandits, here direct queries and comparisons are both available, and therefore duels are available as an additional choice.

In many applications, both comparisons (duels) and direct queries are available, and comparisons can be available at a cheaper price than direct queries. For example in preference elicitation, the user can give scores on recommended items, as well as (more easily) compare two items to choose the preferred one. Similarly, for hyperparameter search of information retrieval (IR) models, direct queries typically involve collecting relevance scores from paid workers, whereas comparisons can be obtained by interleaving the ranking of two different models, and observing user click on the retrieval results [139]. Such comparisons usually come with less cost in both time and money. As another example, in material synthesis, we aim to optimize the desired properties

of materials by controlling the input parameters (temperature, pressure, etc.) [71, 194]. While material synthesis is expensive, comparisons can be carried out by asking material scientists.

4.2 Our Contribution

We develop and evaluate new algorithms for the MAB-DC problem, for both discrete and continuous arm spaces. a new algorithm for non-convex optimization with dueling choices, which we refer to as Comparison-based GP-UCB (COMP-GP-UCB). Our theoretical and experimental results show the strengths of our algorithm.

- In the discrete case, we propose th Double Filtering (DF) algorithm that combines duels and pulls. The algorithm alternates between dueling arms and pulling arms. We show that DF can get the best of both worlds: It chooses the easier way between duels and pulls to elinimate suboptimal arms.
- Our main contribution is on the continuous MAB-DC problem. Here we consider MAB-DC in the setting of optimizing a nonconvex blackbox function $f : \mathcal{X} \to \mathbb{R}$ using as few queries to (a noisy version of) $f(x)$ as possible, and with no gradient information directly available. We propose the COMP-GP-UCB algorithm. When we can obtain comparisons based on the target function, we show that comparisons can be as powerful as direct queries: COMP-GP-UCB can achieve the *same* rate of convergence as its label-only counterparts, while using only comparisons and no direct queries. This solves the open problem raised in [160], to develop continuous dueling bandit algorithms with no-regret guarantees.
- Next, we assume that comparisons are based on a misspecified function f_c, where f_c approximates f. COMP-GP-UCB in this case uses comparisons to optimize a function f_r which has the same optimizer as f_c, and then use direct queries to search in a smaller region for the optimum of the target function. The regret rate of COMP-GP-UCB is then better than the label-only counterparts, and it depends on the difference between f_c and f: the better the approximation, the lower the regret we can get from COMP-GP-UCB. We further demonstrate a version of the algorithm that adapts to this difference. Our algorithm also extends multi-fidelity GP optimization to the setting where information is transferred actively from a lower fidelity to a higher fidelity while only assuming that the optimizer of the lower fidelity (source function) is within a constant distance of the optimizer of the higher fidelity (target function), instead of the fidelities being close everywhere.
- In our experiments, we test DF and COMP-GP-UCB against existing algorithms, and show that they achieves a superior performance under synthetic and real-world settings.

4.3 Related Work

Our MAB-DC extends the framework of dueling bandits, and we refer readers to [42] for a review. The existing works on dueling bandits mostly focus on the discrete K-armed dueling bandit case [74, 199, 206]. Most of these algorithms have a regret rate logarithmic in T; however this regret rate depends on a constant lower bound on the ε function, i.e., a constant gap between the optimal

arm and the other arms. Our MAB-DC takes both pulls and duels, and can have a logarithmic regret if the gap from duels is small but the gap from pulls is large.

The continuous case of multi-armed bandits is also variously known as zeroth order non-convex optimization or black-box optimization[41, 49, 76, 79, 89, 175]. Although zeroth order *convex* optimization is generally efficient [96], optimizing a *non-convex* f under smoothness constraints requires the same effort as estimating f almost everywhere, and usually leads to a query complexity exponential in d, where d is the feature space dimensionality [49, 76, 79, 175]. We build our work on GP-UCB[156], a method for optimizing unknown functions under the Gaussian process (GP) assumption by optimizing the Upper Confidence Bound (UCB). Closest to our setting is a line of recent research on multi-fidelity GP optimization [103, 104, 148], which assumes that we can query the target functions at multiple fidelities of different costs and precisions. We detail the relation and difference of our setting with multi-fidelity optimization in Section 4.5.5. To briefly describe it, our setting is harder since we cannot directly query the function on which comparisons are based. Moreover, the multi-fidelity assumptions such as fidelities being close in sup-norm do not hold for our setting since any constant shift of the comparison function yields the same comparisons. We instead consider an **active transfer learning** setting where information from a function that can be learned using comparisons is transferred actively to optimize the target function (refer to Section 4.5 for details).

For dueling bandits in the continuous case, [8] studied a reduction to the traditional MAB problem; and [160] proposes an algorithm based on Thompson sampling to deal with comparisons. Continuous dueling bandits has also been studied under the framework of derivative-free optimization [96, 110]. Kumagai [110] obtains optimal regret rates for a convex f. However to the best of our knowledge, no previous work has theoretical guarantees on optimizing a non-convex f. Also, these results cannot be applied when the comparisons are biased (i.e., a Condorcet winner on comparisons might not be the best point for direct queries).

Finally, there is another line of research that combines direct queries and comparisons for classification or regression problems [105, 185, 187]. Our methods differ from theirs because we focus on the optimization setting, and only care about points near the optimum. These methods make direct queries across the whole feature space to learn the underlying function well, which is unnecessary for optimization.

4.4 Discrete Case: The K-Armed MAB-DC Problem

In this section, we first consider the discrete case where the arm space $\mathcal{X} = \{x_1, ..., x_K\}$, where K is the number of arms. We propose an algorithm to combine duels and pulls, so that we get the best of both worlds: It uses either duels or pulls to eliminate suboptimal arms, and as a result the cumulative regret is the smaller one between the duel regret and the pull regret. We call our algorithm Double Filtering (DF) since it uses both duels and pulls. We introduce our notation in Section 4.4.1, and algorithm with theoretical results in Section 4.4.2. We verify our theoretical results on synthetic data in Section

4.4.1 Problem Setup

To define the MAB-DC problem, we make assumptions on the pull and duel results. Suppose the pull rewards are random variables with means $\mu_1, ..., \mu_K$. We assume the reward distribution of each bandit is sub-Gaussian with parameter R:

$$E[\exp(tY_i - t\mu_i)] \leq \exp(R^2t^2/2)$$

for all $t \in \mathbb{R}$. We also assume the dueling bandits setup for each pair of bandits. In each iteration, we can also choose to duel a pair of bandits (i, j), and we have $P_{ij} = \Pr[x_i \succ x_j] = \varepsilon_{ij} + 1/2$.

We assume a total ordering over the arms for duels, and WLOG suppose it is $x_1 \succ x_2 \succ \cdots \succ x_K$. We assume that the comparisons satisfy the strong stochastic transitivity: $\varepsilon_{ik} \geq \max\{\varepsilon_{ij}, \varepsilon_{jk}\}$ for $i \succ j \succ k$. We also assume the triangle inequality that $\varepsilon_{ik} \leq \varepsilon_{ij} + \varepsilon_{jk}$. We do not need the pull reward means $\mu_1, ..., \mu_K$ to satisfy this ordering, but we assume that $\mu_1 \geq \max_{i=2,...,K} \mu_i$.

Upon making a query, we accumulate regret

$$r_t = \begin{cases} \max\{\mu_1 - \mu_i, \varepsilon_{1i}\}, & \text{if pull arm } x_i, \\ \max\{\frac{2\mu_1 - \mu_i - \mu_j}{2}, \frac{\varepsilon_{1i} + \varepsilon_{1j}}{2}\}, & \text{if duel arm } x_i, x_j. \end{cases}$$

Our goal is to minimize the cumulative regret r_t over a time frame T. Note that here the regret is defined based on the number of queries; i.e., we assume that duels and pulls cost the same.

4.4.2 Algorithm and Analysis

The algorithm is described in Algorithm 7. The algorithm alternates between dueling bandits and traditional MAB on the working set W_l. Conceptually, DF combines the process of Beat-the-Mean Bandits [199] and Successive Elimination [70]. We pick these two algorithms because they both have the working set concept; we can potentially combine other algorithms that uses a current working set as well.

The following theorem shows the performance guarantee of DF.

Theorem 45. *Let* $c(n^*) = 3\sqrt{\frac{1}{n}\log(4KT)}$. [1] *Suppose* $T \geq K$. *Let* $\Delta_k = \mu_1 - \mu_k$. *The regret of Algorithm 7 is at most*

$$R(T) \leq O\left(\sum_{k=2}^{K} \frac{\log T}{\max\{\varepsilon_{1k}, \Delta_k\}}\right).$$

Remark. Theorem 45 illustrate a best-of-both-worlds scenario for MAB-DC: the regret rate depends on the larger gap between the duels and pulls. This is achieved by the shared working set among the duel part and the pull part. Note that this larger gap is arm specific, meaning that for each suboptimal arm, DF finds whether it is easier to eliminate it through duels or pulls.

[1] We assume that the time horizon T is known before the algorithm starts, following [199]. If it is unknown, we can use a $\log(t^2)$ factor instead and this will introduce another $\log(\max\{\varepsilon_{1k}, \Delta_k\})$ factor in the cumulative regret.

Algorithm 7 Double Filtering

Input: Bandits $\mathcal{B} = \{x_1, ..., x_K\}$

$W_l \leftarrow \mathcal{B}, l \leftarrow 1$
$\forall b \in \mathcal{B}, n_b \leftarrow 0, w_b \leftarrow 0, m_b \leftarrow 0, s_b \leftarrow 0$
Define $\hat{P}_b \equiv w_b/n_b, \forall b \in \mathcal{B}$, or $\hat{P}_b = 1/2$ if $n_b = 0$
Define $\hat{\mu}_b \equiv s_b/m_b, \forall b \in \mathcal{B}$, or $\hat{\mu}_b = 0$ if $m_b = 0$
Define $n^* \equiv \min_{b \in W_l} n_b$, $m^* \equiv \min_{b \in W_l} m_b$
while Budget not exhausted and $|W_l| > 1$ **do**
    $b = \arg\min_{b \in W_l} n_b$ ▷ Break ties randomly
    Select $b' \in W_l$ at random, compare b with b'
    If b wins, $w_b \leftarrow w_b + 1$
    $n_b \leftarrow n_b + 1$
    if $\min_{b \in W_l} \hat{P}_{b'} + c(n^*) \leq \max_{b \in W_l} \hat{P}_b - c(n^*)$ **then**
        $b' = \arg\min_{b \in W_l} \hat{P}_b$
        For all $\tilde{b} \in W$, remove comparison with b' from $n_{\tilde{b}}, w_{\tilde{b}}$
        $W_l \leftarrow W_l \setminus \{b'\}$
        $l \leftarrow l + 1$
    end if
    $b = \arg\min_{b \in W_l} m_b$ ▷ Break ties randomly
    Pull arm b and get feedback Y
    $s_b \leftarrow s_b + Y$
    $m_b \leftarrow m_b + 1$
    if $\min_{b \in W_l} \hat{\mu}_{b'} + c(m^*) \leq \max_{b \in W_l} \hat{\mu}_b - c(m^*)$ **then**
        $b' = \arg\min_{b \in W_l} \hat{\mu}_b$
        $W_l \leftarrow W_l \setminus \{b'\}$
        For all $\tilde{b} \in W$, remove comparison with b' from $n_{\tilde{b}}, w_{\tilde{b}}$
        $l \leftarrow l + 1$
    end if
end while

Output: Any arm $b \in W_l$

4.4.3 Experiments

We verify our theoretical insights by running experiments on synthetic data. We compare to the following baselines:

BeatTheMean: We compare to the duel-only [199], which is essentially one component of our method.

SuccessiveElimination: We use successive elimination as a pull-only baseline.

UCB: We also compare with the widely-used upper confidence bound baseline.

Our experiment setup resembles that in [199]. In all settings, we use $T = 5 \times 10^6$ and compute the cumulative regret, while varying the number of arms from 40 to 200. We consider two settings of setting up the reward values of each arm:

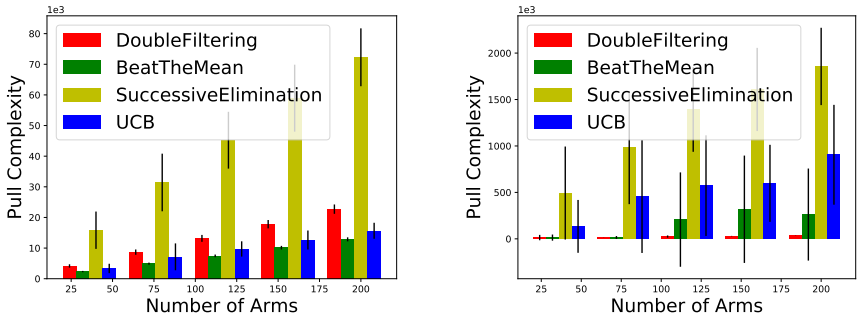
- **Consistent setting**: We generate a set of arm rewards μ_i for $i \in [K]$ by sampling from

uniform distribution on $[0, 1]$. We set $\mu_0 = 1$ and let arm 0 be the best arm. When pulling an arm i, the algorithm suffers regret $1 - \mu_i$, and gets a feedback of $\mu_i + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 1)$. When the algorithm duels two arms i, j, it gets feedback with distribution $\Pr[i \succ j] = 1/(1 + \exp((\mu_j - \mu_i)/\lambda))$, following the BTL model with parameter λ; in our experiment we use $\lambda = 0.01$ to ensure that duels and pulls have a similar gap value.

- **Inconsistent setting**: We generate two sets of arm values μ_i, μ'_i following uniform distribution on $[0, 1]$. We fix $\mu_0 = \mu'_0 = 1$ so that arm 0 is the best arm. We use μ_i as the reward mean for every arm i, but use μ'_i for generating the comparisons. Therefore in this case we will have different (and independent) gaps for duels and pulls.

In both settings, we generate the comparisons following In the consistent setting, all the arms have the same gap in duels and pulls, so we would expect all the baselines to perform well; however in the inconsistent setting, the gaps are different in duels and pulls. In other words, an arm with a small gap on duels might have large gap on pulls, and therefore large gap on its regret. In this case we expect our DoubleFiltering algorithm to work well. We repeat the experiment for 40 times and record the standard deviation.

Figure 4.1 partially verifies our observation. In the consistent setting, most methods come with a small variance, while DoubleFiltering incurs a slightly worse regret than BeatTheMean and UCB. This is because DoubleFiltering uses both duels and pulls to find the best arm, and therefore takes twice the number of queries to identify the best arm in this case. On the other hand, in the inconsistent settings pull-only and duel-only algorithms suffers a large regret and a large variance, but DoubleFiltering still performs very well in the inconsistent case. Overall our results show that DoubleFiltering is robust to the discrepancy between pulls and duels.



(a) Consistent Setting (b) Inconsistent Setting

Figure 4.1: Results on synthetic data. The bars indicates standard deviation from 20 experiments.

4.5 Continuous Case: MAB-DC for Optimizing a Nonconvex Function

We consider MAB-DC with a continuous arm space $\mathcal{X} \subseteq \mathbb{R}^d$. We set our problem in the zeroth-order nonconvex optimization setting, i.e., we aim to maximize a function $f : \mathcal{X} \to \mathbb{R}$. In each

iteration t of optimization, we can query (expensive) direct queries to f at a chosen point x_t, and obtain $y = f(x_t) + \varepsilon$, $\varepsilon \in [-\eta, \eta]$ and $E[\varepsilon] = 0$, with $\eta > 0$ a known constant[2].

Comparison Probabilities. In each iteration we can also choose to obtain (cheap) comparisons for a pair of points $(x_t, x_t') \in \mathcal{X} \times \mathcal{X}$. We assume that comparisons are based on a function f_c which can be potentially different from f (as described later in this section). A common assumption in the literature is to use a link function to assume a distribution of the comparisons, i.e., we assume $\Pr[x \succ x'] = \sigma(f_c(x) - f_c(x'))$ for some function σ. Common link functions include logistic function (BTL model[38]), or Gaussian cdf (Thurstone model [162]).

Connecting comparisons and direct queries. To make comparisons helpful for optimization, we also require that f_c is a good approximation of f. Here we differentiate between two settings:

- Dueling-Choice Bandits with unbiased comparisons: We assume comparison comes from the same function as the target function, i.e., $f_c = f$ or, more generally, that f_c and f have the same optimizer ($\zeta = 0$ as described below). This may be the case when comparison and direct queries come from the same agent, such as the preference elicitation example in the introduction.
- Dueling-Choice Bandits with misspecified comparisons: We assume $f_c \approx f$. In many cases, comparisons are from a different source (e.g. experts) than direct queries (e.g. experiments), and this can result in a biased f_c. To this end, we assume a bounded difference near the optimum:

  Assumption 1. *Let $f^* = \max_x f(x)$ and $f_c^* = \max_x f_c(x)$. There exists a constants ζ such that for any point $x \in \mathcal{X}$ we have $|(f_c^* - f_c(x)) - (f^* - f(x))| \leq \zeta$.*

  In words, when we get ε-close to the maximum of f, we are at least $(\varepsilon + \zeta)$-close to the maximum of f_c, and vice versa. Under this assumption, we would require both comparison and direct queries if we want to achieve optimization error smaller than ζ.

  We note that our results can be generalized to the case where Assumption 1 only holds for $x \in \{x : f^* - f(x) \leq \tau\}$ for some fixed constant τ.

Smoothness Assumptions. We assume that the target function f lies in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_\kappa$ induced by kernel κ, and that the RKHS norm of f is bounded: $\|f\|_\kappa \leq B$ for a known constant B. This assumption is also analyzed in [56, 156] for traditional bandits. We note that every function $f \in \mathcal{H}_\kappa$ has a finite kernel norm. When κ is the linear kernel, $\|f\|_\kappa \leq B$ induces that f is B-Lipschitz.

Budgets and Regrets. We analyze the problem of optimizing f under a given cost budget Λ. Suppose a direct query costs λ_l units of some resource and a comparison costs $\lambda_c < \lambda_l$. Also, let $\overline{n}_\Lambda = \lceil \frac{\Lambda}{\lambda_c} \rceil$ be the upper bound on number of queries when we use all the budget on comparisons, and $\underline{n}_\Lambda = \lfloor \frac{\Lambda}{\lambda_l} \rfloor$ be the corresponding lower bound when we only use direct queries. Also let $q_t =$ label if we make direct queries at iteration t, and $q_t =$ comp otherwise. We analyze the

[2]Our methods can also be extended to the setting where ε follows a sub-Gaussian distribution with parameter η. We assume a bounded ε for simplicity here.

simple regret under budget Λ, defined as follows:

$$S(\Lambda) = \min_t r_t \tag{4.1}$$
$$= \min_t \begin{cases} f^* - f(x_t) & \text{if } q_t = \text{label}, \\ \min\{f^* - f(x_t), f^* - f(x'_t)\}, & \text{if } q_t = \text{comp}. \end{cases}$$

In words, we calculate the minimum regret achieved by either comparison or direct queries. We compute simple regret over all direct queries; for comparisons, we adopt the notion of weak regret employed in [200]. Here we choose simple regret because our target is to optimize function f, and cumulative regret is typically not relevant for our setting. Our method can also be easily extended to the optimizer error setting, where the algorithm gives an estimation of the optimum when it ends. In analyzing the regret rates, we use $O(\cdot)$ to ignore constants, and $\tilde{O}(\cdot)$ to ignore log terms in the regret bounds.

4.5.1 The Gaussian Process Back End

We base our methods on Gaussian Process, with kernel function κ. If f was sampled from the Gaussian process $\mathcal{GP}(0, \kappa)$, and the direct queries were coming from f plus a Gaussian noise, i.e., $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^t$ with $y_i = f(x_i) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \eta^2)$, then the posterior distribution at $f(x)|\mathcal{D}$ would be a Gaussian $\mathcal{N}(\mu_t(x), \sigma_t(x))$ with

$$\mu_t(x) = k^T (K + \eta^2 I_t)^{-1} Y, \tag{4.2}$$
$$\sigma_t(x) = \kappa(x, x) - k^T (K + \eta^2 I_t)^{-1} k.$$

Here $Y = (y_1, ..., y_t)^T$, $k = (\kappa(x, x_1), ..., \kappa(x, x_t))^T$, and matrix $K \in \mathbb{R}^{t \times t}$ is given by $K_{ij} = \kappa(x_i, x_j)$, and I_t is the $t \times t$ identity matrix.

Remark. We note that the Gaussian noise and prior is only assumed to derive updates to the mean and variance in the algorithm, and we do not assume the actual feedbacks follow a Gaussian model, nor the functions are sampled from the Gaussian process. We only assume that f have bounded norm in $\mathcal{H}_\kappa$ and that ε is bounded in $[-\eta, \eta]$, as stated in the beginning of Section 4.5. This is the same as the *agnostic* setting in GP-UCB [56, 156].

The Maximum Information Gain. As in previous works on GP [56, 103], our results will depend on the *maximum information gain* [156] between function measurements and the function values, defined as below:

Definition 1. *Suppose $A \subseteq \mathcal{X}$ is a subset of feature space, and $\tilde{A} = \{x_1, ..., x_n\} \subseteq A$ is a finite subset of A. Then the maximum information gain on A with n evaluations is defined as $\Phi_n(A) = \max_{\tilde{A} \subseteq A, |\tilde{A}|=n} I(f_{\tilde{A}} + \varepsilon_{\tilde{A}}; f_{\tilde{A}})$, where $f_{\tilde{A}} = [f(x)]_{x \in \tilde{A}}$, $\varepsilon_{\tilde{A}} \sim \mathcal{N}(0, \eta^2 I)$, and $I(X, Y) = H(X) - H(X|Y)$ is the mutual information.*

When $\mathcal{X} \subseteq \mathbb{R}^d$ is compact and convex, [156] shows that i) for linear kernel κ, $\Phi_n(\mathcal{X}) = O(d \log n)$; ii) for squared exponential (SE, or RBF) kernel, $\Phi_n(\mathcal{X}) = O((\log n)^{d+1})$; iii) For Matérn kernels $\kappa(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} (\frac{\sqrt{2\nu}z}{\rho})^\nu B_\nu (\frac{\sqrt{2\nu}z}{\rho})$, we have $\Phi_n(\mathcal{X}) = O\left(n^{\frac{d(d+1)}{2\nu + d(d+1)}} \log n\right)$.

Review of GP-UCB and IGP-UCB. Previous sequential optimization has adopted the upper confidence bound (UCB) principle, where we maintain a high-confidence upper bound $\phi : \mathcal{X} \to \mathbb{R}$

for all $x \in \mathcal{X}$, such that $f(x) \leq \phi(x)$ with high probability. Our algorithm builds on UCB algorithms for GP, namely GP-UCB [156] and IGP-UCB [56] (the latter is an improvement of the former).

The GP-UCB [156] and IGP-UCB [56] can be unified as in Algorithm 8. In time step t of optimization, IGP-UCB queries the point that maximizes the confidence upper bound in the form $\mu_{t-1}^{(l)}(x) + \beta_t \sigma_{t-1}^{(l)}(x)$, where $\mu_{t-1}^{(l)}, (\sigma_{t-1}^{(l)})^2$ are the posterior mean and variance function of the GP from step $t-1$, and β_t is a multiplier that increases with t. In time step t of optimization, IGP-UCB queries the point that maximizes the confidence upper bound in the form $\mu_{t-1}^{(l)}(x) + \beta_t \sigma_{t-1}^{(l)}(x)$, where $\mu_{t-1}^{(l)}, (\sigma_{t-1}^{(l)})^2$ are the posterior mean and variance function of the GP from step $t-1$, and β_t is a multiplier that increases with t. The algorithms only differ at their assumptions and thus the choice of β_t. Our setting of $\beta_t^{(r)}$ and β_t is similar to IGP-UCB, as we focus more on the agnostic function setting.

Algorithm 8 GP-UCB and IGP-UCB

Input: Budget Λ

1: Set $D_0^l = \emptyset$, $(\mu_0^{(l)}, \sigma_0^{(l)}) = (0, \kappa^{1/2}), t \leftarrow 0$
2: **for** $t = 1, 2, ..., \underline{n}_\Lambda$ **do**
3: Compute $x_t = \arg\max_{x \in \mathcal{X}} \mu_{t-1}(x) + \beta_t \sigma_{t-1}(x)$
4: Query $f(x_t)$ and obtain feedback y_t
5: Use y_t and (4.2) to perform posterior updates, and obtain $\mu_t^{(l)}, \sigma_t^{(l)}$
6: **end for**

4.5.2 The Borda Function f_r

A straightforward way to incorporate comparisons into optimization is to use them to compute a GP posterior of either f or f_c. However, we will face several difficulties. Firstly, the posterior based on comparisons cannot be analytically computed. Also, we cannot compute the joint posterior based on both direct queries and comparisons, since f and f_c can be different. Lastly, comparisons might not be truthful and can be inconsistent; i.e., human might give contradicting comparisons like $x_1 \succ x_2 \succ x_3 \succ x_1$ [206].

We instead consider a different function directly related to f_c, defined as $f_r(x) = \Pr[x \succ X]$, where X is randomly chosen from $\mathcal{X}$. In words, $f_r(x)$ is the probability that x beats a random point $X \in \mathcal{X}$. We refer to f_r as the Borda function, inspired by Borda scores in the dueling bandits literature [90, 206]. An advantage of using f_r is that we can obtain unbiased estimates of $f_r(x)$ by comparing x to a random point in $X \in \mathcal{X}$.

It is easy to see that f_r should have the same optimizer as f_c. We make the following assumption to ensure usefulness of comparisons:

Assumption 2. *Let $f_r^* = \max_x f_r(x)$ and $f_c^* = \max_x f_c(x)$. There exists constants L_1, L_2 such that for every $x \in \mathcal{X}$ we have $\frac{1}{L_1}(f_c^* - f_c(x)) \leq f_r^* - f_r(x) \leq L_2(f_c^* - f_c(x))$.*

In other words, difference in f_c will cause a difference of similar scale in f_r. This requires that the comparisons induces a Borda function f_r such that f_r is close to f_c at its optimum, and that f_r and f_c has the same optimizer. We note that this is a quite weak assumption, as we do not restrict

the result of comparing individual points x, x' to comply with $f_c(x) - f_c(x')$, i.e., comparisons do not need to be consistent. We can show that Assumption 2 holds under the link function setting, when σ is Lipschitz continuous:

Proposition 46. *Suppose comparisons follows a link function σ with a Lipschitz constant between $[1/L_1, L_2]$, i.e., $\frac{|\sigma(x)-\sigma(y)|}{|x-y|} \in [\frac{1}{L_1}, L_2]$, $\forall x, y \in \mathbb{R}$, then Assumption 2 holds.*

We comment that common link functions such as BTL [38] and Thurstone [162] all have bounded Lipschitz functions if f_c is bounded.

Lastly, we note that [8] also compare x to a random point X, and use the feedback to update the function estimates. However, their method relies on a linear link function $\sigma(x) = \frac{1+x}{2}$ and cannot be applied for BTL or Thurstone models.

Algorithm 9 COMP-GP-UCB

Input: Comparison bias ζ, comparison exploration threshold γ, confidence δ

Set $D_0^r = D_0^l = \emptyset$, $(\mu_0^{(r)}, \sigma_0^{(r)}) = (\mu_0^{(l)}, \sigma_0^{(l)}) = (0, \kappa^{1/2})$, $t \leftarrow 0$
repeat
  Compute $x_t = \arg\max_{x \in \mathcal{X}} \mu_{t-1}^{(r)}(x) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x)$
  QUERY(x_t, comp)
until $\beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t) \leq \gamma$ or budget exhausted
Let $\hat{f}_r = \mu_{t-1}^{(r)}(x_t) - \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t)$
while Budget not exhausted **do**
  Let $\phi_t^{(r)}(x) = \mu_{t-1}^{(r)}(x) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x) - \hat{f}_r + L_2 \zeta$
  Compute $x_t = \arg\max_{x \in \mathcal{X}: \phi_t^{(r)}(x) \geq 0} \mu_{t-1}^{(l)}(x) + \beta_t \sigma_{t-1}^{(l)}(x)$
  if $\beta_t^{(r)}(x_t) \sigma_{t-1}^{(r)}(x_t) \geq \gamma$ **then** QUERY(x_t, comp)
  else QUERY(x_t, label)
  end if
  $t \leftarrow t + 1$
end while

procedure QUERY(query point x_t, query type q_t)
  if $q_t =$ comp **then**
    Sample x' randomly from $\mathcal{X}$ and query to compare (x_t, x'), obtain z_t
    Update $D_t^c \leftarrow D_{t-1}^c \cup \{(x_t, z_t)\}$, $D_t^l \leftarrow D_{t-1}^l$
    Perform Bayesian update for $\mu_t^{(r)}, \sigma_t^{(r)}$ based on D_t^c with $y_t = z_t$ following (4.2)
  else
    Query direct labels for x_t and obtain y_t
    Update $D_t^l \leftarrow D_{t-1}^l \cup \{(x_t, y_t)\}$, $D_t^c \leftarrow D_{t-1}^c$
    Perform Bayesian update for $\mu_t^{(l)}, \sigma_t^{(l)}$ based on D_t^l following (4.2)
  end if
end procedure

4.5.3 The COMP-GP-UCB Algorithm

We describe our COMP-GP-UCB along with its analysis in this section. When ζ is known and given, COMP-GP-UCB is formally described in Algorithm 9. Our algorithm works both for unbiased comparisons ($\zeta = 0$) and misspecified comparisons ($\zeta > 0$). COMP-GP-UCB is an anytime algorithm, meaning that it does not need to know the total budget Λ before it begins. For any input $\zeta \geq 0$, the high-level idea is to constrain the search region for f using comparisons to the set $\mathcal{H} := \{x : f_r(x) \geq f_r^* - L_2\zeta\}$ where $f_r^* = \max_x f_r(x)$. $\mathcal{H}$ is guaranteed to contain the optimizer f under our assumptions; To see this, let x^* be any optimizer of f, and we have $f_r^* - f_r(x^*) \leq L_2(f_c^* - f_c(x^*)) \leq L_2(f^* - f(x^*) + \zeta) = L_2\zeta$. The first inequality follows from Assumption 2 and the second one follows from Assumption 1. It is easy to see that $\mathcal{H}$ is much smaller than $\mathcal{X}$ if comparisons are mostly correct (i.e., ζ is small); therefore we can explore more efficiently by restricting the search on $\mathcal{H}$.

COMP-GP-UCB takes as input ζ, a parameter γ to control exploration on comparisons, and a confidence level δ. We keep track of posteriors $(\mu^{(l)}, \sigma^{(l)}), (\mu^{(r)}, \sigma^{(r)})$ for f and f_r respectively, and construct confidence intervals $\mu_{t-1}^{(l)}(x) \pm \beta_t \sigma_{t-1}^{(l)}(x)$, $\mu_{t-1}^{(r)}(x) \pm \beta_t^{(r)} \sigma_{t-1}^{(r)}(x)$. Since f_r is unknown, to approximate $\mathcal{H}$, the algorithm adopts a two-phase approach: In the first phase (Step 2-5), we optimize f_r using comparison queries until $\beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t) \leq \gamma$, i.e., the queried point has confidence of at least γ. At the end of the first phase, we compute $\hat{f}_r$ as a lower bound for f_r^*. Next, we start the second phase exploring f (Step 7-14). We select the query point x_t based on a filtering $\phi_t^{(r)}(x) \geq 0$; the filtering approximates the constraint set $\mathcal{H}$ by combining the current UCB of f_r and the LCB $\hat{f}_r$ from the first phase. Then the algorithm optimizes the UCB of f under the constraint of $\phi_t^{(r)}(x) \geq 0$. While doing this, we check the UCB of f_r at the maximizer x_t and if we are not confident about $f_r(x_t)$, we query a comparison, or otherwise we make a direct query as in GP-UCB.

The query process is described in the procedure QUERY. For direct queries, we query x_t directly, and update the posterior of f according to (4.2); for comparisons, we compare x_t with a random point x', and use the result as feedback to update posterior of f_r. We note that this comparison result is an unbiased estimate of $f_r(x_t)$.

The Two-Phase Approach. Both phases are critical for the algorithm to succeed. The first phase is important in two ways: Firstly, it helps to get a low regret in the unbiased comparisons setting, and in the initial stages of the algorithm when only comparison queries are used for the biased (misspecified comparison) setting. Also, it gives a lower bound $\hat{f}_r \leq f_r^*$ of the optimum of f_r at Step 6 which will be used to approximate the constraint set $\mathcal{H}$. Then we use the second phase to obtain low regret in the biased comparison case.

Choice of $\phi_t^{(r)}$. The choice of $\phi_t^{(r)}$ is critical for the algorithm to succeed. We want that the region $S = \{x : \phi_t^{(r)}(x) \geq 0\}$ is not too small or too large: we need that every maximizer x^* of f is in S, while also excluding as many points as possible using the information from f_r. To achieve the former, we have added $L_2\zeta$ to the confidence interval to account for the difference in f_c and f. To achieve the latter, we need both a good UCB of f_r and a good LCB of $f_r^* = \max f_r(x)$. The good UCB is ensured by the check at Step 10; we only make direct queries when we are confident enough about $f_r(x_t)$. The good LCB is ensured by the first phase, where we compute $\hat{f}_r$; without the first phase $\hat{f}_r$ can be arbitrarily bad and it will lead to suboptimal direct queries. In the proof

we show that when $\phi_t^{(r)}(x) \geq 0$ and $\beta_t^{(r)}(x)\sigma_{t-1}^{(r)}(x) \geq \gamma$, x belongs to an approximation of $\mathcal{H}$. So the two constraints combined ensure that we use direct queries to explore $\mathcal{H}$.

We now present our theoretical results. We defer full proofs to the appendix due to space constraints. We first analyze the unbiased comparison case. In this case, we have $\zeta = 0$, and we only need comparisons to achieve low regret. Therefore we run COMP-GP-UCB with $\zeta = \gamma = 0$; in this case, the algorithm only executes the first phase, and only uses comparisons to optimize f_r. We obtain the following guarantee.

Theorem 47. *Suppose Assumption 2 holds, and $f_c = f$. Let $\beta_t^{(r)} = 2B + \sqrt{2\left(\Phi_{t-1}(\mathcal{X}) + 1 + \log(1/\delta)\right)}$. There exists a constant C dependent on d, κ such that COMP-GP-UCB with $\zeta = \gamma = 0$ has a simple regret bounded by*

$$S(\Lambda) \leq C\left(B + \sqrt{(\Phi_{\overline{n}_\Lambda}(\mathcal{X}) + \log(1/\delta))}\right)\sqrt{\frac{\Phi_{\overline{n}_\Lambda}(\mathcal{X})}{\overline{n}_\Lambda}}. \tag{4.3}$$

Remark. IGP-UCB [56] in the label-only setting has regret of form

$$S_{\text{IGP-UCB}}(\Lambda) \leq \tag{4.4}$$
$$C\left(B + \sqrt{\left(\Phi_{\underline{n}_\Lambda}(\mathcal{X}) + \log(1/\delta)\right)}\right)\sqrt{\frac{\Phi_{\underline{n}_\Lambda}(\mathcal{X})}{\underline{n}_\Lambda}},$$

where $\underline{n}_\Lambda = \lfloor \frac{\Lambda}{\lambda_l} \rfloor$. This is the same form as (4.3), but with $\overline{n}_\Lambda$ replaced with $\underline{n}_\Lambda$. Recall that $\overline{n}_\Lambda$ is the number of queries when we use all the budget on comparisons, and $\underline{n}_\Lambda$ is the number for using all budget on direct queries. In other words, COMP-GP-UCB *has the same rate as IGP-UCB as if direct queries are as cheap as comparisons*. When comparisons are much cheaper than direct queries, COMP-GP-UCB leads to a great advantage by significantly reducing the number of direct queries needed.

We then analyze COMP-GP-UCB in the misspecified comparison setting($\zeta > 0$). In this setting, comparisons act as a filter on $\mathcal{X}$ to reduce the search region for direct queries. When f_c approximates f well (i.e., a small ζ), the set $\mathcal{H} = \{x : f_r(x) \geq f_r^* - L_2\zeta\}$ is much smaller than the feature space $\mathcal{X}$. Therefore by using comparisons, we wish to replace the $\Phi_{n_0}(\mathcal{X})$ term in (4.4) by $\Phi_{n_0}(\mathcal{H})$, effectively exploring a smaller region. We show that COMP-GP-UCB can have a similar behavior by exploring on a slightly larger set dependent on γ, defined as $\mathcal{H}^\gamma = \{x \in \mathcal{X} : f_r(x) \geq f_r^* - L_2\zeta - 4\gamma\}$. The following theorem characterizes the regret of COMP-GP-UCB under the misspecified comparison setting.

Theorem 48. *Suppose Assumptions 2 and 1 hold, and ζ is known. Let $\beta_t^{(r)}$ be the same as in Theorem 47, and $\beta_t = 2B + \sqrt{2\left(\Phi_{t-1}(\mathcal{H}^\gamma) + 1 + \log(1/\delta)\right)}$. There exists constants Λ_0, C dependent on $\zeta, \gamma, B, d, \kappa$ such that if when $\Lambda \geq \Lambda_0$ we have $S(\Lambda) \leq \min\{S_1, S_2\}$, where*

$$S_1 = 2L_1\gamma + \zeta +$$
$$C\left(B + \sqrt{(\Phi_{\overline{n}_\Lambda}(\mathcal{X}) + \log(1/\delta))}\right)\sqrt{\frac{\Phi_{\overline{n}_\Lambda}(\mathcal{X})}{\overline{n}_\Lambda}},$$
$$S_2 = C\left(B + \sqrt{\left(\Phi_{\underline{n}_\Lambda}(\mathcal{H}^\gamma) + \log(1/\delta)\right)}\right)\sqrt{\frac{\Phi_{\underline{n}_\Lambda}(\mathcal{H}^\gamma)}{\underline{n}_\Lambda}}.$$

We discuss about the bounds and setup of parameters before coming to the proof of Theorem 48.

Remarks. The regret bound in Theorem 48 enjoys best of both worlds from comparisons and direct queries. The first bound has the same form as in Theorem 47 but with another $2L_1\gamma + \zeta$ term. This comes from the first phase of COMP-GP-UCB, and the extra term comes from the fact that $f_c \neq f$. In the second phase, the algorithm achieves the second bound S_2, which is the rate of using $\underline{n}_\Lambda$ direct queries to explore $\mathcal{H}^\gamma$. Compared with (4.4), the second bound has the same rate on $\underline{n}_\Lambda$, but with a reduced search region $\mathcal{H}^\gamma$ and a startup budget Λ_0 for comparisons to work. When f_c is a good approximation to f, $\mathcal{H}^\gamma$ is much smaller than $\mathcal{X}$ and will lead to a great improvement in the number of direct queries needed.

Setup of parameters. 1. Setting γ: γ acts as a threshold for exploring comparisons in both phases of COMP-GP-UCB. A small γ will lead to a small $\mathcal{H}^\gamma$ and therefore better regret rates; but it will also make the algorithm spend more time on comparisons before moving to direct queries, i.e., a large Λ_0. One plausible choice for γ is to set $\gamma = \frac{1}{L_2}\zeta$, and this will make $\mathcal{H}^\gamma \approx \mathcal{H}$.
2. Setting β_t: The setup for β_t in Theorem 48 requires knowing $\Phi_t(\mathcal{H}^\gamma)$ before algorithm starts and this is unrealistic to set up. However, in practice the default choice for β_t is often very loose and hand-tuned values are used instead (e.g., [103] uses $\beta_t = 0.2d\log(2t)$). In this sense this setup for β_t does not affect its practical use. For theoretical purposes, we can also set $\beta_t = \beta_t^{(r)}$; this leads to a regret rate of $\tilde{O}\left(\frac{(B+\sqrt{\Phi_{\underline{n}_\Lambda}(\mathcal{X})})\sqrt{\Phi_{\underline{n}_\Lambda}(\mathcal{H}^\gamma)}}{\sqrt{\underline{n}_\Lambda}}\right)$, slightly larger than the current rate but still smaller than GP-UCB.

Proof Sketch. We prove Theorem 48 and Theorem 47 follows as a corollary. For the first bound, if we have left phase 1 and entered phase 2, let T_0 be the time that we leave phase 1. By routine calculation we can show

$$S(\Lambda) \leq f^* - f(x_{T_0}) \leq L_1(f_r^* - f_r(x_{T_0})) \leq 2L_1\gamma + \zeta. \tag{4.5}$$

On the other hand, if we do not finish phase 1 (e.g., when $\zeta = \gamma = 0$), we can follow the proof of IGP-UCB [56] and show that

$$S(\Lambda) \leq C\beta_{\overline{n}_\Lambda}^{(r)}\sqrt{\frac{\Phi_{\overline{n}_\Lambda}(\overline{\mathcal{H}^\gamma})}{\overline{n}_\Lambda}} + \zeta. \tag{4.6}$$

Combining (4.5) and (4.6) we get the first bound S_1.

Now we show the second bound S_2. Suppose the algorithm makes n queries. For any set $A \subseteq \mathcal{X}$, let $T_n^r(A)$ be the number of comparison queries into A when the algorithm has made n queries, and $T_n^l(A)$ be the number of direct queries. We have

$$n = T_n^r(\mathcal{X}) + T_n^l(\overline{\mathcal{H}^\gamma}) + T_n^l(\mathcal{H}^\gamma).$$

For the first term, we show that there exists a constant C_κ such that $T_n^r(\mathcal{X}) \leq C_\kappa\left(\frac{\beta_n^{(r)}}{\gamma}\right)^{p+2}$, where $p = d$ for SE kernel and $p = 2d$ for Matérn kernel. For the second term, we show that our algorithm makes sure that $T_n^l(\overline{\mathcal{H}^\gamma}) = 0$, i.e., it always query in $\overline{\mathcal{H}^\gamma}$ when it uses direct queries. These two results combined can show that we allocate at least $\underline{n}_\Lambda/2$ direct queries to explore $\mathcal{H}^\gamma$. The second bound S_2 then follows by bounding the regret similar to IGP-UCB. □

Algorithm 10 COMP-GP-UCB for unknown ζ

Input: Threshold γ, comparison bias starting point ζ_0, bias upper bound $\zeta_{\max}$, budget Λ

1: Set $D_0^r = D_0^l = \emptyset$, $(\mu_0^{(r)}, \sigma_0^{(r)}) = (\mu_0^{(l)}, \sigma_0^{(l)}) = (0, \kappa^{1/2})$, $t \leftarrow 0, k \leftarrow 0, N_l \leftarrow 0$
2: **repeat**
3: Compute $x_t = \arg\max_{x \in \mathcal{X}} \mu_{t-1}^{(r)}(x) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x)$
4: QUERY(x_t, comp)
5: **until** $\beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t) \leq \gamma$
6: Let $\hat{f}_r = \mu_{t-1}^{(r)}(x_t) - \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t)$
7: **while** $\zeta_k \leq \zeta_{\max}$ **do**
8: Let $\phi_t^{(r)}(x) = \mu_{t-1}^{(r)}(x) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x) - \hat{f}_r + 2L_2\zeta_k$
9: Compute $x_t = \arg\max_{x \in \mathcal{X}: \phi_t^{(r)}(x) \geq 0} \mu_{t-1}^{(l)}(x) + \beta_t \sigma_{t-1}^{(l)}(x)$
10: **if** $\beta_t^{(r)}(x_t) \sigma_{t-1}^{(r)}(x_t) \geq \gamma$ **then** QUERY(x_t, comp)
11: **else**
12: QUERY(x_t, label)
13: $N_l \leftarrow N_l + 1$
14: **end if**
15: **if** $N_l \geq \frac{\underline{n}_\Lambda}{2\lceil \log(\zeta_{\max}/\zeta_0) \rceil}$ **then**
16: $N_l \leftarrow 0, \zeta_{k+1} \leftarrow 2\zeta_k, k \leftarrow k + 1$
17: **end if**
18: $t \leftarrow t + 1$
19: **end while**

4.5.4 COMP-GP-UCB with Unknown ζ

In practice, we cannot know ζ in general, and it is even hard to verify whether $\|f_c - f\|_\infty \leq \zeta$ holds for a given ζ. On the other hand, we can often know an upper bound $\zeta_{\max}$ such that $\zeta \leq \zeta_{\max}$. For example, if we know both f and f_c are bounded in $[-B_\infty, B_\infty]$ we naturally have $\|f_c - f\|_\infty \leq 2B_\infty$. However, Algorithm 9 is not useful if we set $\zeta = 2B_\infty$, because that will lead to a constraint set $\mathcal{H} = \{x : f_r(x) \geq f_r^* - 2L_2B_\infty\}$ that can be as large as $\mathcal{X}$ and we have to explore the whole feature space with direct queries. We develop a slightly different method in Algorithm 10 that tries to search ζ between an initial value ζ_0 and the upper bound $\zeta_{\max}$, and adapts to the true ζ.

Algorithm 10 works in the finite-horizon scenario, where the budget Λ is given as input. The process of Algorithm 10 is mostly similar to Algorithm 9, except that it uses ζ_k in the second phase in place of ζ. We optimize the function as if Assumption 1 holds for ζ_k. ζ_k starts from ζ_0; at step 15, once we have spent enough queries at the current estimate of ζ, we double the current ζ_k. We iterate until we reach $\zeta_k > \zeta_{\max}$. The threshold for N_l, $\frac{\underline{n}_\Lambda}{2\lceil \log(\zeta_{\max}/\zeta_0) \rceil}$, is chosen such that we divide a label budget of $\underline{n}_\Lambda/2$ direct queries equally among the $\lceil \log(\zeta_{\max}/\zeta_0) \rceil$ possible values of the ζ_k's. The constant 2 is chosen arbitrarily here; any choice of $\underline{n}_\Lambda/c$ for a constant $c > 1$ will obtain the same rate.

We present our theoretical results as a corollary to Theorem 48. Since we cannot find the exact ζ, our results depend on a slightly larger $\bar{\zeta} = \max\{2\zeta, \zeta_0\}$. We use $\hat{H}^\gamma = \{x \in \mathcal{X} : f_r(x) \geq$

$f_r^* - 2L_2\bar{\zeta} - 4\gamma\}$ to represent the constraint set of interest when ζ is replaced by $\bar{\zeta}$.

Corollary 49. *Suppose Assumptions 2 and 1 hold, and $\zeta \leq \zeta_{\max}$. Under the same setting of $\beta_t^{(r)}, C, \Lambda_0$ as in Theorem 48, and $\beta_t = 2B + \sqrt{2\left(\Phi_{t-1}(\hat{H}^\gamma) + 1 + \log(1/\delta)\right)}$, the simple regret of Algorithm 10 satisfies $S(\Lambda) \leq \min\{S_1, S_2\}$, where*

$$S_1 = 2L_1\gamma + 2\zeta + C\left(B + \sqrt{(\Phi_{\overline{n}_\Lambda}(\mathcal{X}) + \log(1/\delta))}\right)\sqrt{\frac{\Phi_{\overline{n}_\Lambda}(\mathcal{X})}{\overline{n}_\Lambda}},$$

$$S_2 = C\left(B + \sqrt{\left(\Phi_{\underline{n}_\Lambda}(\hat{H}^\gamma) + \log(1/\delta)\right)}\right)\sqrt{\frac{\Phi_{\underline{n}_\Lambda}(\hat{H}^\gamma)}{\underline{n}_\Lambda}}.$$

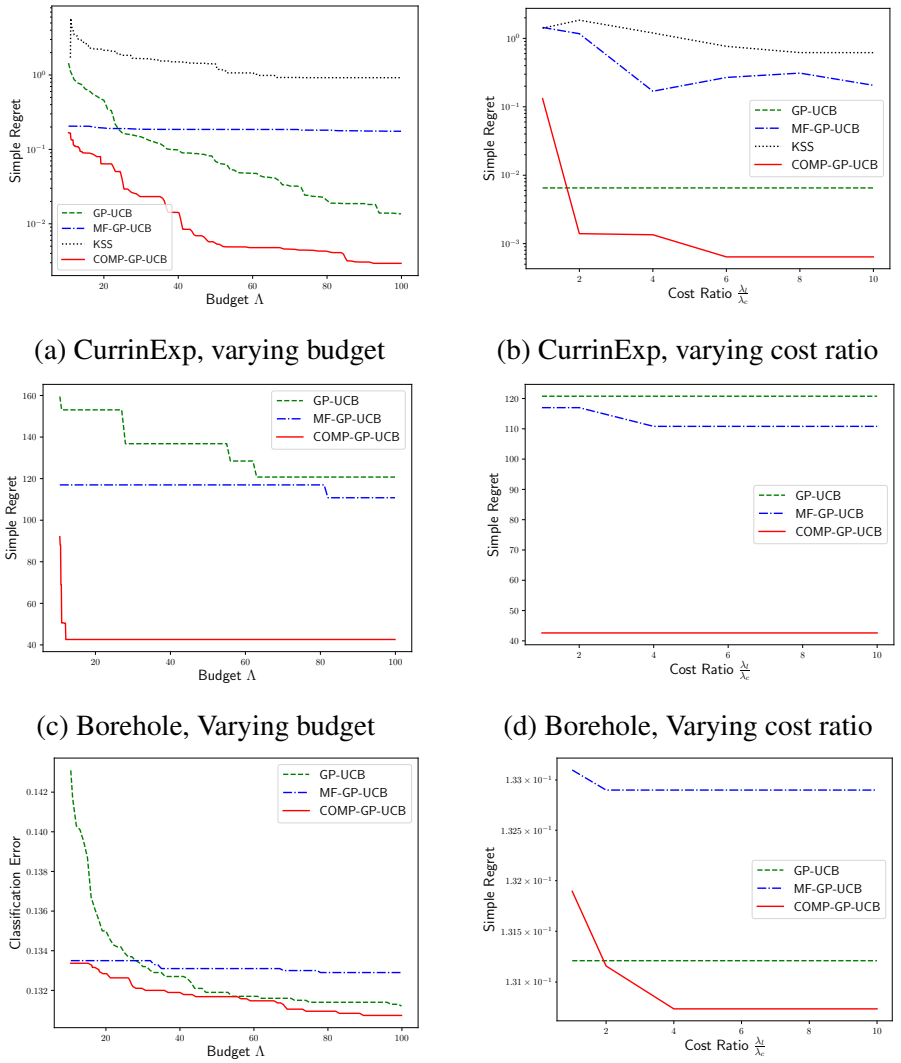
Remark. 1. The regret rate of Corollary 49 is almost the same as Theorem 48, except the set $\mathcal{H}^\gamma$ is replaced with $\hat{H}^\gamma$. We note that all the terms in the regret rate depend only on $\bar{\zeta}$ or ζ, and do not depend on $\zeta_{\max}$. This means Algorithm 10 can adapt to unknown level of comparison bias ζ. 2. Similar to Theorem 48, Corollary 49 also requires the unknown quantity $\Phi_t(\hat{H}^\gamma)$ to set β_t; in practice we can also use a similar hyper-parameter search to find this quantity. γ also takes a similar effect as in Algorithm 9, and $\gamma = \frac{1}{L_2}\zeta_0$ can lead to $\hat{\mathcal{H}}^\gamma \approx \mathcal{H}$ and a practical algorithm. Again, setup of these parameters only depends on $\bar{\zeta}$ and is not affected by $\zeta_{\max}$.

4.5.5 Comparison with MF-GP-UCB [103]

Our setting and method share some common characteristics as the multi-fidelity method MF-GP-UCB [103], and we formally discuss them here. Our setup is similar to MF-GP-UCB in the two-fidelity case, where the algorithm has access to the target function f and its approximation $f^{(1)}$, with $\|f - f^{(1)}\|_\infty \leq \zeta$ for some known $\zeta > 0$. Although we also assume f_c is a good approximation for f (in a weaker sense of being close in terms of f^* and f_c^*, see Assumption 1), our setting is harder than MF-GP-UCB and their algorithm cannot be directly applied in our case. This is because we cannot directly query f_c: f_c is only available through comparisons, and we will get the same set of comparisons from f_c and $f_c + c$ for any constant c. In our case, we can only get unbiased estimates for f_r. However, it is unlikely that $\|f_r - f\|_\infty$ is small, because $f_r(x) \in [0, 1]$ for all x since it is the probability of beating a random point, whereas f can have arbitrary values.

MF-GP-UCB bears some resemblance to the second phase in our Algorithm 9, but they are principally different in choosing the next query point x_t. In the MF-GP-UCB algorithm, we have access to another function f' similar to f. The algorithm constructs two sets of UCBs $\phi(x), \phi'(x)$ for f and f' separately, and use $\min\{\phi(x), \phi'(x)\}$ as a final UCB. In our case, UCBs of f_r and f are not comparable. Instead we use a novel constrained optimization approach based on observations in the first phase.

Another difference is that MF-GP-UCB needs the function difference ζ known beforehand, whereas our modified COMP-GP-UCB (Algorithm 10) can adapt to an unknown ζ. We note that MF-GP-UCB does use a doubling mechanism in their experiments to make it practical, but they do not provide any theoretical guarantees.



(a) CurrinExp, varying budget

(b) CurrinExp, varying cost ratio

(c) Borehole, Varying budget

(d) Borehole, Varying cost ratio

(e) SVM Tuning, varying budget

(f) SVM Tuning, varying cost ratio

Figure 4.2: Empirical results comparing COMP-GP-UCB with baseline methods. KSS stands for KernelSelfSparring.

4.5.6 Experiments

We perform experiments against plausible baselines to verify our theory and illustrate the efficacy of our algorithm, on both synthetic and real-world data. For comparison, we include state-of-art algorithms in the label-only and comparison-only setting, as well as an adapted version of MF-GP-UCB, as described below.

Experiment Setup. For synthetic data, we use functions from the multi-fidelity literature to produce comparisons on a lower fidelity and direct labels on a higher fidelity. In particular we use Currin exponential (CurrinExp, $d = 2$) and Borehole ($d = 8$) [184] functions. We note that f and f_c have different values and maximizers. We do NOT add additional noise on direct queries for f.

For real-world data, we experiment with the SVM tuning task from the MF-GP-UCB paper

[103] and train a SVM classifier on the MAGIC Gamma dataset [68]. We tune the RBF kernel bandwidth and the soft margin coefficient within range $(10^{-3}, 10^{1})$ and $(10^{-1}, 10^{5})$. We randomly sample a training set of size 2,000 and a validation set of 500 from the original dataset. Direct labels take in the specified bandwidth and margin coefficient and return the corresponding validation accuracy. On the other hand, comparisons only use a (fixed) subset of size 500 from the training set (noisy but cheap) and return the validation accuracy on the same validation data.

Baselines. We evaluate the performance of COMP-GP-UCB against the following baselines: (1) GP-UCB[156]: The label-only algorithm optimizing UCB of GP posterior. (2) KernelSelfSparring[160]: A comparison-only algorithm that uses Thompson Sampling to optimize comparisons. We note that since $f \neq f_c$, optimizing comparisons cannot lead to the global optimum. (3) MF-GP-UCB[103]: Although MF-GP-UCB is not directly applicable in our case, we try to use it by using comparisons as the lower fidelity. When the algorithm selects to query the lower fidelity on x_t, we compare x to a random point $X \in \mathcal{X}$ and use the result as feedback, the same process as COMP-GP-UCB.

Implementation Details. All methods use the RBF kernel for GP. For all methods we compute the simple regret (4.1) w.r.t. f [3]. The results are averaged over 20 runs.

We apply basic techniques in Bayesian Optimization to conduct the experiments.

- *Initial queries:* All the algorithms were initialized with uniform random queries with an initial budget of $\Lambda_0 = 10$. For multi-fidelity methods (MF-GP-UCB and COMP-GP-UCB), we use $\Lambda_0/2$ on comparisons and $\Lambda_0/2$ on direct queries; for GP-UCB we use all Λ_0 on labels.
- *Choice of kernel parameters:* We estimate the kernel bandwidth and scale by maximizing marginal likelihood with respect to the initial random queries. We also update the kernel parameters by maximizing the marginal likelihood for the GP over the lower fidelity function after every 20 iterations, and for the GP over the true function after every 5 iterations.
- *Setup of β_t and $\beta_t^{(r)}$* We follow MF-GP-UCB and set $\beta = 0.5 * \log(2 * t + \epsilon) = \beta_t^{(r)}$.
- *Choosing query points.* We use the DiRect algorithm [101] to maximize the marginal likelihood subject to parameter bounds, and to find the next query points.
- *Choice of hyperparameters ζ, γ.* For synthetic experiments we set $\zeta = f(x^*) - f_c(x^*)$, where x^* is the maximizer of f. For experiments on real data, we heuristically set ζ to be a large enough value (we use 0.15 in our experiments). We initialize $\gamma = \text{range}(f) \cdot \zeta$, and whenever COMP-GP-UCB has queried 10 comparisons in a row, we double γ (following MF-GP-UCB).

Cost Ratio. In practice, the relation between the costs of labels and comparisons can be complex. We call $\frac{\lambda_l}{\lambda_c}$ the *cost ratio* between labels and comparisons; the larger the cost ratio, the cheaper the comparisons. Our algorithm generally works for a cost ratio $\frac{\lambda_l}{\lambda_c} > 1$. We test the performance under various cost ratios in our experiment. Unless otherwise specified, for all the experiments with a varying budget we follow the setup of MF-GP-UCB and use $\lambda_c = 0.1$ and $\lambda_l = 1$, and use a total budget of up to $\Lambda = 100$. For all the experiments with a varying cost ratio, we set $\Lambda = 100, \lambda_l = 1$, and vary the cost ratio between $[1, 10]$.

Results on synthetic data. The results are summarized in Figure 4.2a & 4.2b. Firstly we compare

[3] We find that KernelSelfSparring is extremely slow for $d = 8$ so we only test it for CurrinExp.

the performance on CurrinExp by varying the total budget from 10 to 100 (Figure 4.2a). COMP-GP-UCB shows the best performance over all budget setups. It is worth noting that MF-GP-UCB performs worse than label-only GP-UCB in our setting; this is because the target function of MF-GP-UCB in this case essentially optimizes the function f_r, which is bounded in $[0, 1]$, resulting in a very large approximation bias. In contrast, COMP-GP-UCB is able to use comparisons in an efficient way to reduce the search space for optimization. In the appendix, we also include the case where $f = f_c$ and $\lambda_c = \lambda_l$ (i.e., no bias on comparisons and cost ratio equals 1).

Then in Figure 4.2b, we fix the total budget to be $\Lambda = 100$ and cost of labels $\lambda_l = 1$, and vary the cost ratio from 1 to 10 by varying comparison costs. COMP-GP-UCB achieves the best performance for all setups except for $\lambda_c = \lambda_l = 1$ when it is worse than the label-only GP-UCB algorithm. This is expected since our algorithm targets to use cheaper comparisons. Our algorithm can be more effective even with a fairly small cost ratio (≥ 2).

The result on Borehole is depicted in Figure 4.2c. KSS is extremely slow for Borehole due to the dimension ($d = 8$), so we exclude it from the plot. As with CurrinExp, COMP-GP-UCB achieves the best performance with a large gap under all budget setups. Different than CurrinExp, COMP-GP-UCB displays an advantage over the baselines in all cost ratios including $\lambda_l = \lambda_c$ (Figure 4.2d). This might suggest that it is easier to find the maximum using f_r (the lower fidelity) than using f (the higher fidelity).

Alternative Definitions of Regret. Since the simple regret notion defined for settings with both direct and comparison queries is necessarily unfair for methods designed to handle only one query type, we perform an additional experiment to examine the comparison regret and direct query regret separately. Namely, define

$$R_c(\Lambda) = \begin{cases} \min\limits_{t, q_t = \text{comp}} f^* - f(x_t), & \text{if } \exists t \text{ s.t } q_t = \text{comp}, \\ \infty & \text{otherwise.} \end{cases}$$

and

$$R_l(\Lambda) = \begin{cases} \min\limits_{t, q_t = \text{label}} f^* - f(x_t), & \text{if } \exists t \text{ s.t } q_t = \text{label}, \\ \infty & \text{otherwise.} \end{cases}$$

In Figure 4.3, we plot the comparison regret R_c and direct query regret R_l for the CurrinExp synthetic function (same setting as Figure 4.2a). We plot R_c and R_l respectively for both COMP-GP-UCB and MF-GP-UCB, R_l for GP-UCB, and R_c for KSS. Our results show that even if we only consider comparisons or direct queries, COMP-GP-UCB still outperforms the baselines in most budget settings. Overall, COMP-GP-UCB has a large direct query regret initially because we mostly do comparisons in the initial stage, but it achieves a low regret when we have a larger budget. We note that COMP-GP-UCB can still query both comparisons and direct queries in this setting, and it leads to a better regret for both comparisons and direct queries.

Results with Single Fidelity and Same Cost. Although COMP-GP-UCB is designed for the case where comparisons are cheaper than direct queries, it is also interesting to see how it performs when comparisons and direct queries cost the same. We conduct an experiment with $f = f_c$ and $\lambda_c = \lambda_l = 1$ in Figure 4.4. Note that MF-GP-UCB is not applicable in this setting since it is for the multi-fidelity setting. COMP-GP-UCB performs on par (for CurrinExp) or better (for Borehole) than the baselines in our result. Note that while our theory suggests that the convergence rate of

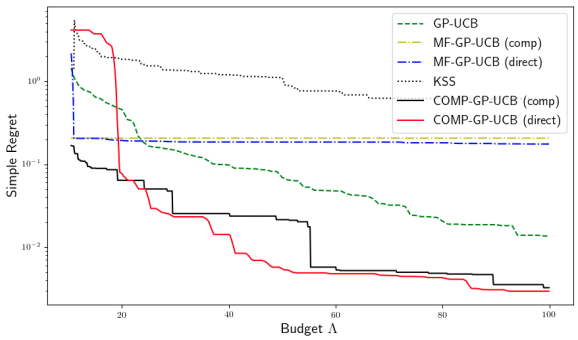
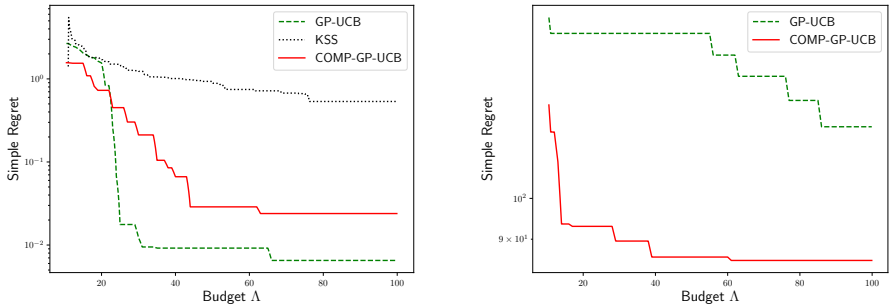


Figure 4.3: Result comparing comparison and direct regret.



(a) CurrinExp (b) Borehole

Figure 4.4: Empirical results comparing COMP-GP-UCB with baselines, under a single fidelity and $\lambda_c = \lambda_l = 1$. KSS stands for KernelSelfSparring.

COMP-GP-UCB is the same as GP-UCB when $\lambda_c = \lambda_l$ and $f = f_c$, in practice the underlying function f_r (of COMP-GP-UCB) might be easier to optimize than f, because it is bounded and can be smoother than f. For the case of CurrinExp, f_r and f are of similar shape and values, so COMP-GP-UCB leads to a slower convergence rate because comparisons are less informative than direct queries; for Borehole, the f_r appears to be easier to optimize than the higher fidelity. One possible reason is that f_r is smoother in shape than f since it is bounded in $[0, 1]$; this interesting phenomenon needs further investigation as part of future work. We note that cost ratio $\frac{\lambda_l}{\lambda_c} = 2$ is enough for COMP-GP-UCB to surpass the performance of GP-UCB on CurrinExp (see Figure 4.2b).

Results on real data. The results are in Figure 4.2e & 4.2f. KernelSelfSparring (KSS) has an error rate much higher than other methods (larger than 0.16), so we exclude it in the plot. Similar to the synthetic data case, COMP-GP-UCB outperforms other baselines. The advantage of COMP-GP-UCB over GP-UCB is smaller than the synthetic case, which might possibly result from the larger difference between f_c and f. Note that for real data we still vary the cost ratio because in practice the actual cost ratio might depend on various factors like computational cost and data collection. Nevertheless, Figure 4.2f shows that COMP-GP-UCB has an advantage over the baselines for cost ratio at least 2; this is very likely to happen since the training set for comparisons is only 1/4 the size of that for direct queries.

4.6 Conclusion

We consider a novel dueling-choice setting when both direct queries and comparisons are available for discrete and continuous multi-armed bandits problem. For the discrete case, we propose the DF algorithm that takes the best of both worlds from duels and pulls. For the continuous case, we propose the COMP-GP-UCB algorithm that can achieve benign regret rates in the dueling-choice setting, and can adapt to unknown biases in the comparisons. Our algorithm can also be of independent interest for other multi-fidelity or transfer learning settings where information gleaned from one fidelity or source domain can be actively transferred to optimize the target domain function, under milder conditions than existing literature.

Our method takes a two-phase approach in the biased comparison case for continuous optimization. One important future work is to investigate whether the two phases can be combined in one joint phase for better guarantees and performance. In general with many types of queries, corralling bandits [3, 10] might also be helpful for solving optimization with both direct and comparison queries.

4.7 Proofs

4.7.1 Proof of Theorem 45

Proof. Firstly we can show that for each arm and each time step, with probability $1 - 1/(4KT)$, the confidence intervals we compute is correct. We compute the regret under this event E_0.

We first use the following lemma:

Lemma 50 (Lemma 3, [199])**.** *Under event E_0, arm b_k is removed from W_l if $n_k \geq O\left(\frac{\log T}{\varepsilon_{1k}^2}\right)$.*

Lemma 51. *Under event E_0, arm b_k is removed from W_l if $m_k \geq O\left(\frac{\log(T)}{\Delta_k^2}\right)$.*

Now consider a variant of Algorithm 7, where in addition to removing duels with b' in Step 13 and 24, we also remove $\tilde{b}$ in $m_{\tilde{b}}$ and $s_{\tilde{b}}$. This ensures that $m_b = n_b$ for all b and we always choose the same b in Step 7 and 17.

Suppose in round l (i.e., when we remove the l-th arm), the worst arm by duels is $\varepsilon_{1k_1} = \arg\max_{k \in W_l} \varepsilon_{1k}$ and worst arm by pulls is $\Delta_{k_2} = \arg\max_{k \in W_l} \Delta_k$. Let $\gamma = \max\{\varepsilon_{1k_1}, \Delta_{k_2}\}$ and $\nu_k = \max\{\varepsilon_{1k}, \Delta_k/\}$ for every arm b_k. Consider two ways that an arm can be removed:
i) Some arm j is removed by duels. Consider the number of duels we remove: the first part is n_j duels that initiated from b_j. We know that $n_j \leq \min\{\frac{\log T}{\varepsilon_{1k_1}^2}, \frac{\log T}{\Delta_{k_2}^2}\} \leq \frac{\log T}{\gamma^2}$, since otherwise arm k_1 or k_2 will be removed. The other part is the duels that were initiated from other arm $b_{j'} \in W_l$. Note that n_k for every $k \in W_l$ only is at most $\frac{\log T}{\gamma}$, since every time we remove one arm, n_k cannot be too large or otherwise we will remove arm b_k. So in total there are at most $O(|W_l|\frac{\log T}{\gamma^2})$ duels from other arms in W_l. Note that the other arm from these comparisons is uniformly distributed in W_l, and thus the expectation of number of such removals is at most $O(\frac{\log T}{\gamma^2})$. So from Chernoff bound we know that with probability $1 - 1/T$ the number of removals is at most $O(\frac{2\log T}{\gamma^2})$. Therefore the incurred regret from the removed duels is at most $O(\nu_j \cdot \frac{\log T}{\gamma^2}) \leq O(\frac{\log T}{\gamma})$.

Similarly, we remove $O(\frac{\log T}{\gamma^2})$ pulls from m_j. Also following the same statement we incur at most $O(\frac{\log T}{\gamma})$ regret from the removed pulls.

ii) Some arm j is removed by pulls. Following the same argument, we incur at most $O(\frac{\log T}{\gamma})$ regret from the removed duels and pulls.

Putting everything together, we calculate that the total regret is bounded by $O(\sum_{l=1}^{K-1} \frac{\log T}{\gamma_l})$, where γ_l is the value of γ in round l. It is straightforward to see that this value is bounded by the rate in the statement of the theorem.

□

4.7.2 Proof of Proposition 46

Let x_c^* be a maximizer of f_c. Because of the link function assumption, x_c^* is also a optimizer of f_r. We have

$$\begin{aligned} f_r^* - f_r(x) &= \mathbb{E}[\sigma(f_c(x_c^*) - f_c(X))] - \mathbb{E}[\sigma(f_c(x) - f_c(X))] \\ &= \mathbb{E}[\sigma(f_c(x_c^*) - f_c(X)) - \sigma(f_c(x) - f_c(X))] \\ &\leq \mathbb{E}[L_2 | f_c(x_c^*) - f_c(x)|] = L_2(f_c^* - f_c(x)). \end{aligned}$$

The lower bound can be proved similarly.

4.7.3 Proof of Theorem 47 and 48

We show Theorem 48 and Theorem 47 follows as a direct corollary. We first use results in IGP-UCB[56] to obtain confidence bands of f_r:

Lemma 52 (Theorem 2, [56]). *Define* $\beta_t^{(r)} = 2\|f_r\|_\kappa + \sqrt{2\left(\Phi_{t-1}(\mathcal{X}) + 1 + \log(2/\delta)\right)}$. *Then with probability* $1 - \delta/2$ *we have for all time* t *and any point* $x \in \mathcal{X}$,

$$|\mu_{t-1}^{(r)}(x) - f_r(x)| \leq \beta_t^{(r)} \sigma_{t-1}^{(r)}(x).$$

Lemma 52 also applies to $f, x \in \mathcal{H}^\gamma, \mu^{(l)}, \sigma^{(l)}$ by setting $\beta_t = 2\|f\|_\kappa + \sqrt{2\left(\Phi_{t-1}(\mathcal{H}^\gamma) + 1 + \log(1/\delta)\right)}$.

We also use the following lemma to bound the sum of posterior variances:

Lemma 53 (Lemma 8, [103]). *Let* $A \subseteq \mathcal{X}$. *Suppose we have n queries* $(x_t)_{t=1}^n$ *of which* s *points are in* A. *Then the posterior* σ_t *satisfies*

$$\sum_{x_t \in A} \sigma_{t-1}^2(x_t) \leq \frac{2}{\log(1 + \eta^{-2})} \Phi_s(A).$$

Suppose the event in Lemma 52 holds for f and f_r. We first prove the first bound by looking at comparison queries. Firstly, in the first phase when we compute x_t in step 3 we have

$$\begin{aligned} f_r^* - f_r(x_t) &\leq \mu_{t-1}^{(r)}(x_r^*) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_r^*) - (\mu_{t-1}^{(r)}(x_t) - \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t)) \\ &\leq \mu_{t-1}^{(r)}(x_t) + \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t) - (\mu_{t-1}^{(r)}(x_t) - \beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t)) \\ &= 2\beta_t^{(r)} \sigma_{t-1}^{(r)}(x_t). \end{aligned} \tag{4.7}$$

The first inequality uses Lemma 52, and the second inequality is from that x_t is the maximizer of $\mu_{t-1}^{(r)}(x) + \beta_t^{(r)}\sigma_{t-1}^{(r)}(x)$.

Suppose we finish phase 1 and enter phase 2. Let T_0 be the time we leave phase 1, then we must have $\beta_{T_0}^{(r)}\sigma_{T_0-1}^{(r)}(x_{T_0}) \le \gamma$. So

$$\begin{aligned} S(\Lambda) &\le f^* - f(x_{T_0}) \\ &\le f_c(x^*) - f_c(x_{T_0}) + \zeta \\ &\le f_c^* - f_c(x_{T_0}) + \zeta \\ &\le L_1(f_r^* - f_r(x_{T_0})) + \zeta \le 2L_1\gamma + \zeta. \end{aligned} \tag{4.8}$$

The second inequality is from Assumption 1, the fourth inequality is from Assumption 2, and the last inequality is from (4.7).

If we do not finish phase 1, then the number of comparison queries is $N_1 \ge \overline{n}_\Lambda - 1$ and $N_1 \le \overline{n}_\Lambda$, and we have

$$\begin{aligned} \left(\sum_t (f_r^* - f_r(x_t))\right)^2 &\le N_1 \sum_{t:q_t=\text{comparison}} (f_r^* - f_r(x_t))^2 \\ &\le N_1 \sum_{t:q_t=\text{comparison}} 4\left(\beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t)\right)^2 \\ &\le C_1 N_1 \left(\beta_{N_1}^{(r)}\right)^2 \Phi_{N_1}(\mathcal{H}^\gamma). \end{aligned} \tag{4.9}$$

Here $C_1 = \frac{8}{\log(1+\eta^{-2})}$. The first step is from the Cauchy-Schwarz inequality, and the second step is from (4.7); the last step uses Lemma 53.

So

$$\begin{aligned} S(\Lambda) &\le \frac{1}{N_1}\sum_t (f^* - f(x_t)) \\ &\le \frac{2L_1}{N_1}\sum_t (f_r^* - f_r(x_t)) + \zeta \\ &\le \frac{\beta_{N_1}^{(r)}}{N_1}\sqrt{C_1 N_1 \Phi_{N_1}(\mathcal{H}^\gamma)} + \zeta \\ &\le \frac{C\beta_{\overline{n}_\Lambda}^{(r)}}{\overline{n}_\Lambda}\sqrt{\overline{n}_\Lambda \Phi_{\overline{n}_\Lambda}(\mathcal{H}^\gamma)} + \zeta \\ &\le C\left(B + \sqrt{(\Phi_{\overline{n}_\Lambda}(\mathcal{H}^\gamma) + \log(1/\delta))}\right)\sqrt{\frac{\Phi_{\overline{n}_\Lambda}(\mathcal{H}^\gamma)}{\overline{n}_\Lambda}} + \zeta. \end{aligned} \tag{4.10}$$

The second inequality follows from the same process as in (4.8); the third inequality follows from (4.9); the fourth inequality is from $\overline{n}_\Lambda \ge N_1 \ge \overline{n}_\Lambda - 1$. Here C is a constant whose value may change from line to line. Combining (4.8) and (4.10) we get the first bound.

To show the second bound, we examine the regret from direct queries. We first show that x^* is never excluded from our feasible region:

Claim 54. $\phi_t^{(r)}(x^*) \geq 0$ *for all* t.

Proof. Suppose x^* is a maximizer of f in $\mathcal{X}$. Then we have

$$\begin{aligned}\phi_t^{(r)}(x^*) &= \mu_t^{(r)}(x^*) + \beta_t^{(r)}\sigma_t^{(r)}(x^*) - \max_{x'}\left\{\mu_t^{(r)}(x') - \beta_t^{(r)}\sigma_t^{(r)}(x')\right\} + L_2\zeta \\ &\geq f_r(x^*) - f_r^* + L_2\zeta \\ &= -L_2(f_c^* - f_c(x^*)) + L_2\zeta \\ &\geq -L_2(f^* - f(x^*) + \zeta) + L_2\zeta \\ &= -L_2\zeta + L_2\zeta = 0.\end{aligned}$$

The first inequality is from Lemma 52; the second inequality is from Assumption 1. □

Let N be the (random) total number of queries under budget Λ. We know that the support of N lies in $[\underline{n}_\Lambda, \overline{n}_\Lambda]$; we now suppose n is any number in $[\underline{n}_\Lambda, \overline{n}_\Lambda]$, and prove properties of Algorithm 9 when it uses n queries.

For any set $A \subseteq \mathcal{X}$, let $T_n^r(A)$ be the number of comparison queries into A when the algorithm has made n queries, and $T_n^l(A)$ be the number of direct queries. We have

$$n = T_n^r(\mathcal{X}) + T_n^l(\overline{\mathcal{H}^\gamma}) + T_n^l(\mathcal{H}^\gamma)$$

since $\overline{\mathcal{H}^\gamma} \cup \mathcal{H}^\gamma = \mathcal{X}$. We bound the first two terms using the following two lemmas:

Lemma 55. *There exists a constant* C_κ *dependent on* κ, d *such that* $T_n^r(\mathcal{X}) \leq C_\kappa\left(\frac{\beta_n^{(r)}}{\gamma}\right)^{p+2}$, *where* $p = d$ *for SE kernel and* $p = 2d$ *for Matérn kernel.*

This lemma is proved in Section 4.7.4.

Lemma 56. $T_n^l(\overline{\mathcal{H}^\gamma}) = 0$.

Proof. Suppose $q_t =$ label for some t. Then we must have $\phi_t^{(r)}(x_t) \geq 0$, and that $\beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t) < \gamma, \beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t^{(r)}) < \gamma$, here $x_t^{(r)}$ is the value of x_t on line 6. Then we have

$$\begin{aligned}f_r(x_t) &\geq \mu_{t-1}^{(r)}(x_t) - \beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t) \\ &= \phi_t^{(r)}(x_t) - 2\beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t) + \max_{x'}\left\{\mu_{t-1}^{(r)}(x') - \beta_t^{(r)}\sigma_{t-1}^{(r)}(x')\right\} - L_2\zeta \\ &\geq \phi_t^{(r)}(x_t) - 2\beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t) + \mu_{t-1}^{(r)}\left(x_t^{(r)}\right) - \beta_t^{(r)}\sigma_{t-1}^{(r)}\left(x_t^{(r)}\right) - L_2\zeta \\ &\geq 0 - 2\gamma + f_r^* - 2\gamma - L_2\zeta \\ &= f_r^* - 4\gamma - L_2\zeta.\end{aligned}$$

The first inequality is by applying 52; the second inequality is by letting $x' = x_t^{(r)}$; the third inequality is by noticing that

$$\mu_{t-1}^{(r)}\left(x_t^{(r)}\right) - \beta_t^{(r)}\sigma_{t-1}^{(r)}\left(x_t^{(r)}\right) \geq \left[\mu_{t-1}^{(r)}\left(x_t^{(r)}\right) + \beta_t^{(r)}\sigma_{t-1}^{(r)}\left(x_t^{(r)}\right)\right] - 2\beta_t^{(r)}\sigma_{t-1}^{(r)}\left(x_t^{(r)}\right) \geq f_r^* - 2\gamma.$$

□

Lemma 55 shows that we will not make too many queries on comparisons, whereas Lemma 56 shows that we always query $x_t \in \mathcal{H}^\gamma$ when $q_t = \text{label}$. Now let N_0 be the smallest number such that for any $\Lambda \geq N_0 \lambda_c$ we have

$$\lambda_l \left(\frac{2r\sqrt{d}}{\varepsilon_{\overline{n}_\Lambda}} \right)^d \left(\frac{2\eta \beta^{(r)}_{\overline{n}_\Lambda}}{\gamma} \right)^2 = C_\kappa \left(\frac{\beta^{(r)}_{\overline{n}_\Lambda}}{\gamma} \right)^{p+2} \leq \frac{\Lambda}{2}$$

where C_κ and p are defined in (4.12). Such Λ_0 is guaranteed to exist, since $\beta^{(r)}_{\overline{n}_\Lambda}$ grows in sublinear rate for linear, SE and Matérn kernels on $\overline{n}_\Lambda$, and therefore Λ. Thus the number of queries to f_c, $T^c_n(\mathcal{X})$, is at most $\underline{n}_\Lambda/2$, and therefore we query at least $\underline{n}_\Lambda/2$ times on direct queries, $T^l_N(\mathcal{H}^\gamma) \geq \underline{n}_\Lambda/2$.

We now follow a similar path as for comparison queries to bound the regret based on direct queries. Note that if we use direct query at round t, we have $x_t \in \mathcal{H}^\gamma$ and that

$$\begin{aligned} f^* - f(x_t) &\leq \mu^{(l)}_{t-1}(x^*) + \beta_t \sigma^{(l)}_{t-1}(x^*) - (\mu^{(l)}_{t-1}(x_t) - \beta_t \sigma^{(l)}_{t-1}(x_t)) \\ &\leq \mu^{(l)}_{t-1}(x_t) + \beta_t \sigma^{(l)}_{t-1}(x_t) - (\mu^{(l)}_{t-1}(x_t) - \beta_t \sigma^{(l)}_{t-1}(x_t)) \\ &= 2\beta_t \sigma^{(l)}_{t-1}(x_t). \end{aligned} \tag{4.11}$$

The first inequality uses Lemma 52, and the second inequality uses Claim 54 and that x_t is the maximizer of $\mu^{(l)}_{t-1}(x) + \beta_t \sigma^{(l)}_{t-1}(x)$.

Now therefore

$$\begin{aligned} \left(\sum_{t:q_t=\text{label}} (f^* - f(x_t)) \right)^2 &\leq T^t_n(\mathcal{H}^\gamma) \sum_{t:q_t=\text{label}} (f^* - f(x_t))^2 \\ &\leq T^t_n(\mathcal{H}^\gamma) \sum_{t:q_t=\text{label}} 4 \left(\beta_t \sigma^{(l)}_{t-1}(x_t) \right)^2 \\ &\leq C_1 T^t_n(\mathcal{H}^\gamma) (\beta_n)^2 \Phi_{T^t_n(\mathcal{H}^\gamma)}(\mathcal{H}^\gamma) \end{aligned}$$

Here $C_1 = \frac{8}{\log(1+\eta^{-2})}$. The first step is from the Cauchy-Schwarz inequality, and that $T^l_n(\overline{\mathcal{H}^\gamma}) = 0$; the second step is from (4.11); the last step uses Lemma 53.

So

$$\begin{aligned} S(\Lambda) &\leq \frac{1}{T^t_N(\mathcal{H}^\gamma)} \sum_{t:x_t \in \mathcal{H}^\gamma, q_t=\text{label}} (f^* - f(x_t)) \\ &\leq \frac{\beta_N}{T^t_N(\mathcal{H}^\gamma)} \sqrt{C_1 T^t_N(\mathcal{H}^\gamma) \Phi_{T^t_N(\mathcal{H}^\gamma)}(\mathcal{H}^\gamma)} \\ &\leq \frac{C\beta_{\underline{n}_\Lambda}}{\underline{n}_\Lambda} \sqrt{\underline{n}_\Lambda \Phi_{\underline{n}_\Lambda(\mathcal{H}^\gamma)}(\mathcal{H}^\gamma)} \\ &\leq C \left(B + \sqrt{\left(\Phi_{\underline{n}_\Lambda}(\mathcal{H}^\gamma) + \log(1/\delta) \right)} \right) \sqrt{\frac{\Phi_{\underline{n}_\Lambda}(\mathcal{H}^\gamma)}{\underline{n}_\Lambda}}. \end{aligned}$$

4.7.4 Proof of Lemma 55

We use the following lemma from [103][4]:

Lemma 57 (Lemma 13, [103])**.** *Let $A \subseteq \mathcal{X}$ such that its L2 diameter diam$(A) \leq D$. Say we have n queries $(x_t)_{t=1}^n$ of which s points are in A. Then the posterior variance of the GP, $\kappa'(x,x)$ at any $x \in A$ satisfies*

$$\kappa'(x,x) \leq \begin{cases} C_{SE}D^2 + \frac{\eta^2}{s}, & \text{if } \kappa \text{ is the SE kernel,} \\ C_{Mat}D + \frac{\eta^2}{s}, & \text{if } \kappa \text{ is the Matérn kernel,} \end{cases}$$

for appropriate kernel dependent constants C_{SE}, C_{Mat}.

Consider the SE kernel and the comparison oracle, and a $\varepsilon_n = \frac{\gamma}{\beta_n^{(r)}\sqrt{8C_{SE}}}$ covering $(B_i)_{i=1}^n$ of $\mathcal{X}$. We claim that the number of comparison queries inside any B_i is at most $\lceil 2(\frac{\eta\beta_n^{(r)}}{\gamma})^2 \rceil$: suppose we have already queried $\lceil 2(\frac{\eta\beta_n^{(r)}}{\gamma})^2 \rceil$ samples in B_i at some time $t < n$. By Lemma 57 we have

$$\max_{x \in B_i} \kappa_{t-1}^{(r)}(x,x) \leq C_{SE}(2\varepsilon_n)^2 + \frac{\eta^2}{2(\frac{\eta\beta_n^{(r)}}{\gamma})^2} \leq \left(\frac{\gamma}{\beta_n^{(r)}}\right)^2.$$

Therefore $\beta_n^{(r)}\sigma_{t-1}^{(r)}(x) \leq \beta_t^{(r)}\sigma_{t-1}^{(r)}(x) \leq \gamma$. Note that whenever $q_t = \text{comp}$, we always have $\beta_t^{(r)}\sigma_{t-1}^{(r)}(x_t) \leq \gamma$; so the event that $q_t = \text{comp}$ and $x_t \in B_i$ will not happen until time n. We can obtain a similar result for Matérn kernel with $\varepsilon_n = \frac{\gamma^2}{4C_{Mat}(\beta_n^{(r)})^2}$. Therefore we have

$$T_n^r(\mathcal{X}) \leq \Omega_{\varepsilon_n}(\mathcal{X})\lceil 2(\frac{\eta\beta_n^{(r)}}{\gamma})^2 \rceil \leq \left(\frac{2r\sqrt{d}}{\varepsilon_n}\right)^d \left(\frac{2\eta\beta_n^{(r)}}{\gamma}\right)^2 = C_\kappa \left(\frac{\beta_n^{(r)}}{\gamma}\right)^{p+2}. \tag{4.12}$$

Here $\Omega_{\varepsilon_n}(\mathcal{X})$ is the covering number of $\mathcal{X}$, and we bound the covering number as $\Omega_{\varepsilon_n}(\mathcal{X}) \leq \left(\frac{2r\sqrt{d}}{\varepsilon_n}\right)^d$. Here $C_\kappa = 2^{2.5d+2}r^d d^{d/2}C_{SE}^{d/2}\eta^2$ and $p = d$ for SE kernel, while $C_\kappa = 2^{3d+2}r^d d^{d/2}C_{Mat}^d\eta^2$ and $p = 2d$ for Matérn kernel.

4.7.5 Proof of Corollary 49

Proof. Firstly, for the first regret bound, we have the same guarantee as in Theorem 48 since the first phase is exactly the same. For the second bound, when $\Lambda \geq \Lambda_0$, we allocate at least budget of $\Lambda/2$ on direct queries. Since we double ζ_k in each iteration, at some iteration $k_0 = O(\log(\bar{\zeta}/\zeta_0))$ we will have $\zeta_{k_0} \in [\zeta, \bar{\zeta}]$. Let $\tilde{N} = \frac{n_\Lambda}{2\log(\zeta_{\max}/\zeta_0)}$. From the proof of Theorem 48, we have the regret $S(\Lambda) \leq \beta_{\tilde{N}}\sqrt{\frac{8\Phi_{\tilde{N}}(\hat{H}^\gamma)}{\log(1+\eta^{-2})\tilde{N}}}$ in iteration k_0. The theorem then follows by realizing $\beta_n, \Phi_n(\hat{H}^\gamma)$ grows sublinearly in n.

□

[4]The original lemma from [103] assumes $f \sim GP(0,\kappa)$ and $\varepsilon \sim \mathcal{N}(0,\eta^2)$, but exactly the same proof applies without these assumptions.

Chapter 5

The Thresholding Bandit Problem with Dueling Choices

The Thresholding Bandit Problem (TBP, [120]) is an important pure-exploration multi-armed bandit (MAB) problem. Specifically, given a set of K arms with different mean rewards, the TBP aims to find arms whose mean rewards are above a pre-set threshold of τ. The TBP has a wide range of applications, such as anomaly detection, candidate filtering, and crowdsourced classification. For example, a popular crowdsourced classification model [2, 54] assumes that there are K items with the latent true labels $\theta_i \in \{0, 1\}$ for each item. The labeling difficulty of the i-th item is characterized by its soft label $\mu_i \in [0, 1]$, which is defined as the probability that a random crowd worker will label the i-th item as positive. It is clear that the item is easy to label when μ_i is close to 0 or 1, and difficult when μ_i is close to 0.5. In MAB, μ_i is the mean reward of arm i, and pulling this arm leads to a Bernoulli observation with mean μ_i. Moreover, it is natural to assume that the soft label μ_i is consistent with the true label, i.e., $\mu_i \geq 0.5$ if and only if $\theta_i = 1$. Therefore, identifying items belonging to class 1 is equivalent to detecting those arms with $\mu_i > \tau$ with $\tau = 0.5$.

Existing literature on TBP considers the setting that only solicits information from pulling arms directly. However, in many applications of TBP, comparisons/duels can be obtained at a much lower cost than direct pulls. In crowdsourcing, a worker often compares two items more quickly and accurately than labeling them separately. It will be cheaper and time efficient to ask a worker which image is more relevant to a query as compared to asking for an absolute relevance score of an image (see [153]). Another example is in material synthesis, a pull will need an expensive synthesis of the material, whereas duels can be carried out easily by querying experts. In such settings, directly pulling an arm is expensive and could incur a large sample complexity since each arm needs to be pulled a number of times. This chapter considers two sources of information: in addition to direct pulls of arms as in the classical TBP, one can also *duel* two arms to find out the arm with a greater mean at a lower cost. We refer to this problem as the TBP with Dueling Choices (TBP-DC), since dueling and pulling are both available in each round.

It is important to note that some direct pulls are still necessary for solving a TBP even if one can duel two arms. Without direct assessments of arms, we can at best rank all the arms with duels. However, we then cannot know the target threshold τ and therefore cannot identify a boundary

on the ranking. On the other hand, using an appropriate dueling strategy, the number of required direct pulls can be much lower than that in the classical TBP setting, where only direct pulling is available. We further note that TBP-DC is also different from the top-K arm identification problem considered in wither MAB (see, e.g., [40, 50, 51, 205]) or dueling bandits (see, e.g., [130]), because the number of arms with means greater than the threshold τ is unknown to us.

Our contributions. First, we develop the Rank-Search (RS) algorithm for TBP-DC, which alternates between refining the rank over all items using duels and a binary search process using pulls to figure out the threshold among ranked items. We analyze the number of duels and pulls required for RS under the fixed confidence setting, i.e., to recognize the set of arms with reward larger than τ with probability at least $1 - \delta$. To better illustrate our main idea, we further provide concrete examples, which show that the proposed RS only requires $O(\log^2 K)$ direct labels, while the classical TBP requires at least $\Omega(K)$ labels. Then, we show complementary lower bounds that RS is near-optimal in both duel and pull complexity. Finally, we provide practical experiments to demonstrate the performance of RS.

5.1 Related Works

TBP is a special case of the pure-exploration combinatorial MAB problem. As with other pure-exploration MAB problems[40] , algorithms for combinatorial bandits fall into either *fixed-budget* or *fixed-confidence* categories. In the former setting, the algorithm is given a time horizon of T and tries to minimize the probability of failure. In the latter setting, the algorithm is given a target failure probability and tries to minimize the number of queries. For TBP, the CLUCB algorithm [52] can solve TBP under the pull-only and fixed confidence setting, with optimal sample complexity. [52] also develops the CSAR algorithm for the fixed-budget setting which can also be used for TBP. The result was improved by recent followup work [120, 131] under the fixed budget setting. [54] considered TBP in the context of budget allocation for crowdsourced classification in the Bayesian framework.

Motivated by crowdsourcing and other applications, this chapter proposes a new setup since we allow both pulling one arm and dueling two arms in each round, with the underlying assumption that dueling is more cost-effective than pulling. To the best of our knowledge, this setting has not been considered in the previous work. Most close in spirit to our work is a series of recent papers [105, 185, 187], which consider using pairwise comparisons for learning classifiers. The methods in those papers are however not directly applicable to TBP-DC because their final goal is to learn a classification boundary, instead of labeling each item without feature information.

5.2 Preliminary

Suppose there are K arms, which are denoted by $\mathcal{A} = [K] = \{1, 2, ..., K\}$. Each arm $i \in \mathcal{A}$ is associated with a mean reward μ_i. Without loss of generality, we will assume that $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_K$. Given a target threshold τ, our goal is to identify the positive set $S_\tau = \{i : \mu_i \geq \tau\}$ and the negative set $S_\tau^c = \{i : \mu_i < \tau\}$.

Modes of interactions. Each instance of TBP-DC is uniquely defined by the tuple $(M, \boldsymbol{\mu})$,

where M is the preference matrix (defined below) and $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^K$ is the reward mean vector. In each round of our algorithm, we can choose one of two possible interactions:

- **Direct Queries (Pulls)**: We choose an arm $i \in \mathcal{A}$ and get a noisy reward Y from arm i. We assume that each arm i is associated a reward distribution ν_i with mean μ_i, and that ν_i is sub-Gaussian with parameter R: $\mathbb{E}_{Y\sim\nu_i}[\exp(tY - t\mathbb{E}[Y])] \leq \exp(R^2t^2/2)$ for all $t \in \mathbb{R}$. The definition of sub-Gaussian variables includes many common distributions, such as Gaussian distributions or any bounded distributions (e.g., Bernoulli distribution). We denote by $\Delta_i = |\mu_i - \tau|$ the gap between arm i and the threshold.

- **Comparisons (Duels)**: We can also choose to duel two arms $i, j \in \mathcal{A}$ and obtain a random variable Z, with $Z = 1$ indicating the arm i has a larger mean reward than j and $Z = 0$ otherwise. Let $M_{ij} \in [0, 1]$ characterize the probability that a random worker believes that arm i is "more positive" than arm j. The outcome of duels is therefore characterized by the matrix M. The (Borda) score of each arm in dueling is defined as

$$p_i := \frac{1}{K-1} \sum_{j\in[K]\setminus\{i\}} M_{ij}, \tag{5.1}$$

  i.e., the probability of arm i beating another randomly chosen arm j.

  In contrast to previous work [153, 161, 200] that usually assumes parametric or structural assumptions on M, we allow an arbitrary preference matrix M; the only assumption is that the score of any positive arm is larger than any negative arm, i.e., $p_i > p_j, \forall i \in S_\tau, j \in S_\tau^c$. We note that this is a very weak condition since arbitrary relations within the positive and negative sets are allowed. This assumption also holds if $(1, 2, ..., K)$ is the Borda ranking of M; or the underlying comparison model follows the Strong Stochastic Transitivity (SST, [75, 151]). We note that the problem is very difficult under this assumption: For example, even if μ_i are bounded away from τ by a constant, the p_i (knowledge gained from duels) may be arbitrarily close, hence making the problem much harder.

Taking crowdsourced binary classification as an example, $Y_i \in \{0, 1\}$ would correspond to a binary label of the i-th item obtained from a worker, where $\mu_i = \Pr_{Y_i\sim\nu_i}[Y = 1]$. For this case we have $\tau = 1/2$. Dueling outcome Z_{ij} will correspond to asking a worker to compare item i with item j and $Z_{ij} = 1$ if the worker claims that item i is "more positive" than item j.

The fixed-confidence setting. Given a target error rate δ, our goal is recover the sets $\hat{S}_\tau$ and $\hat{S}_\tau^c$, such that $\Pr[S_\tau = \hat{S}_\tau, S_\tau^c = \hat{S}_\tau^c] \geq 1 - \delta$, with as fewer pulls and duels as possible. Since in practice duels are often cheaper than pulls, we want to minimize the number of pulls while also avoiding too many duels.

5.2.1 Problem Complexity

We define two problem complexities w.r.t pulls and duels separately.

Pull complexity. Following previous works on TBP and pure-exploration bandits [52, 120], we introduce the following quantity to characterize the intrinsic problem complexity with direct

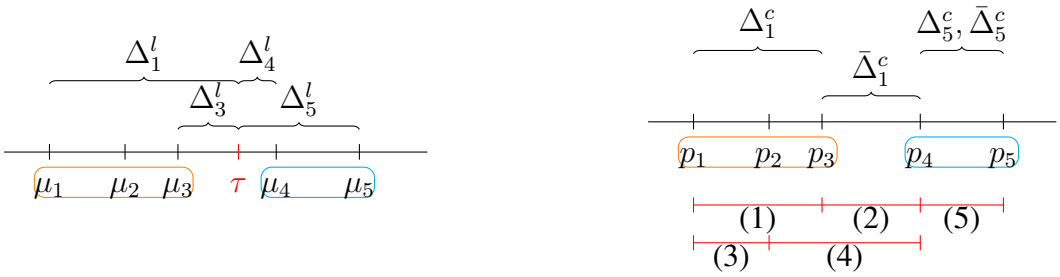


Figure 5.1: Graphical illustration of the quantities Δ_i^l (left) and $\Delta_i^c, \bar{\Delta}_i^c$ (right) for $K = 5$ arms, with $S_\tau = \{4, 5\}$. We have $i_u = 4$ and $i_l = 3$. $\bar{\Delta}_1^c$ is equal to the max of $\min\{(1), (2)\}$ and $\min\{(3), (4)\}$; $\bar{\Delta}_5^c$ is equal to $\min\{(2), (5)\}$.

pulls. In particular, for any arm i let $\Delta_i^l = |\mu_i - \tau|$ be the gap between arm i and threshold. Then the pull complexity is defined as $H_l = \sum_{i=1}^K \frac{1}{(\Delta_i^l)^2}$. Chen et al. [52] shows that there exists an algorithm using at most $O(H_l \log(KH_l/\delta))$ pulls. Moreover, they show a lower bound that any pull-only algorithm would require at least $\Omega(H_l \log(1/\delta))$ pulls to give correct output with probability $1 - \delta$. We add another notation for a "partial" label complexity: let $H_l(m)$ be the sum of the largest m terms in H_l. Namely, we sort $\mu_1, \ldots, \mu_K$ by their gap with threshold, i.e., $\Delta_{i_1}^l \le \Delta_{i_2}^l \le \cdots \le \Delta_{i_K}^l$ (cf. Figure 5.1 left), and $H_l(m) = \sum_{j=1}^m \frac{1}{(\Delta_{i_j}^l)^2}$.

Duel complexity. Now we define the complexity w.r.t. duels. Our goal is to use duels to infer the (positive or negative) label of arms without actually pulling them. Therefore the difficulty of inferring a positive arm $i \in S_\tau$ will depend on its difference with the "worst" positive arm, and similarly $i \in S_\tau^c$ with the "best" negative arm. Let $i_l = \arg\max_{i \in S_\tau^c} p_i$ be the best negative arm and $i_u = \arg\min_{i \in S_\tau} p_i$ be the worst positive arm, where p_i is defined in Equation (5.1). And for any arm $i \in S_\tau$, let $\Delta_i^c = p_i - p_{i_u}$ be the gap with arm i_u and similarly for any arm $j \in S_\tau^c$ define $\Delta_j^c = p_{i_l} - p_j$. Intuitively, the complexity of identifying arm i through duels should depend on Δ_i^c, and we therefore define $H_{c,1} = \sum_{i=1}^K \frac{1}{(\Delta_i^c)^2}$.

Moreover, it is worthwhile noting that the complexity of inferring a positive arm i using arm i_u will not only depend on $p_i - p_{i_u}$, but also on $p_{i_u} - p_{i_l}$. If the gap $p_{i_u} - p_{i_l}$ is very small, we cannot easily differentiate i_u from the other negative arms. On the other hand, we can use any positive arm j to infer about arm i, when $p_{i_u} \le p_j < p_i$. To this end, we define

$$\bar{\Delta}_i^c = \begin{cases} \max\limits_{j \in S_\tau} \min\{p_j - p_{i_l}, p_i - p_j\} & \text{if } i \in S_\tau, \\ \max\limits_{j \in S_\tau^c} \min\{p_j - p_i, p_{i_u} - p_j\}, & \text{if } i \in S_\tau^c, \end{cases}$$

See Figure 5.1 right for a reference. And we define another duel complexity as $H_{c,2} = \sum_{i \in \mathcal{A} \setminus \{i_u, i_l\}} \frac{1}{\left(\bar{\Delta}_i^c\right)^2}$.

Relation between Δ_i^c and $\bar{\Delta}_i^c$. Although we always have $\Delta_i^c \ge \bar{\Delta}_i^c$ and thus $H_{c,1} \le H_{c,2}$, in many situations Δ_i^c and $\bar{\Delta}_i^c$ are of similar scales. To see this, notice that $\bar{\Delta}_i^c \ge \min\{\Delta_i^c, p_{i_u} - p_{i_l}\}$. In many cases in practice, we would expect a gap between S_τ and S_τ^c, and therefore $p_{i_u} - p_{i_l}$ will be a constant. We give a formal proposition about the relation between $H_{c,2}$ and $H_{c,1}$ under Massart noise condition in Section 5.5.

In Section 5.3, we present an upper bound using $H_{c,2}$, and in Section 5.4, we present a lower bound using $H_{c,1}$.

5.3 Algorithm and Analysis

We present our Rank-Search algorithm in this section. We give a detailed description of the algorithm in Section 5.3.1, and analyze its theoretical performance in Section 5.3.2.

5.3.1 Algorithm Description

Algorithm 11 Rank-Search (RS)

Input: Set of arms $\mathcal{A}$, noise tolerance δ, threshold τ

1: Confidence level $\gamma_0 \leftarrow 1/4$, $S \leftarrow \mathcal{A} = [K]$, counter $t \leftarrow 0$
2: For every $i \in S$, let $n_i \leftarrow 0, w_i \leftarrow 0$
3: ▷ n_i: Comparison count, w_i: Win count
4: **while** $S \neq \emptyset$ **do**
5: **while** $\exists i \in S, n_i \leq \frac{1}{\gamma_t^2} \log\left(\frac{8|S|(t+1)^2}{\delta}\right)$ **do**
6: **for** $i \in S$ **do**
7: Draw $i' \in [K]$ uniformly at random, and compare arm i with arm i'
8: If arm i wins, $w_i \leftarrow w_i + 1$
9: $n_i \leftarrow n_i + 1$
10: **end for**
11: **end while**
12: Compute $\hat{p}_i \leftarrow w_i/n_i$ for all $i \in S$
13: Rank arms in S according to their $\hat{p}_i$: $S = (i_1, i_2, ..., i_{|S|}), \hat{p}_{i_1} \leq \hat{p}_{i_2} \leq \cdots \leq \hat{p}_{i_{|S|}}$
14: Get $(k, T) = \text{Binary-Search}(S, \tau, \delta/4(t+1)^2)$
15: If $k < |S|$, let $\overline{S} = \{i \in S : \hat{p}_i - \hat{p}_{i_{k+1}} > 2\gamma_t\}$; for $i \in \overline{S}$, set $\hat{y}_i = 1$
16: If $k > 0$, let $\underline{S} = \{i \in S : \hat{p}_i - \hat{p}_{i_k} < -2\gamma_t\}$; for $i \in \underline{S}$, set $\hat{y}_i = 0$
17: $S \leftarrow S - \overline{S} - \underline{S} - T$
18: $\gamma_{t+1} \leftarrow \gamma_t/2$
19: $t \leftarrow t + 1$
20: **end while**

Output: $\hat{S}_\tau = \{i : \hat{y}_i = 1\}, \hat{S}_\tau^c = \mathcal{A} \setminus \hat{S}_\tau$

Algorithm 11 describes the Rank-Search algorithm. At a high level, RS alternates between ranking items using duels (Line 5-13), and a binary search using pulls (Line 14 and Algorithm 12). We first initialize the work set S with all arms, and comparison confidence $\gamma_0 = 1/4$. In the rank phase, we iteratively compare each arm $i \in S$ with a random arm, as an unbiased estimator for p_i. After each arm has received $\frac{\log(2/\delta_t)}{\gamma_t^2}$ comparisons, we rank the arms in S according to their win rates $\hat{p}_i$. Then Algorithm 12 performs binary search on the sorted S to find the boundary between positive and negative arms (detailed below).

Our binary search is a standard process: it starts with the middle of the sequence, and if the middle arm is positive, we move to the first half (i.e., arms with smaller estimated means), and otherwise, we move to the second half (i.e., arms with larger estimated means). Algorithm 2 just behaves as if S is perfectly ranked. It is worthwhile noting that since S is not ranked according to the real p_i's, there might be negative samples larger than positive samples in S. Nevertheless, the binary search just proceeds as if S is perfectly ranked. We figure out the label of the middle point using Figure-Out-Label (Algorithm 13). Figure-Out-Label aims to solve the simple TBP in the one-arm setting: We keep a confidence interval $\hat{\mu}_i \pm \gamma$ in each round and return the label once τ is not in the interval.

Binary-Search returns the boundary k. Let $\overline{S} = \{i \in S : \hat{p}_i - \hat{p}_{i_{k+1}} > 2\gamma_t\}$ be the arms that are separated from arm i_{k+1}, and we label $i \in \overline{S}$ as positive; we do similarly for negative arms. Then we update working set S with all the unlabeled arms, and shrink confidence γ_t.

Algorithm 12 Binary-Search

Input: Sequence $S = (i_1, i_2, ..., i_{|S|})$, threshold τ, confidence δ_0

$k_{\min} \leftarrow 0, k_{\max} \leftarrow |S|, T = \emptyset$
while $k_{\min} < k_{\max}$ **do**
  $k = \lceil (k_{\min} + k_{\max})/2 \rceil$
  $\hat{y}_{i_k} =$ Figure-Out-Label$(i_k, \tau, \frac{\delta}{\log |S|})$
  $T = T \cup \{i_k\}$
  if $\hat{y}_{i_k} = 1$ **then**
    $k_{\max} = k - 1$
  else
    $k_{\min} = k$
  end if
end while

Output: Boundary $k_{\min}$, labeled arms T

Algorithm 13 Figure-Out-Label

Input: Arm i, threshold τ, confidence δ_1

$t \leftarrow 0$
Define $m_i \leftarrow 0, s_i \leftarrow 0$
repeat
  while $m_i \leq 2^t$ **do**
    Query Y_i, and let $s_i \leftarrow s_i + Y_i, m_i \leftarrow m_i + 1$
  end while
  Compute $\hat{\mu}_i \leftarrow s_i/m_i$
  $\gamma = R\sqrt{\frac{2\log(4(t+1)^2/\delta_1)}{m_i}}$
  $t \leftarrow t + 1$
until $|\hat{\mu}_i - \tau| > \gamma$

Output: Predicted label $\hat{y}_i = I(\hat{\mu}_i > \tau)$

5.3.2 Theoretical Analysis

We now present the theorem about performance of RS.

Theorem 58. *Let* $\gamma^* = \min_{i\in\mathcal{A}\setminus\{i_u,i_l\}} \bar{\Delta}_i^c$ *and* $\Delta^* = \min_i \Delta_i^l$. *Then with probability* $1-\delta$ *RS succeeds, and the number of duels and pulls it uses are bounded by*

$$n_{duel} \leq 32H_{c,2}\log\frac{4K\log(1/\gamma^*)}{\delta},$$
$$n_{pull} \leq 16R^2H_l(n_l)\log\left(\frac{n_l\log(1/\Delta^*)}{\delta}\right),$$

where n_l *is the number of times Figure-Out-Label is called, and we have* $n_l = O(\log K\log(1/\gamma^*))$.

Remark. First, the results in [52] correspond to using $O\left(H_l(K)\log\left(\frac{H_l(K)K}{\delta}\right)\right)$ pulls to get δ confidence. In terms of number of direct pulls, RS can reduce K dependence to $\log K$ dependence when γ^* is a constant, an exponential improvement.
Second, in terms of number of duels, RS has a requirement based on dueling complexity $H_{c,2}$ instead of H_l. In many cases, $H_{c,2}$ is close to H_l, and we point out several such cases in Section 5.5. Thus, we see that in the Dueling-choice framework, the number of pulls required improves exponentially in dependence on K at the expense of requiring a number of duels proportional to number of pulls in pull-only case. This is similar to the findings in [105, 185, 187] for regression and feature-based classification in the Dueling choice setting.

5.4 Lower Bounds

In this section, we give lower bounds that complement our upper bounds. We first give an arm-wise lower bound in Section 5.4.1 to show that RS is almost optimal in terms of the total number of queries to each individual arm. Then, we discuss the optimality of both n_{duel} and n_{pull} in Section 5.4.2.

For simplicity, in this section we suppose all rewards follow a Gaussian distribution with parameter R, i.e., $\nu_i = \mathcal{N}(\mu_i, R^2)$. Our results can be easily extended to other sub-Gaussian distributions (e.g., when all rewards are binary).

5.4.1 An Arm-Wise Lower Bound

The following theorem gives a lower bound on the number of pulls and duels on a particular arm k.

Theorem 59. *Suppose an algorithm* $\mathcal{A}$ *recovers* S_τ *with probability* $1-\delta$ *for any problem instance* $(M,\boldsymbol{\mu})$ *and* $\delta \leq 0.15$. *For any arm* i, *let* $D_i^{\mathcal{A}}$ *be the number of times that arm* i *is selected for a duel, and* $L_i^{\mathcal{A}}$ *be the number of times that arm* i *is pulled. Let* $c = \min\{\frac{1}{10}, \frac{R^2}{2}\}$. *Then for any problem instance* $(M,\boldsymbol{\mu})$ *with* $M_{ij} \geq \frac{3}{8}$ *for every arm* $i,j \in [K]$, *and a specific arm* $k \in S_\tau$, *we have*

$$\mathbb{E}_{M,\boldsymbol{\mu}}[(\Delta_k^c)^2 D_k^{\mathcal{A}} + (\Delta_k^l)^2 L_k^{\mathcal{A}}] \geq c\log(\frac{1}{2\delta}). \tag{5.2}$$

Theorem 59 shows an arm-wise lower bound that the sum of duels and pulls (weighted by their complexity) must satisfy (5.2). In the pull-only setting, this agrees with the known result that number of pulls needed for an arm k is $\Omega((\Delta_k^l)^{-2})$. And for duel-choice setting, it shows that if we never pull arm k, number of duels involving arm k (against some known arm) is at least $\Omega((\Delta_k^c)^{-2})$. From our proof of Theorem 58, we can easily show the following proposition for the upper bound that RS achieves:

Proposition 60. *For any problem instance $(M, \boldsymbol{\mu})$ and arm k, Algorithm RS succeeds with probability at least $1-\delta$ and there exists a constant C such that the RS algorithm achieves that*

$$\mathbb{E}_{M,\boldsymbol{\mu}}[(\bar{\Delta}_k^c)^2 D_k^{RS} + (\Delta_k^l)^2 L_k^{RS}] \leq C \log\left(\frac{K \log(\frac{K}{\gamma^* \Delta^*})}{\delta}\right). \tag{5.3}$$

Comparing (5.3) with (5.2), our RS algorithm is arm-wise optimal except for the difference of Δ_k^c vs. $\bar{\Delta}_k^c$, and the log factors. This shows that RS is near optimal in the sum $\mathbb{E}_{M,\boldsymbol{\mu}}[(\Delta_k^c)^2 D_k^{\mathcal{A}} + (\Delta_k^l)^2 L_k^{\mathcal{A}}]$.

5.4.2 Optimality of n_{duel} and n_{pull}

In this subsection, we analyze the lower bound of TBP-DC under the case when duels are much cheaper than pulls. In this case, we would like to minimize the number of pulls, and then minimize the number of duels. Intuitively, RS algorithm is optimal in n_{pull} as it uses roughly $O(\log K)$ pulls; this is necessary even if we know a perfect ranking of all arms (due to the complexity of binary search). We consider an extreme case, where we know the label of arm i_u and i_l from pulls, and wish to infer all other labels using duels. The following corollary of Theorem 59 shows a lower bound in this case:

Corollary 61. *Suppose an algorithm $\mathcal{A}$ is given that $i_u \in S_\tau$ and $i_l \in S_\tau^c$, and uses only duels. Under the same assumption as in Theorem 59, the number of duels of $\mathcal{A}$ is at least $\mathbb{E}[n_{duel}^{\mathcal{A}}] \geq c H_{c,1} \log(1/2\delta)$.*

Combining Corollary 61 with the fact that $O(\log K)$ is necessary for TBP-DC, we show that RS is near optimal in both n_{duel} and n_{pull}.

5.5 Implications of Bounds in Special Cases

We provide two examples to compare our theoretical bounds with the classical pull-only TBP. Throughout this section, we will assume that all the observations follow Bernoulli distributions, and $\tau = 1/2$. The examples we raise in this section all follow the Massart noise condition, i.e., $|\mu_i - \tau| \geq c$ that is well known in classification analysis [125]. We first give the following proposition to show that RS is optimal under Massart noise.

Proposition 62. *Suppose $\Delta_i^l \geq c$ for some c for all arm i, and $M_{ij} = \frac{1}{2} + \sigma(\mu_i - \mu_j)$ for some increasing link function $\sigma : \mathbb{R} \to [-1/2, 1/2]$. Also assume for any $x, y \in [\mu_1, \mu_K]$ we have $\sigma(x-y) \geq L(x-y)$ for some constant L. Then we have i) $p_{i_u} - p_{i_l} \geq 2Lc$, ii) $\bar{\Delta}_i^c \geq \min\{2Lc, \Delta_i^c\}$, and iii) $H_{c,2} \leq \frac{1}{4L^2c^2} H_{c,1}$.*

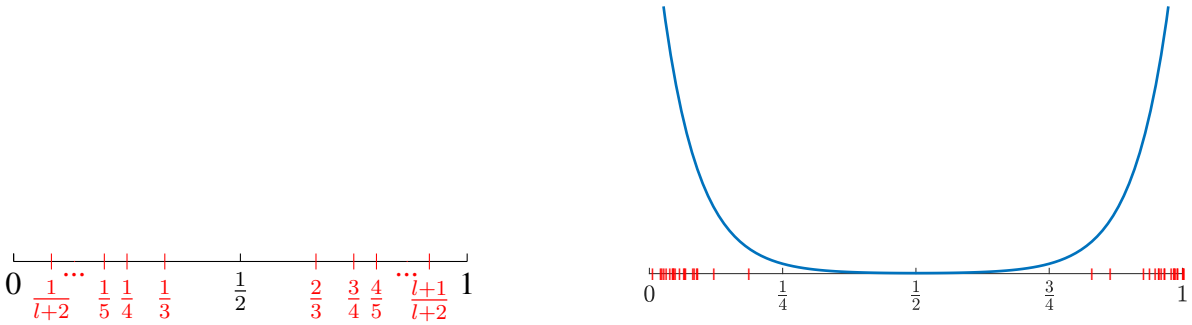


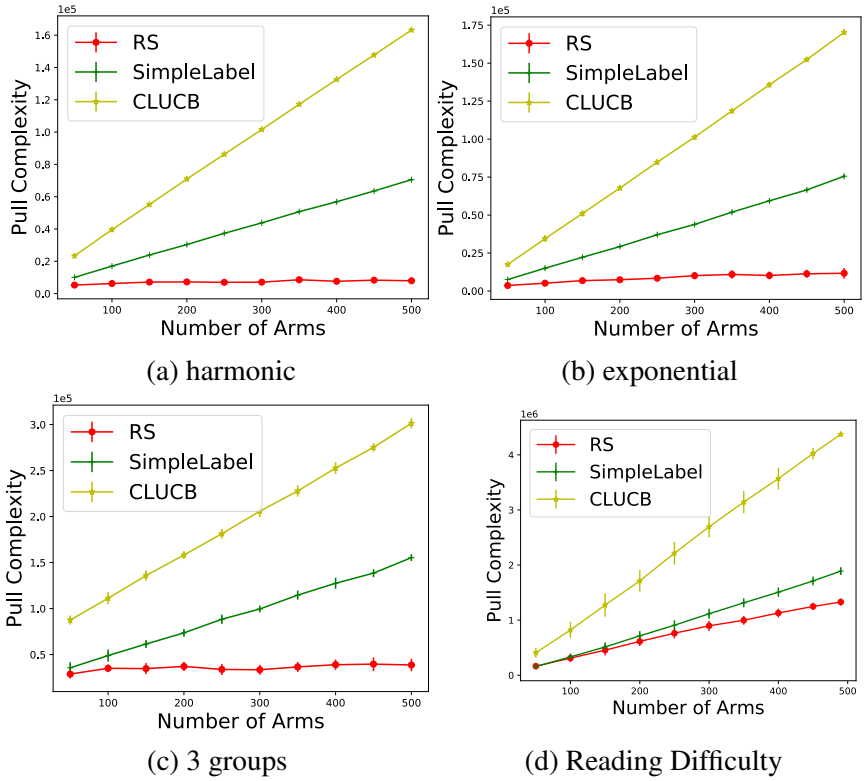
Figure 5.2: Graphical illustration of the examples. Each red vertical line corresponds to one arm i, and $\tau = 1/2$. Left: Example 1 with fixed means. Right: Example 2 with $K = 40$ arms. The blue curve illustrates the pdf of all arm means.

Proposition 62 shows that $H_{c,2} = O(H_{c,1})$ under Massart noise and the assumption that a link function exists. The assumption of such a link function is satisfied by several popular comparison models including the Bradley-Terry-Luce (BTL) [38] and Thurstone models [162]. We now give two positive examples that RS will lead to a gain compared with the pull-only setting. For simplicity we will suppose duels follow a comparison model given as follows: $M_{ij} = \Pr[i \succ j] = \frac{1+\mu_i-\mu_j}{2}$. This is known as the linear link function since it linearly relates the duel win probability with the reward means. Routine calculations show that under a linear link function we have $p_i - p_j = \Theta(\mu_i - \mu_j)$. We require that both our method and pull-only method succeed with probability $1 - \delta$, with a small constant δ (e.g., $\delta = 0.05$). Both of our positive examples assume that the means are dense near the boundaries given by $\mu = 0$ and $\mu = 1$, while a very small fraction of arms have means near 1/2, so that there is a significant gap between the arms i_u and i_l closest to the threshold, as well as any arm i and arm i_u or i_l that is closest to it(cf. Figure 5.2). Although these examples can look artificial at first sight, we note that such a bowl-shaped distribution is common in practice, and is similar to Tsybakov noise [165] assumption used to characterize classification noise in the machine learning literature.

Example 1. Suppose $K = 2l$, and $\mu_i = \frac{1}{(l+3)-i}$ for $1 \le i \le l$, and $\mu_i = 1 - \frac{1}{i-(l-2)}$ for $l + 1 \le i \le 2l$ (see Figure 5.2 left). We will have $\Delta_i^l = \bar{\Delta}_i^c = \Omega(1)$ for all arms $i \in \mathcal{A}$. Then the previous state-of-art CLUCB algorithm requires $O(K \log K)$ pulls, and their lower bounds show that any pull-only algorithm needs at least $\Omega(K)$ pulls. On the other hand, our algorithm requires $O(K \log K)$ duels and only $O(\log^2 K)$ pulls. When pulls are more expensive than duels, there is a significant cost saving when using our RS algorithm.

Example 2. Suppose $K = 2l$. Sample $x_1, ..., x_K$ from an exponential distribution with parameter $\lambda = 4 \log(4l/\delta)$, and let $\mu_i = x_i$ for $1 \le i \le l$, and $\mu_i = 1 - x_i$ for $l + 1 \le i \le 2l$ (see Figure 5.2 right). Then with probability $1 - \delta$: i) $\mu_i \in [0, 1]\ \forall i \in [K]$; ii) $\Delta_i^l = \Omega(1)$, and $H_{c,2} = H_{c,1}$; iii) CLUCB requires $O(K \log K)$ pulls, and any pull-only algorithm requires at least $\Omega(K)$ pulls; iv) Our algorithm requires $O(K \log^3 K)$ duels and $O(\log^2 K)$ pulls.

We provide proofs of the results for these two examples in the appendix.



(a) harmonic (b) exponential

(c) 3 groups (d) Reading Difficulty

Figure 5.3: Empirical results comparing RS and other baselines. Error bars represent standard deviation across 20 experiments.

5.6 Experiments

To verify our theoretical insights, we perform experiments on a series of settings to illustrate the efficacy of RS, on both synthetic and real-world data. For comparison, we include the state-of-art CLUCB in the pull-only setting, and several naive baselines.

5.6.1 Setup and Baselines

We run RS and the other baselines with $\delta = 0.95$, $\tau = 0.5$ and K varying from 50 to 500. Also let $K = 2l$. The duels follows from a linear link function $\Pr[i \succ j] = \frac{1+\mu_i-\mu_j}{2}$, and the mean rewards are given as below:

Experiment 1 (harmonic): This is Example 1 from Section 5.5.

Experiment 2 (exponential): This is Example 2 from Section 5.5.

Experiment 3 (3groups): This is similar to the example in [120]. Let $\mu_i = 0.1$ for $i = 1, 2, ..., l-2$, $\mu_{(l-2):(l+2)} = (0.35, 0.45, 0.55, 0.65)$, and $\mu_i = 0.9$ for $i = l+3, ..., K$.

Experiment 4 (Reading Difficulty): We use the real-world document reading level data from [53]. The data consists of 491 passages, each with a reading difficulty level ranged in 1-12. We randomly take K passages from the whole set, and let $\mu_i = l_i/13$, where l_i is the difficulty level of passage i. The goal here is to identify the difficult passages with level at least 7.

We compare to the following baseline methods:
CLUCB[52]: We implement the CLUCB algorithm which only queries for selective direct pulls in a TBP setting.
SimpleLabel: This is a simple pull-only baseline where we apply Figure-Out-Label to all the arms $i \in \mathcal{A}$ with confidence δ/K.
RankThenSearch: A naive baseline for TBP-DC that first uses a ranking algorithm to rank all the arms, and then performs a binary search to find the boundary. An obvious drawback of this method is that it has to recover the ordering for every pair of arms $(i, j) \in \mathcal{A}$, therefore leading to an enormous number of duels. For the ranking algorithm we use ActiveRank from [90], since it has similar assumptions as ours. Then, we run a single binary search on the sorted sequence, using Figure-Out-Label to identify labels.

We note that previous works on TBP in the fixed budget setting [120, 131] cannot be implemented in our fixed-confidence setting.

For complexity notion, since there is no pre-defined cost ratio between duels and pulls, we compare the pull and duel complexity of RS separately with the baselines. Specifically, we compare pull complexity with SimpleLabel and CLUCB[1], and compare duel complexity with RankThenSearch (since the other two baselines are pull-only algorithms).

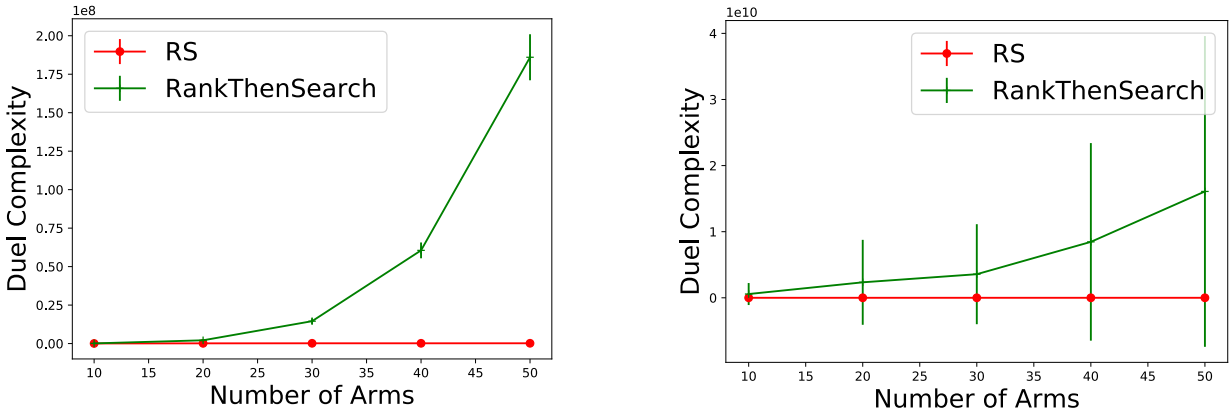
5.6.2 Experiment Results

Comparing Pull Complexity. In Figure 5.3, we plot the empirical pull complexity of RS along with the baselines of CLUCB and SimpleLabel. As expected, the number of pulls of RS is much lower than the baseline algorithms in all three experiments we consider. Interestingly, SimpleLabel also has an advantage over CLUCB in the pull-only setting. We note that CLUCB's $O(H_l \log(\frac{H_l}{\delta}))$ is only optimal up to $\log(H_l)$ factors, and SimpleLabel might have an advantage because its pull complexity is $O(H_l \log(\frac{K \log \Delta^*}{\delta}))$ in the pull-only setting, slightly better than CLUCB. This advantage and the optimal rate for the pull-only setting is of independent interest and we leave it as future work.
Comparing Duel Complexity. We compare duel complexity between RS and RankThenSearch in Figure 5.4. Note that since in **3groups** the arms are not separable, we only compare to RankThenSearch in the first two settings. Our results show that ActiveRank acquires an incredible number of duels in order to rank the arms: to rank 50 arms it acquires more than one billion (1×10^9) duels. This prohibitive cost makes it impossible to adopt the rank and then search method.

5.7 Conclusion

We formulate a new setting of the Thresholding Bandit Problem with Dueling Choices, and provide the RS algorithm, along with upper and lower bounds on its performance. There are several possible directions for future work. First, it is of interest to tighten the bounds so that

[1] Although RankThenSearch does achieve a very low pull complexity, we show in the appendix that it is impractical in terms of duel complexity.



(a) harmonic

(b) exponential

Figure 5.4: Empirical results comparing RS and RankThenSearch. Error bars represent standard deviation across 20 experiments.

they can match in the notion of duel complexity that shows up in the lower and upper bounds. We believe it should be possible to improve the lower bound by randomizing the arms closest to the threshold. Second, we mention that the proposed algorithm RS is adaptive in the setting when comparisons are not helpful, in that it achieves the same pull complexity as SimpleLabel however it requires $2^{K/\log K}$ duels. It is of interest to investigate whether adaptive algorithms can be designed with lower duel complexity. Third, we will explore the performance of the proposed algorithms in real-world applications (e.g. performance on crowdsourced classification in Amazon MTurk Platform) taking into account the relative costs of pulls vs. duels. Finally, it is also of interest to investigate the advantage SimpleLabel might have over CLUCB in pull-only settings.

5.8 Proofs

5.8.1 Proof of Theorem 58

First we show that with high probability our confidence interval in Algorithm 11 and 13 bounds p_i and μ_i.

Lemma 63. *With probability $1-\delta$ the following holds:*

- *At step 13 in Algorithm 11 we have $|p_i - \hat{p}_i| \leq \gamma_t$ for all $i \in S$ and all t;*
- *At step 8 in Algorithm 13 we have $|\mu_i - \hat{\mu}_i| \leq \gamma$ for all arms i that are passed to Algorithm 13.*

Proof. The lemma follows from standard concentration inequality. Using Hoeffding's inequality and a union bound we know that in each round of Algorithm 11 we have

$$\Pr[\exists i, |p_i - \hat{p}_i| > \gamma_t] \leq |S| \exp(-2n_i \cdot \gamma_t^2) \leq \frac{\delta}{4t^2}.$$

Sum it up we have $|p_i - \hat{p}_i| \leq \gamma_t$ holds for all $i \in S$ and all rounds t with probability at most $\delta/2$.

Similarly, from Hoeffding's inequality for sub-Gaussian random variables and a union bound we have for any run of Figure-Out-Label,

$$\begin{aligned}\Pr[\exists t, |\mu_i - \hat{\mu}_i| > \gamma] &\leq \sum_{t=0}^{\infty} \exp(-\frac{\gamma^2}{2R^2}) \\ &\leq \sum_{t=0}^{\infty} \frac{\delta_1}{4(t+1)^2} \leq \delta_1.\end{aligned}$$

Now sum the probability over all runs of Figure-Out-Label we have

$$\Pr[\text{Every Figure-Out-Label is correct}] = \sum_{t=0}^{\infty} \frac{\delta}{4(t+1)^2} \log|S| \cdot \frac{1}{\log|S|} <= \delta/2.$$

The lemma follows from another union bound. □

We now assume the event in Lemma 63 happens. Now we can show that we never make a mistake when we estimate labels in Algorithm 13 using direct pulls. Firstly, upon termination of Figure-Out-Label we have $|\hat{\mu}_i - \tau| > \gamma$. Not losing generality, suppose we have $\hat{y}_i = 1$ as the output. Then we have $\hat{\mu}_i - \tau > \gamma$, and thus $\mu_i > \tau$, so $i \in S_\tau$. Similarly we do not make a mistake when $\hat{y}_i = 0$.

To show the correctness when we infer labels in step 15 and 16 in Algorithm 11, we first need the following lemma for binary search in an arbitrary noisy sequence:

Lemma 64. *Binary-Search always returns within $\lceil \log(|S|) + 1 \rceil$ iterations, and the first output k satisfies i) $\hat{y}_{i_{k+1}} = 1$ if $k < |S|$; and ii) $\hat{y}_{i_k} = 0$ if $k > 0$.*

Proof. Firstly, Algorithm 12 always terminates, because $k = \lceil (k_{\min} + k_{\max})/2 \rceil$ satisfies $k_{\max} - k_{\min} \geq 2\max\{k - k_{\min}, k_{\max} - k\}$. For simplicity, define imaginary labels $\hat{y}_0 = 0, \hat{y}_{|S|+1} = 1$. We prove by induction that we always have $\hat{y}_{i_{k_{\min}}} = 0$ and $\hat{y}_{i_{k_{\max}+1}} = 1$. This is true for the first iteration; for subsequent iterations, if we move to the left (Line 7) we have $\hat{y}_{i_k} = \hat{y}_{i_{k_{\max}+1}} = 1$; if we move the right (Line 9) we have $\hat{y}_{i_k} = \hat{y}_{i_{k_{\min}}} = 0$. Therefore the claim holds. Note that upon termination we must have $k_{\max} = k_{\min}$. The lemma then follows from the claim. □

Now if we let $\hat{y}_i = 1$ in step 15 in Algorithm 11, we have $\hat{p}_i - \hat{p}_{i_{k+1}} \geq 2\gamma_t$, and therefore $p_i > p_{i_{k+1}}$. Since, we have $\hat{y}_{i_{k+1}} = 1$ from Lemma 64 and its label is estimated correctly by Algorithm 13, $y_{i_{k+1}} = 1$ and thus $i_{k+1} \in S_\tau$. Since $i_{k+1} \in S_\tau$, $p_{i_{k+1}} \geq p_j$ for all $j \in S_\tau^c$ and same holds for $p_i > p_{i_{k+1}}$ meaning $i \in S_\tau$. Similarly we do not make a mistake on step 16.

Now we consider the number of duels taken to infer when any arm $i = \mathcal{A} \setminus \{i_u, i_l\}$ is in $\overline{S}$ or $\underline{S}$ and hence is eliminated from further duels. Not losing generality, suppose $i \in S_\tau$, and thus $\mu_i > \tau$. We show that the arm i is eliminated from further duels when we have $4\gamma_t < \bar{\Delta}_i^c$. Suppose we have $i \notin \overline{S}$ i.e. $\hat{p}_{i_{k+1}} \geq \hat{p}_i - 2\gamma_t$ at the end of the binary search in round t. Let $j = \arg\max_{j \in S_\tau} \min\{p_j - p_{i_l}, p_i - p_j\}$ be the maximizer to obtain $\bar{\Delta}_i^c$.

By Lemma 63 and definition of $\bar{\Delta}_i^c$ we have

$$\hat{p}_j \leq p_j + \gamma_t \leq p_i - \bar{\Delta}_i^c + \gamma_t < p_i - 3\gamma_t \leq \hat{p}_i - 2\gamma_t,$$

so $\hat{p}_j < \hat{p}_i - 2\gamma_t \leq \hat{p}_{i_{k+1}}$. So arm j is ranked before arm i_{k+1}; and since $\hat{y}_{i_k} = 0$ by Lemma 64, we have $i_k \notin S_\tau$ since its label is estimated correctly by Algorithm 13, and therefore arm j is ranked no later than arm i_k, thus $\hat{p}_j \leq \hat{p}_{i_k}$. However, from definitions of $\bar{\Delta}_i^c$ and arm i_l, we have

$$p_j \geq p_{i_l} + \bar{\Delta}_i^c \geq p_{i_k} + 4\gamma_t.$$

And therefore by Lemma 63 we have $\hat{p}_j \geq p_j - \gamma_t \geq p_{i_k} + 3\gamma_t \geq \hat{p}_{i_k} + 2\gamma_t$, which makes a contradiction. Therefore we will have $\hat{p}_{i_{k+1}} < \hat{p}_i - 2\gamma_t$ i.e. arm $i \in \overline{S}$, and arm i will be excluded from S in iteration t. In a similar way we can argue that for $i \in S_\tau^c$, it is excluded from S when $\bar{\Delta}_i^c > 4\gamma_t$.

Therefore we would need $\frac{\log(8|S|t^2/\delta)}{(\bar{\Delta}_i^c/4)^2}$ duels to eliminate arm i from further duels. Sum this over all arms i and use the fact that $t \leq \log(1\gamma^*)$, we get the number of duels is $O(H_{c,2}\log(\frac{K\log(1/\gamma^*)}{\delta}))$ to identify all arms except $\{i_l, i_u\}$. When every arm $i \in \mathcal{A} \setminus \{i_u, i_l\}$ has been given a label, i_u, i_l will be given a label during binary search.

Now we bound the number of direct pulls. We figure out the label of arm i when $2\gamma \geq |\mu_i - \tau|$ in Algorithm 13. Therefore for each sample we need $2\sqrt{R^2\frac{2\log(2/\delta_0)}{T}} \leq |\mu_i - \tau|$ pulls; note that we only require pulls during binary search. Each binary search runs Algorithm 13 for at most $\log K$ times, and we do $\log(1/\gamma^*)$ times of binary search. Combining these terms we get the number of pulls.

5.8.2 Proof of Theorem 59

Our include a proof that follows a very similar process as [90], but adapting to the case where both duels and pulls are available.

Not losing generality, suppose $k \in S_\tau$; the proof for $k \in S_\tau^c$ is similar. We first use a lemma from bandit literature [106] that links KL divergence with error probability. Let $\nu = \{\nu_j\}_{j=1}^m$ be a collection of m probability distributions supported on $\mathbb{R}$. Consider an algorithm $\mathcal{A}$ that selects an index $i_t \in [m]$ and receives an independent draw X from ν_i. i_t only depends on its past observations; i.e., i_t is $\mathcal{F}_{t-1}$ measurable, where $\mathcal{F}_t$ is the σ-algebra generated by $i_1, X_1, ..., i_t, X_t$. Let χ be a stopping rule of $\mathcal{A}$ that determines the termination of $\mathcal{A}$. We assume that χ is measurable w.r.t $\mathcal{F}_t$ and $\Pr[\chi < \infty] = 1$. Let $Q_i(\chi)$ be the number of times that ν_i is selected by $\mathcal{A}$ until termination. For any $p, q \in (0,1)$, let $d(p,q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$ be the KL divergence between two Bernoulli distributions with parameter p, q. We use the following lemma:

Lemma 65 ([106], Lemma 1)**.** *Let $\nu = \{\nu_j\}_{j=1}^m, \nu' = \{\nu_j'\}_{j=1}^m$ be two collections of m probability distributions on $\mathbb{R}$. For any event $\mathcal{E} \in \mathcal{F}_\chi$ with $\Pr_\nu[\mathcal{E}] \in (0,1)$ we have*

$$\sum_{i=1}^m \mathbb{E}_\nu[Q_i(\chi)]KL(\nu_i, \nu_i') \geq d(\Pr_\nu[\mathcal{E}], \Pr_{\nu'}[\mathcal{E}]). \tag{5.4}$$

Now, define the event $\mathcal{E}$ to be the event that $\mathcal{A}$ succeeds under M and $\boldsymbol{\mu}$, i.e., $\mathcal{E} \equiv \left\{S_\tau = \hat{S}_\tau, S_\tau^c = \hat{S}_\tau^c\right\}$. Under the relation $M_{ij} = 1 - M_{ji}$ the comparison is uniquely defined by the probabilities $\{M_{ij}, 1 \leq i < j \leq K\}$; and pull is uniquely defined by the mean vector $\boldsymbol{\mu}$. For any two arms

i, j, let $D_{ij}(\chi)$ be the number of times that arms i and j duel before stopping. Therefore for two problem settings $(M, \boldsymbol{\mu})$ and $(M', \boldsymbol{\mu}')$, by Lemma 65 we have

$$\sum_{i=1}^{K}\sum_{j=i+1}^{K}\mathbb{E}_{M,\boldsymbol{\mu}}[D_{ij}]d(M_{ij}, M'_{ij}) + \frac{1}{2R^2}\sum_{i=1}^{K}(\mu_i - \mu'_i)^2 \geq d(\Pr_{M,\boldsymbol{\mu}}[\mathcal{E}], \Pr_{M',\boldsymbol{\mu}'}[\mathcal{E}]). \tag{5.5}$$

The second term in (5.5) follows from the KL divergence between Gaussian variables. We now construct another feasible profile $(M', \boldsymbol{\mu}')$ and that $\mu_k < \tau$ and that $p'_k < p'_j$ for any $j \in S_\tau$ according to M'. Therefore in this case $k \notin S_\tau(M', \boldsymbol{\mu}')$, where $S_\tau(M', \boldsymbol{\mu}')$ is the set of arms with reward larger than τ under $M', \boldsymbol{\mu}'$. Since $\mathcal{A}$ succeeds with probability $1-\delta$ for any problem setting, we have $\Pr_{M,\boldsymbol{\mu}}[\mathcal{E}] \geq 1-\delta$ and $\Pr_{M',\boldsymbol{\mu}'}[\mathcal{E}] \leq \delta$. Therefore

$$d(\Pr_{M,\boldsymbol{\mu}}[\mathcal{E}], \Pr_{M',\boldsymbol{\mu}'}[\mathcal{E}]) \geq d(\delta, 1-\delta) \geq \log\frac{1}{2\delta},$$

which holds for $\delta \leq 0.15$.

We now specify $M', \boldsymbol{\mu}'$. Let

$$M'_{ij} = \begin{cases} M_{kj} - (p_k - p_{i_u}), & \text{if } i = k, j \neq k, \\ M_{kj} + (p_k - p_{i_u}), & \text{if } j = k, i \neq k, \\ M_{kj} & \text{otherwise.} \end{cases}$$

and $\mu'_k = 2\tau - \mu_k$. It is easy to see that $\mu'_k \leq \tau$, and therefore $k \notin S_\tau(M', \boldsymbol{\mu}')$. We now show that the profile $M', \boldsymbol{\mu}'$ by showing that $p'_k < p'_j$. In the new profile we have

$$p'_k = \frac{1}{K-1}\sum_{j\neq k}M'_{kj} = \frac{1}{K-1}\sum_{j\neq k}(M_{kj} - (p_k - p_{i_u})) = p_k - p_k + p_{i_u} = p_{i_u}.$$

For other arms $i \neq k$ we have

$$p'_i = \frac{1}{K-1}\sum_{j\neq i}M'_{ij} = p_i + \frac{1}{K-1}(p_k - p_{i_u}).$$

And therefore $p'_k = p_{i_u} < p'_i$ for any $i \in S_\tau(M', \boldsymbol{\mu}')$, and therefore $(M', \boldsymbol{\mu}')$ is feasible. Also since $M_{ij} \in [\frac{3}{8}, \frac{5}{8}]$ we have

$$M'_{ij} \leq \frac{5}{8} + (\frac{5}{8} - \frac{3}{8}) = \frac{7}{8},$$

and similarly $M'_{ij} \geq \frac{1}{8}$. So for any $j \neq k$ we have

$$d(M_{kj}, M'_{kj}) \leq \frac{(M_{kj} - M'_{kj})^2}{M'_{kj}(1 - M'_{kj})} = 10(p_k - p_{i_u})^2 \tag{5.6}$$

Now consider the sums on the LHS of (5.5). Note that $M'_{ij} = M_{ij}$ when $i \neq k, j \neq k$; and also $\mu_k - \mu'_k = 2(\mu_k - \tau)$ and $\mu_i - \mu'_i = 0$ for $i \neq k$. Combining (5.5) and the uniform bound in (5.6)

we have

$$\begin{aligned}
&\sum_{i=1}^{K}\sum_{j=i+1}^{K}\mathbb{E}_{M,\boldsymbol{\mu}}[D_{ij}]d(M_{ij},M'_{ij})+\frac{1}{2R^2}\sum_{i=1}^{K}(\mu_i-\mu'_i)^2\\
\leq\ & 10(p_k-p_{i_u})^2\sum_{j\neq k}\mathbb{E}_{M,\boldsymbol{\mu}}[D_{kj}]+\frac{2(\mu_k-\tau)^2}{R^2}\mathbb{E}[L_k]\\
=\ & 10(p_k-p_{i_u})^2\mathbb{E}_{M,\boldsymbol{\mu}}[D_k]+\frac{2(\mu_k-\tau)^2}{R^2}\mathbb{E}[L_k]
\end{aligned}$$

Combining the expectations we get the desired results.

5.8.3 Proof of Corollary 61

The corollary follows directly from Theorem 59: For $k\notin\{i_u,i_l\}$ we have $L_k=0$, and therefore

$$\mathbb{E}_{M,\boldsymbol{\mu}}[D_k^{\mathcal{A}}]\geq\frac{c\log(\frac{1}{2\delta})}{(\Delta_k^c)^2}.$$

Sum this over all arm $k\notin\{i_u,i_l\}$ we get the desired result.

5.8.4 Proof of Proposition 62

Under the link function assumption we have

$$\begin{aligned}
p_{i_u}-p_{i_l}&=\frac{1}{K-1}\sum_{i\neq i_u}\sigma(\mu_{i_u}-\mu_i)-\frac{1}{K-1}\sum_{i\neq i_l}\sigma(\mu_{i_l}-\mu_i)\\
&\geq\frac{1}{K-1}\sum_{i=1}^{K}[\sigma(\mu_{i_u}-\mu_i)-\sigma(\mu_{i_l}-\mu_i)]\\
&\geq\frac{K}{K-1}L(\mu_{i_u}-\mu_{i_l})\geq 2Lc.
\end{aligned}$$

For any $i\in S_\tau$, use $j=i_u$ and we have $\bar{\Delta}_i^c\geq\min\{2Lc,\Delta_i^c\}$, and this holds similarly for $i\in S_\tau^c$. Finally for iii), notice that $2Lc\leq 1$ because otherwise $\sigma(2c)>1$, and $\Delta_i^c\leq 1$. Thus we have $\bar{\Delta}_i^c\geq 2Lc\Delta_i^c$, and it leads to iii).

5.8.5 Proof for Example 1

The results follow easily from Theorem 58: We have $\Delta_i^l=|\mu_i-\tau|\geq\frac{1}{6}$ for every arm i. Under the linear link function, we have $p_i-p_j=\Theta(\mu_i-\mu_j)$, and thus $\bar{\Delta}_i^c=\Omega(1)$ for every arm $i\notin\{l,l+1\}$. Therefore $H_l=O(K)$ and $H_{c,2}=O(K)$, and the results follow.

5.8.6 Proof for Example 2

For each x_i, we have $\Pr[x \leq 1/4] \leq \delta/(4l)$. Using a union bound, we have that with probability $1 - \delta/2$ we have $x_i \leq 1/4 \forall i \in [K]$. Let this event be E_B. So under E_B all sample means are in $[0, 1/4]$ and $[3/4, 1]$, so pull-only algorithm requires $\Omega(K)$ pulls.

On the other hand, let $x_{(l)}$ and $x_{(l-1)}$ be the l-th and $(l-1)$-th order statistic of $x_1, ..., x_l$, i.e., the largest and second largest element of $x_1, ..., x_l$. Then $x_{(l)} - x_{(l-1)}$ is distributed according to a exponential distribution with parameter λ. Routine calculation shows that

$$\Pr[x_{(l)} - x_{(l-1)} \geq \frac{-\log(1-\delta/4)}{\lambda}] \geq 1 - \delta/4.$$

Plug in λ and E_B we have

$$\Pr[x_{(l)} - x_{(l-1)} \geq \frac{-\log(1-\delta/4)}{4\log(4l/\delta)}, E_B] \geq 1 - \delta/2.$$

Under this event and symmetrically for $l + 1 \leq i \leq 2l$, we have $H_{c,2} = O(K \log^2 K)$; thus $n_{\text{duel}} = O(K \log^3 K)$ and $n_{\text{pull}} = O(\log^2 K)$.

Chapter 6

Preference-based Reinforcement Learning with Finite-Time Guarantees

In reinforcement learning (RL), an agent typically interacts with an unknown environment to maximize the cumulative reward. It is often assumed that the agent has access to numerical reward values. However, in practice, reward functions might not be readily available or hard to design, and hand-crafted rewards might lead to undesired behaviors, like reward hacking [9, 35]. On the other hand, preference feedback is often straightforward to specify in many RL applications, especially those involving human evaluations. Such preferences help shape the reward function and avoid unexpected behaviors. Preference-based Reinforcement Learning (PbRL, [183]) is a framework to solve RL using preferences, and has been widely applied in multiple areas including robot teaching [57, 93, 94], game playing [180, 182], and in clinical trials [204].

Despite its wide applicability, the theoretical understanding of PbRL is largely open. To the best of our knowledge, the only prior work with a provable theoretical guarantee is the recent work by Novoseller et al. [133]. They proposed the Double Posterior Sampling (DPS) method, which uses Bayesian linear regression to derive posteriors on reward values and transition distribution. Combining with Thompson sampling, DPS has an asymptotic regret rate sublinear in T (number of time steps). However, this rate is based on the *asymptotic* convergence of the estimates of reward and transition function, whose complexity could be exponential in the time horizon H. Also, the Thompson sampling method in [133] can be very time-consuming, making the algorithm applicable only to MDPs with a few states. To fill this gap, we naturally ask the following question:

Is it possible to derive efficient algorithms for PbRL with finite-time guarantees?

While traditional value-based RL has been studied extensively, including recently [20, 98, 201], PbRL is much harder to solve than value-based RL. Most efficient algorithms for value-based RL utilize the value function and the Bellman update, both of which are unavailable in PbRL: the reward values are hidden and unidentifiable up to shifts in rewards. Even in simple tabular settings, we cannot obtain unbiased estimate of the Q values since any offset in reward function results in the same preferences. Therefore traditional RL algorithms (such as Q learning or value iteration) are generally not applicable to PbRL.

Our Contributions. We give an affirmative answer to our main question above, under general assumptions on the preference distribution.

- We study conditions under which PbRL can recover the optimal policy for an MDP. In

particular, we show that when comparisons between trajectories are noiseless, there exists an MDP such that preferences between trajectories are not transitive; i.e., there is no unique optimal policy (Proposition 66). Hence, we base our method and analysis on a general assumption on preferences between trajectories, which is a generalization of the linear link function assumption in [133].

- We develop provably efficient algorithms to find ε-optimal policies for PbRL, with or without a simulator. Our method is based on a synthetic reward function similar to recent literature on RL with rich observations [67, 129] and reward-free RL [99]. We combine this reward-free exploration and dueling bandit algorithms to perform policy search. To the best of our knowledge, this is the first PbRL algorithm with finite-time theoretical guarantees. Our method is general enough to incorporate many previous value-based RL algorithms and dueling bandit methods as a subroutine.
- We test our algorithm against previous baselines in simulated environments. Our results show that our algorithm can beat previous baselines, while being very simple to implement.

6.1 Related Work

We refer readers to [183] for a complete overview of PbRL. In the PbRL framework, there are several ways that one can obtain preferences: We can obtain preferences on i) trajectories, where the labeler tells which of two trajectories is more rewarding; ii) actions, where for a fixed state s, the labeler tells which action gives a better action-value function; or iii) states, where similarly, the labeler tells which state gives a better value function. In this paper, we mainly study preferences over trajectories, which is also the most prevalent PbRL scenario in literature. Our method is potentially applicable to other forms of preferences as well.

PbRL is relevant to several settings in Multi-Armed Bandits. Dueling bandits [42, 200] is essentially the one-state version of PbRL, and has been extensively studied in the literature [73, 74, 189, 199]. However, PbRL is significantly harder because in PbRL the observation (preference) is based on the *sum* of rewards on a trajectory rather than individual reward values. For the same reason, PbRL is also close to combinatorial bandits with full-bandit feedback [12, 58, 143]. Although lower bounds for these bandit problems extends to PbRL, developing PbRL algorithms is significantly harder since we are not free to choose any combination of state-action pairs.

6.2 Problem Setup

MDP Formulation. Suppose a finite-time Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, H, r, p, s_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, H is the number of steps, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function[1], $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, and s_0 is the starting state. For simplicity we assume $S \geq H$. We consider finite MDPs with $|\mathcal{S}| = S, |\mathcal{A}| = A$. We start an episode from the initial state s_0, and take actions to obtain a trajectory $\tau =$

[1]For simplicity, here we assume the reward function to be deterministic.

$\{(s_h, a_h)\}_{h=0}^{H-1}$, following a policy $\pi : \mathcal{S} \to \mathcal{A}$. We also slightly overload the notation to use $r(\tau) = \sum_{(s,a)\in\tau} r(s,a)$ to denote the total reward of τ. For any policy π, let $\tau_h(\pi, s)$ be a (randomized) trajectory by executing π starting at state s from step h to the end.

We further assume that the state space $\mathcal{S}$ can be partitioned into H disjoint sets $\mathcal{S} = \mathcal{S}_1 \cup \cdots \mathcal{S}_H$, where $\mathcal{S}_h$ denotes the set of possible states at step h. Let $\Pi : \{\pi : \mathcal{S} \to \mathcal{A}\}$ be the set of policies. We use Π^H to denote the set of non-stationary policies; here a policy $\pi = (\pi_1, ..., \pi_H) \in \Pi^H$ executes policy π_h in step h for $h \in [H]$ [2]. Also, let $\pi_{h_1:h_2}$ be the restriction of policy $\pi \in \Pi^H$ to step $h_1, h_1 + 1, ..., h_2$. We use the value function $v_{h_0}^{\pi}(s) = \mathbb{E}[\sum_{h=h_0}^{H-1} r(s_h, \pi(s_h))|\pi, s_{h_0} = s]$ to denote the expected reward of policy π starting at state s in step h_0; for simplicity let $v^{\pi} = v_0^{\pi}$. Let $\pi^* = \arg\max_{\pi\in\Pi^H} v_0^{\pi}(s_0)$ denote the optimal (non-stationary) policy. We assume that $r(s,a) \in [0,1]$ for every $(s,a) \in \mathcal{S} \times \mathcal{A}$, and that $v^*(s) = v^{\pi^*}(s) \in [0,1]$ for every state s.

Preferences on Trajectories. In PbRL, the reward $r(s,a)$ is hidden and is not observable during the learning process, although we define the value function and optimal policy based on r. Instead, the learning agent can query to compare two trajectories τ and τ', and obtain a (randomized) preference $\tau \succ \tau'$ or $\tau' \succ \tau$, based on $r(\tau)$ and $r(\tau')$. We also assume that we can compare partial trajectories; we can also compare two partial trajectories $\tau = \{(s_h, a_h)\}_{h=h_0}^{H}$ and $\tau' = \{(s'_h, a'_h)\}_{h=h_0}^{H}$ for any $h_0 \in [H]$. Let $\phi(\tau, \tau') = \Pr[\tau \succ \tau'] - 1/2$ denote the preference between τ and τ'.

PAC learning and sample complexity. We consider PAC-style algorithms, i.e., an algorithm needs to output a policy $\hat{\pi}$ such that $v^{\hat{\pi}}(s_0) - v^*(s_0) \le \varepsilon$ with probability at least $1 - \delta$. In PbRL, comparisons are collected from human workers and the trajectories are obtained by interacting with the environment. So obtaining comparisons are often more expensive than exploring the state space; we therefore compute separately the sample complexity in terms of the number of total steps and number of comparisons. Formally, let $\text{SC}_s(\varepsilon, \delta)$ be the number of exploration steps needed to obtain an ε-optimal policy with probability $1 - \delta$, and SC_p be the number of preferences (comparisons) needed in this process. We omit ε, δ from the sample complexities when the context is clear.

Dueling Bandit Algorithms. Our proposed algorithms uses a dueling bandit algorithm as a subroutine. To utilize preferences, our algorithms use a PAC dueling bandit algorithm to compare policies starting at the same state. Dueling Bandits [200] has been well studied in the literature. Examples of PAC dueling bandit algorithms include Beat the Mean [199], KNOCKOUT [74], and OPT-Maximize [73]. We formally define a dueling bandit algorithm below.

Definition 2 ((ε,δ)-correct Dueling Bandit Algorithm)**.** *Let $\varepsilon > 0, \delta > 0$. $\mathcal{M}$ is a (ε,δ)-correct PAC dueling bandit algorithm if for any given set of arms $\mathcal{X}$ with $|\mathcal{X}| = K$, i) $\mathcal{M}$ runs for at most $\Psi(\varepsilon,\delta)\varepsilon^{-\alpha}$ steps, where $C(\varepsilon,\delta) = poly(K, \log(1/\varepsilon), \log(1/\delta))$ and $\alpha \ge 1$; ii) in every step, $\mathcal{M}$ proposes two arms a, a' to compare; iii) Upon completion, $\mathcal{M}$ returns an arm $\hat{a}$ such that $\Pr[\hat{a} \succ a] \ge 1/2 - \varepsilon$ for every arm $a \in \mathcal{X}$, with probability at least $1 - \delta$.*

One important feature of existing PAC dueling bandit algorithms is whether they require knowing ε before they start - algorithms like KNOCKOUT and OPT-Maximize [73, 74] cannot start without knowledge of ε; Beat-the-Mean [199] does not need to know ε to start, but can return an ε-optimal arm when given the correct budget. We write $\mathcal{M}(\mathcal{X}, \varepsilon, \delta)$ for an algorithm with access to arm set $\mathcal{X}$, accuracy ε and success probability $1 - \delta$; we write $\mathcal{M}(\mathcal{X}, \delta)$ for a dueling

[2] We follow the standard notation to denote $[H] = \{0, 1, ..., H - 1\}$.

bandit algorithm without using the accuracy ε.

Results in the value-based RL and bandit setting. In traditional RL where we can observe the reward value at every step, the minimax rate is largely still open [97]. The upper bound in e.g., [20] translates to a step complexity of $O\left(\frac{H^3SA}{\varepsilon^2}\right)$ to recover an ε-optimal policy, but due to scaling of the rewards the lower bound [61] translates to $\Omega\left(\frac{HSA}{\varepsilon^2}\right)$. Very recently, Wang et al. [173] show an upper bound of $O\left(\frac{HS^3A^2}{\varepsilon^3}\right)$, showing that the optimal H dependence might be linear. It is straightforward to show that the lower bound in [61] translates to a step complexity of $\Omega\left(\frac{HSA}{\varepsilon^2}\right)$ and a comparison complexity of $\Omega\left(\frac{SA}{\varepsilon^2}\right)$ for PbRL. Lastly, we mention that the lower bounds for combinatorial bandits with full-bandit feedback [58] can transform to a lower bound of the same scale for PbRL.

6.2.1 Preference Probablities

As in ranking and dueling bandits, a major question when using preferences is how to define the winner. One common assumption is the existence of a Condorcet winner: Suppose there exists an item (in our case, a policy) that beats all other items with probability greater than 1/2. However in PbRL, because we compare trajectories, preferences might not reflect the true relation between policies. For example, assume that the comparisons are perfect, i.e., $\tau_1 \succ \tau_2$ if $r(\tau_1) > r(\tau_2)$ and vice versa. Now suppose policy π_1 has a reward of 1 with probability 0.1 and 0 otherwise, and π_2 has a reward of 0.01 all the time. Then a trajectory from π_1 is only preferred to a trajectory from π_2 with probability 0.1 if the comparisons give deterministic results on trajectory rewards. Extending this argument, we can show that non-transitive relations might exist between policies:

Proposition 66. *Slightly overloading the notation, for any $h \in [H]$ and $s_h \in \mathcal{S}_h$, let $\phi_s(\pi_1, \pi_2) = \Pr[\tau_h(\pi_1, s) \succ \tau_h(\pi_2, s)] - 1/2$ denote the preference between policies π_1 and π_2 when starting at s_h in step h. Suppose comparisons are noiseless. There exists a MDP and policies π_0, π_1, π_2 such that for some state $s \in \mathcal{S}$, $\phi_s(\pi_0, \pi_1), \phi_s(\pi_1, \pi_2), \phi_s(\pi_2, \pi_0)$ are all less than 0.*

Proposition 66 shows that making assumptions on the preference distribution on trajectories cannot lead to a unique solution for PbRL. Instead, since our target is an optimal policy, we make assumptions on the preferences between *trajectories*:

Assumption 3. *There exists a universal constant $C_0 > 0$ such that for any policies π_1, π_2 and state $s \in \mathcal{S}$ with $v_{\pi_1}(s) - v_{\pi_2}(s) > 0$, we have $\phi_s(\pi_1, \pi_2) \geq C_0(v_{\pi_1}(s) - v_{\pi_2}(s))$.*

I.e., the preference probabilities depends on the value function difference. Following previous literatures in dueling bandits [74, 199], we also define the following properties on preferences:

Definition 3. *Define the following properties on preferences when the following holds for any h, $s \in \mathcal{S}_h$ and three policies π_1, π_2, π_3 such that $v_h^{\pi_1}(s) > v_h^{\pi_2}(s) > v_h^{\pi_3}(s)$:*
Strong Stochastic Transitivity: $\phi_{s_h}(\pi_1, \pi_3) \geq \max\{\phi_{s_h}(\pi_1, \pi_2), \phi_{s_h}(\pi_2, \pi_3)\}$;
Stochastic Triangle Inequality: $\phi_{s_h}(\pi_1, \pi_3) \leq \phi_{s_h}(\pi_1, \pi_2) + \phi_{s_h}(\pi_2, \pi_3)$.

These properties are not essential for our algorithms, but are required for some dueling bandit algorithms that we use as a subroutine. To see the relation between policy preferences and reward preferences, we show the next proposition on several special cases:

Proposition 67. *If either of the following is true, the preferences satisfy Assumption 3, SST and*

STI:
i) There exists a constant C' such that for every pair of trajectories τ, τ' we have $\phi(\tau, \tau') = C(r(\tau) - r(\tau'))$.
ii) The transitions are deterministic, and $\phi(\tau, \tau') = c$ for $r(\tau) > r(\tau')$ and some $c \in (0, 1/2]$.

The first condition in Proposition 67 is the same as the assumption in [133], so our Assumption 3 is a generalization of theirs. We also note that although we focus on preferences over trajectories, preference between policies, as in Assumption 3, is also used in practice [78].

Algorithm 14 PPS: Preference-based Policy Search

Input: Dueling bandit algorithm $\mathcal{M}$, dueling accuracy ε_1, sampling number N_2, success probability δ

Initialize $\hat{\pi} \in \Pi^H$ randomly
for $h = H-1, ..., 0$ **do**
  for $s_h \in \mathcal{S}_h$ **do**
    for $n = 1, 2, ..., N_2$ **do**
      Start an instance of $\mathcal{M}(\mathcal{A}, \varepsilon_1, \delta/S)$
      Receive query (a, a') from $\mathcal{M}$
      Rollout $a \circ \hat{\pi}_{h+1:H}$ from s_h, get trajectory τ
      Rollout $a' \circ \hat{\pi}_{h+1:H}$ from s_h, get trajectory τ'
      Compare τ, τ' and return the result to $\mathcal{M}$
    end for
    if $\mathcal{M}$ has finished **then**
      Update $\hat{\pi}_h$ to the optimal action according to $\mathcal{M}$
    end if
  end for
end for

Output: Policy $\hat{\pi}$

6.3 PbRL with a Simulator

In this section, we assume access to a simulator (generator) that allows access to any state $s \in \mathcal{S}$, executes an action $a \in \mathcal{A}$ and obtains the next state $s' \sim p(\cdot|s, a)$. We first introduce our algorithm, to then follow with theoretical results. We also show a lower bound that our comparison complexity is almost optimal.

Although value-based RL with a simulator has been well-studied in the literature [17, 18], methods like value iteration cannot easily extend to PbRL, because the reward values are hidden. Instead, we base our method on dynamic programming and policy search [21] - by running a dueling bandit algorithm on each state, one can determine the corresponding optimal action. The resulting algorithm, Preference-based Policy Search (PPS), is presented in Algorithm 14. By inducting from $H-1$ to 0, PPS solves a dueling bandit problem at every state $s_h \in \mathcal{S}_h$, with arm rewards specified by $a \circ \hat{\pi}_{h+1:H}$ for $a \in \mathcal{A}$, where $\hat{\pi}$ is the estimated best policy after step $h+1$,

and $\circ$ stands for concatenation of policies. By allocating an error of $O(\varepsilon/H)$ on every state, we obtain the following guarantee:

Theorem 68. *Suppose the preference distribution satisfies Assumption 3. Let $\varepsilon_1 = \frac{C_0\varepsilon}{H}, N_0 = \Psi(\varepsilon_1, \delta/S)\varepsilon_1^{-\alpha}$, where C_0 is defined in Assumption 3. Algorithm 14 returns an ε-optimal policy with probability $1-\delta$ using $O\left(\frac{H^{\alpha+1}S\Psi(A,\varepsilon/H,\delta/S)}{\varepsilon^\alpha}\right)$ simulator steps and $O\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/S)}{\varepsilon^\alpha}\right)$ comparisons.*

We can plug in existing algorithms to instantiate Theorem 68. For example, under SST, OPT-Maximize [73] achieves the minimax optimal rate for dueling bandits. In particular, it has a comparison complexity of $O\left(\frac{K\log(1/\delta)}{\varepsilon^2}\right)$ for selecting an ε-optimal arm with probability $1-\delta$ among K arms. Plugging in this rate we obtain the following corollary:

Corollary 69. *Suppose the preference distribution satisfies Assumption 3 and SST. Using OPT-Maximize as $\mathcal{M}$, Algorithm 14 returns an ε-optimal policy with probability $1-\delta$ using $O\left(\frac{H^3SA}{\varepsilon^2}\log(\delta/S)\right)$ simulator steps, and $O\left(\frac{H^2SA}{\varepsilon^2}\log(\delta/S)\right)$ comparisons.*

The proof of Theorem 68 follows from using the performance difference lemma, combined with properties of $\mathcal{M}$. Our result is similar to existing results for traditional RL with a simulator: For example, in the case of infinite-horizon MDPs with a decaying factor of γ, [19] shows a minimax rate of $O\left(\frac{SA}{\varepsilon^2(1-\gamma)^3}\right)$ on step complexity. This is the same rate as in Corollary 69 by taking $H = \frac{1}{1-\gamma}$ effective steps.

6.4 Combining Exploration and Policy Search for General PbRL

In this Section, we present our main result for PbRL without a simulator. RL without a simulator is a challenging problem even in the traditional value-based RL setting. In this case, we will have to explore the state space efficiently to find the optimal policy. Existing works in value-based RL typically derive an upper bound on the Q-value function and apply the Bellman equation to improve the policy iteratively [20, 98, 201]. However for PbRL, since the reward values are hidden, we cannot apply traditional value iteration and Q-learning methods. To go around this problem, we use a synthetic reward function to guide the exploration. We present our main algorithm in Section 6.4.1, along with the theoretical analysis. We discuss relations to prior work in Section 6.4.2.

6.4.1 Preferece-based Exploration and Policy Search (PEPS)

We call our algorithm Preference-based Exploration and Policy Search (PEPS), and present it in Algorithm 15. As the name suggests, the algorithm combines exploration and policy search. For every step $h \in [H]$ and $s_h \in \mathcal{S}_h$, we set up an artificial reward function $r_{s_h}(s,a) = 1_{s=s_h}$, i.e., the agent gets a reward of 1 once it gets to s_h, and 0 everywhere else (Step 4). This function is also used in recent reward-free learning literatures [67, 99, 129]. But rather than using the reward function to obtain a policy cover (which is costly in both time and space), we use it to help the

Algorithm 15 PEPS: Preferece-based Exploration and Policy Search

Input: Dueling Bandit algorithm $\mathcal{M}$, quantity N_0, success probability δ

1: Initialize $\hat{\pi} \in \Pi^H$ randomly
2: **for** $h = H, H-1, ..., 1$ **do**
3: **for** $s_h \in \mathcal{S}_h$ **do**
4: Let $r_{s_h}(s,a) = 1_{s=s_h}$ for all $s \in S, a \in A$
5: Start an instance of EULER$(r_{s_h}, N_0, \delta/(4S))$
6: Start an instance of $\mathcal{M}\left(\mathcal{A}, \delta/(4S)\right)$
7: Get next query (a, a') from $\mathcal{M}$
8: $U \leftarrow \emptyset$
9: **for** $n \in [N_0]$ **do**
10: Obtain a policy $\hat{\pi}_n$ from EULER, and execute $\hat{\pi}_n$ until step h
11: Return the trajectory and reward to EULER
12: **if** current state is s_h **then**
13: Let $\bar{\pi} = a \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a' \circ \hat{\pi}_{h+1:H}$
14: Execute $\bar{\pi}$ till step H, obtain trajectory τ
15: $U \leftarrow U \cup \tau$
16: **end if**
17: **if** $|U| = 2$ **then**
18: Compare the two trajectories in U and return to $\mathcal{M}$
19: $(a, a') \leftarrow$ next action from $\mathcal{M}$
20: **end if**
21: If $\mathcal{M}$ has finished, break
22: **end for**
23: **if** $\mathcal{M}$ has finished **then**
24: Update $\hat{\pi}_h(s_h)$ to the optimal action according to $\mathcal{M}$
25: **end if**
26: **end for**
27: **end for**

Output: Policy $\hat{\pi}$

dueling bandit algorithm. We can use arbitrary tabular RL algorithm to optimize r_{s_h}; here we use EULER [201] with a budget of N_0 and success probability $\delta/4S$. The reason that we chose EULER is mainly for its beneficial regret guarantees; but we can also use other algorithms in practice. We also start an instance of $\mathcal{M}$, the dueling bandit algorithm, without setting the target accuracy; one way to achieve this is to use a pre-defined accuracy for $\mathcal{M}$, or use algorithms like Beat-the-Mean (see Theorem 70 and 72 respectively).

Once we get to s_h (Step 12), we execute the queried action a or a' from $\mathcal{M}$, followed by the current best policy $\hat{\pi}$, as in PPS. If we have collected two trajectories, we compare them and feed it back to $\mathcal{M}$. Upon finishing N_0 steps of exploration, we update the current best policy $\hat{\pi}$ according to $\mathcal{M}$. We return $\hat{\pi}$ when we have explored every state $s \in \mathcal{S}$.

Throughout this section, we use $\iota = \log\left(\frac{SAH}{\varepsilon\delta}\right)$ to denote log factors. We present two versions of guarantees with different sample complexity. The first version has a lower comparison

complexity, while the second version has a lower total step complexity. The different guarantee comes from setting a slightly different target for $\mathcal{M}$ when it finishes in Step 21. In the following first version, we set $\mathcal{M}$ to finish when it finds a $O(\varepsilon/H)$-optimal policy.

Theorem 70. *Suppose the preference distribution satisfies Assumption 3. There exists a constant c_0 such that the following holds: Let $N_0 = c_0\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$, recall $\iota = \log\left(\frac{SAH}{\varepsilon\delta}\right)$. By setting the target accuracy as $\frac{C_0\varepsilon}{2H}$ for $\mathcal{M}$, PEPS obtains an ε-optimal policy with probability $1-\delta$ using step complexity*

$$O(HSN_0) = O\left(\frac{H^{\alpha+1}S^2\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^\alpha S\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^\alpha}\right)$.

Since we set the target accuracy before the algorithm starts, any PAC dueling bandit algorithm can be used to instantiate Theorem 70. So similar to Theorem 68, we can plug in OPT-Maximize [73] to obtain the following corollary:

Corollary 71. *Suppose the preference distribution satisfies Assumption 3 and SST. There exists a constant c_0 such that the following holds: Let $N_0 = c_0\left(\frac{SH^2A\log(S/\delta)}{\varepsilon^3} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$. Using OPT-Maximize with accuracy ε/H as $\mathcal{M}$, PEPS obtains an ε-optimal policy with probability $1-\delta$ using step complexity*

$$O(HSN_0) = O\left(\frac{H^3S^2A\log(S/\delta)}{\varepsilon^3} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^2SA\log(S/\delta)}{\varepsilon^2}\right)$.

In the second version, we simply do not set *any* performance target for $\mathcal{M}$; it will explore until we finish all the N_0 episodes. Therefore, we never break in Step 21 and $\mathcal{M}$ is always finished in Step 23. Since different states will be reached for a different number of times, we cannot pre-set a target accuracy for $\mathcal{M}$. Effectively, we explore every state s to the accuracy of $\varepsilon_s = O\left(\frac{\varepsilon}{(\mu(s_h)SH^{\alpha-1})^{1/\alpha}}\right)$ (see proof in appendix for details), where $\mu(s)$ is the maximum probability to reach s using any policy. Although this version leads to a slightly higher comparison complexity, it leads to a better step complexity since all exploration steps are used efficiently. We have the following result:

Theorem 72. *Suppose the preference distribution satisfies Assumption 3. There exists a constant c_0 such that the following holds: Let $N_0 = c_0\left(\frac{H^{\alpha-1}S\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha} + \frac{S^3AH^2\iota^3}{\varepsilon}\right)$. PEPS obtains an ε-optimal policy with probability $1-\delta$ using step complexity*

$$O(HSN_0) = \left(\frac{H^\alpha S^2\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha} + \frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{H^{\alpha-1}S^2\Psi(A,\frac{\varepsilon}{HS},\delta/4S)}{\varepsilon^\alpha}\right)$.

We need to instantiate Theorem 72 with an $\mathcal{M}$ that does not need a pre-set accuracy. Under SST and STI, we can use Beat-the-Mean [199], which returns an ε-optimal arm among K arms

with probability $1-\delta$, if it runs for $\Omega\left(\left(\frac{K}{\varepsilon^2}\log\left(\frac{K}{\varepsilon\delta}\right)\right)\right)$ steps. We therefore obtain the following corollary:

Corollary 73. *Suppose the preference distribution satisfies Assumption 3, SST and STI. There exists a constant c_0 such that the following holds: Let $N_0 = c_0\left(\frac{HSA\iota}{\varepsilon^2}+\frac{S^3AH^2\iota^3}{\varepsilon}\right)$. Using Beat-the-Mean as $\mathcal{M}$, PEPS obtains an ε-optimal policy with probability $1-\delta$ using step complexity*

$$O(HSN_0) = O\left(\frac{H^2S^2A\iota}{\varepsilon^2}+\frac{S^4AH^3\iota^3}{\varepsilon}\right)$$

and comparison complexity $O\left(\frac{HS^2A\iota}{\varepsilon^2}\right)$.

6.4.2 Discussion

Comparing Corollary 71 and 73. Considering only the leading terms, the step complexity in Corollary 73 is better by a factor[3] of $\tilde{O}(H/\varepsilon)$ but the comparison complexity is worse by a factor of $\tilde{O}(S/H)$ (recall that we assume $S > H$). Therefore the two theorems depict a tradeoff between the step complexity and comparison complexity.

Comparing with lower bounds and value-based RL. Corollary 71 has the same comparison complexity as Corollary 69. The lower bound for comparison complexity is $\tilde{O}(\frac{SA}{\varepsilon^2})$, a $\tilde{O}(H^2)$ factor from our result. Our comparison complexity is also the same as the current best upper bound in value-based RL (in terms of number of episodes), which shows that our result is close to optimal.

Compared with the $\tilde{O}(\frac{HSA}{\varepsilon^2})$ lower bound on step complexity, Corollary 73 has an additional factor of $\tilde{O}(HS)$. The current best upper bound in value-based RL is $\tilde{O}(\frac{H^3SA}{\varepsilon^2})$, which is $O(H/S)$ times our result (recall that we assume $H < S$).

Comparing with previous RL method using policy covers. Some literature on reward-free learning and policy covers [67, 99, 129] use a similar form of synthetic reward function as ours. [67, 129] considers computing a policy cover by exploring the state space using a similar synthetic function as PEPS. However, our results are generally not comparable since they assume that $\mu(s) \geq \eta$ for some $\eta > 0$, and their result depends on η. Our result do not need to depend on this η. Closest to our method is the recent work of [99], which considers reward-free learning with unknown rewards during exploration. However, their method cannot be applied to PbRL since we do not have the reward values even in the planning phase. We note that the $O(S^2)$ dependence on the step complexity is also common in these prior works. Nevertheless, our rates are better in H dependence because we do not need to compute the policy cover explicitly.

6.4.3 Another Version to Accommodate Arbitrary PAC Dueling Algorithm

A drawback of PEPS is that the version in Theorem 72 requires a dueling algorithm $\mathcal{M}$ that does not need a target accuracy, restricting the possible types of algorithms we can use. In this section, we present a two-phase version of our algorithm to allow arbitrary $\mathcal{M}$, with slightly improved log factors on the guarantee.

[3] We use $\tilde{O}$ to ignore log factors.

The two-phase version is presented in Algorithm 16 as PEPS2. PEPS2 separates the process of obtaining a policy cover and using dueling bandits for policy search; in the first phase (Step 1-9) uses the synthetic reward function to obtain a policy cover with EULER. We then estimate $\mu(s)$ for every state $s \in \mathcal{S}$ by executing the policy that we obtain. Using the estimate $\hat{\mu}_s$, we follow the process in PEPS with the target accuracy specified in Step 12.

For every state $s \in \mathcal{S}$, let $\mu(s) = \max_{\pi \in \Pi} p_\pi(s)$ be the maximum probability to reach s.

Theorem 74. *There exists a constant c_0 such that the following holds. Let $N_0 = c_0 \frac{S^3AH^2\iota^3}{\varepsilon}$, $N_1 = \frac{c_0 S^2 \log(4S/\delta)}{\varepsilon^2}$, $N_2 = \frac{H^{\alpha-1}S\Psi(A, \frac{\varepsilon}{HS}, \delta/4S)}{\varepsilon^\alpha}$. With probability $1-\delta$, Algorithm 16 returns an ε-optimal policy using $HS(N_0+N_1+N_2)$ steps and SN_2 comparisons.*

We can plug in the guarantee of OPT-Maximize instantiate Theorem 74. This result is better in terms of the log terms than the result in Corollary 73.

Corollary 75. *There exists constants c_0 such that the following holds: Let $N_0 = c_0 \left(\frac{HSA\log(\delta/S)}{\varepsilon^2} + \frac{S^3AH^2\iota^3}{\varepsilon} \right)$, where $\iota = \log(\frac{SAH}{\varepsilon\delta})$. Using OPT-Maximize as $\mathcal{M}$, PEPS2 obtains an ε-optimal policy with probability $1-\delta$ using step complexity*

$$O(HSN_0) = O\left(\frac{H^2S^2A\log(\delta/S)}{\varepsilon^2} + \frac{S^4AH^3\iota^3}{\varepsilon} \right)$$

and comparison complexity $O\left(\frac{HS^2A\log(S/\delta)}{\varepsilon^2}\right)$.

6.4.4 Adapting PEPS to the Fixed Budget setting

While we present PPS and PEPS under the fixed confidence setting (with a given ε), one can easily adapt it to the fixed budget setting (with a given N, number of episodes) by dividing N evenly among the states. We present a fixed budget version in this section, described in detail in Algorithm 17. To do this, we set $N_0 = N/S$, where S is the number of non-terminating states. Before the algorithm start, we start an instance of $\mathcal{M}$ for every state $s \in \mathcal{S}$; and instead of only compare when reaching the target state, we simply explore according to $\mathcal{M}$ regardless of the current state at step h.

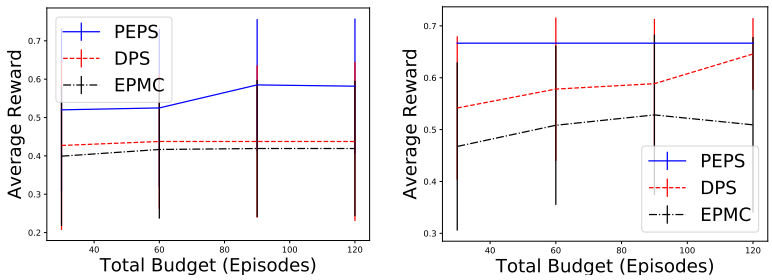
In our experiments We realize PEPS with Q-learning instead of EULER because of its computational efficiency; we use the formulation of [98] with a Hoeffding upper bound on the Q function. For $\mathcal{M}$, we use Beat-the-Mean because it can use any budget.

6.5 Experiments

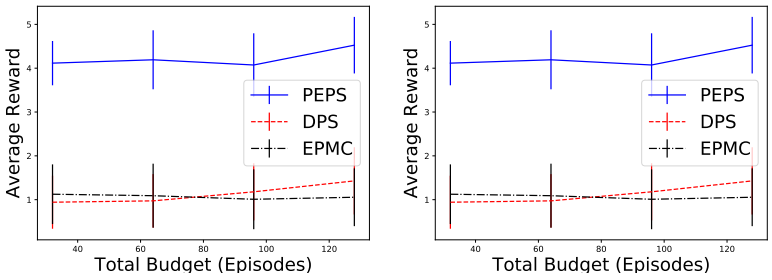
We performed experiments in synthetic environments to compare PEPS with previous baselines. We consider two environments:

Grid World: We implemented a simple Grid World on a 4×4 grid. The agent goes from the upper left corner to the lower right corner and can choose to go right or go down at each step. We randomly put a reward of $1/3$ on three blocks in the grid, and the maximal total reward is $2/3$.

Random MDP: We followed the method in [133] but adapted it to our setting. We consider an MDP with 20 states and 5 steps, with 4 states in each step. The transitions are sampled from a



(a) GridWorld, $c = 1$ (b) GridWorld, $c = 0.001$



(c) Random MDP, $c = 1$ (d) Random MDP, $c = 0.001$

Figure 6.1: Experiment Results comparing PEPS to baselines (DPS & EPMC).

Dirichlet prior (with parameters all set to 0.1) and the rewards are sampled from an exponential prior with scale parameter $\lambda = 5$. The rewards are then shifted and normalized so that the minimum reward is 0 and the mean reward is 1.

Compared methods. We compared to three baselines: i) DPS [133]: We used the version with Gaussian process regression since this version gets the best result in their experiments; ii) EPMC [181], which uses preferences to simulate a Q-function. We followed the default parameter settings for both DPS and EPMC. Details of the algorithms and hyperparameter settings are included in the appendix.

Experiment Setup. Our baselines are not directly comparable to PEPS since their goal is to minimize the regret, instead of getting an ε-optimal policy. However, we can easily adapt all the algorithms (including PEPS) to the fixed budget setting for optimal policy recovery. For the baselines, we ran them until N episodes and then evaluated the current best policy. For PEPS, we used the fixed budget version described in detail in the appendix. For both environments, we varied the budget $N \in [2S', 8S']$, where S' is the number of non-terminating states. The comparisons are generated following the Bradley-Terry-Luce model [38]: $\phi(\tau_1, \tau_2) = \frac{1}{1+\exp(-(r(\tau_1)-r(\tau_2))/c)}$, with c being either 0.001 or 1. In the first setting of c, the preferences are very close to deterministic while comparison between equal rewards is uniformly random; in the latter setting, the preferences are close to linear in the reward difference. We repeated each experiment for 32 times and computed the mean and standard deviation.

Hyperparameters for experiments. For PEPS, we search the hyperparameters for Q learning and Beat-the-Mean. This includes the learning rate of Q-learning (in range $\{0.1, 0.3, 1.0\}$), the ucb bound ratio for Q-learning (in range $\{0.01, 0.1, 1.0\}$), and the ucb bound ratio for Beat-the-Mean (in range $\{0.2, 0.5, 1.0\}$). We also allow the algorithm to random explore with probability in $\{0.05, 0.1, 0.2, 0.5\}$ during Q-learning. For DPS, we follow the default settings to use kernel

variance 0.1 and kernel noise 0.001 for the Gaussian process regression. For EPMC, we use $\alpha = 0.2$ and $\eta = 0.8$.

Results. The results are summarized in Figure 6.1. Overall, PEPS outperforms both baselines in both environments. In Grid World, while all three methods get a relatively high variance when $c = 1$, for $c = 0.001$ PEPS almost always get the exact optimal policy. Also for random MDP, PEPS outperforms both baselines by a large margin (larger than two standard deviations). We note that EPMC learns very slowly and almost does not improve as the budget increases, and this is consistent with the observation in [133]. One potential reason that makes PEPS outperform the baselines is because of the exploration method: Both DPS and EPMC need to estimate the Q function well in order to perform efficient exploration. This estimation can take time exponential in H, and it is not even computationally feasible to test until the Q function converges. As a result, both DPS and EPMC explore almost randomly and cannot recover the optimal policy. On the other hand, our method uses a dueling bandit algorithm to force exploration, so it guarantees that at least states with high reach probability have their optimal action.

6.6 Conclusion

We analyze the theoretical foundations of the PbRL framework in detail and present the first finite-time analysis for general PbRL problems. Based on reasonable assumptions on the preferences, the proposed PEPS method recovers an ε-optimal policy with finite-time guarantees. Experiments show the potential efficacy of our method in practice, and that it can work well in simulated environments. Future work includes testing PEPS in other RL environments and applications, developing algorithms for PbRL with finite-time regret guarantees, as well as PbRL in non-tabular settings.

6.7 Proofs

6.7.1 Proof of Proposition 66

Proof. Let $S = 6$, $\mathcal{S} = \{s_0, ..., s_5\}$, and $A = 3, \mathcal{A} = \{a_0, a_1, a_2\}$, and let $H = 2$. We start at s_0. Let $r(s_0, a) = 0$ for all $a \in \mathcal{A}$. Executing a_0 in s_0 goes to s_1 w.p. 0.2, and to s_2 w.p. 0.8. Executing a_1 in s_0 goes to s_3 with probability 1. Executing a_2 in s_0 goes to s_4 w.p. 0.6 and to s_5 w.p. 0.4. For every action a, let $r(s_1, a) = 1, r(s_2, a) = 0.01, r(s_3, a) = 0.02, r(s_4, a) = 0.5, r(s_5, a) = 0$. See Figure 6.2 for a graphical explanation.

Let $\pi_i(s_0) = a_i$ for $i \in \{0, 1, 2\}$ (actions in other states do not matter). It is easy to verify that $\phi_{s_0}(\pi_1, \pi_2) = -0.3, \phi_{s_0}(\pi_2, \pi_3) = -0.1$, and $\phi_{s_0}(\pi_3, \pi_1) = -0.02$. □

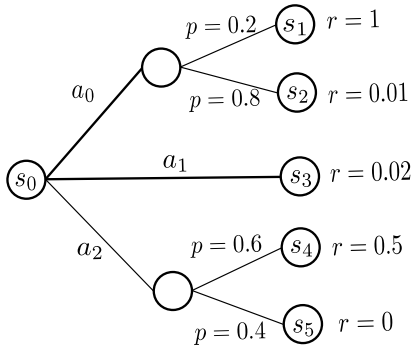


Figure 6.2: Example for proof of Proposition 66.

6.7.2 Proof of Proposition 67

Proof. For i), by the linearity of expectation we have for two random trajectories τ, τ', we have $E[\phi(\tau, \tau')] = E[C(r(\tau) - r(\tau'))]$. Therefore for two policies π_1, π_2

$$\begin{aligned}\phi_s(\pi_1, \pi_2) &= \Pr[\tau(\pi_1, s) \succ \tau(\pi_2, s)] - 1/2 \\ &= E[\phi(\tau(\pi_1, s), \tau(\pi_2, s))] \\ &= E[C(r(\tau(\pi_1, s)) - r(\tau(\pi_2, s)))] = C(v_s(\pi_1) - v_s(\pi_2)).\end{aligned}$$

For ii), when transitions are deterministic each policy corresponds to only one trajectory. Let τ_1 and τ_2 be the two trajectories corresponding to π_1 and π_2 starting at s. Then we have $\phi_s(\pi_1, \pi_2) = \phi(\tau_1, \tau_2)$. Then Assumption 3 is satisfied with $C_0 = c$. □

6.7.3 Proof of Theorem 68

Proof. We only need to show that the output policy $\hat{\pi}$ is ε-optimal. Algorithm 14 loops over $h = 1, 2, ..., H$. At every step h for every state $s \in \mathcal{S}_h$, let $\tilde{\pi}^*(s) = \arg\max_a V(s; a \circ \hat{\pi}_{h+1:H})$. Let P_h^* denote the state distribution of π^* at step h. With probability $1 - \delta$, all instances of $\mathcal{M}$ has finished. Under this event, from the property of $\mathcal{M}$ and the setup of N_0 we have for every state $s \in \mathcal{S}$, $\Pr[\phi_s(\hat{\pi}, \tilde{\pi}^*(s))] \geq 1/2 - \varepsilon_1$. Therefore from Assumption 3 we have

$$|v_{\tilde{\pi}^*(s)}(s) - v_{\hat{\pi}}(s)| \leq \frac{1}{C_0}\phi_s(\tilde{\pi}^*(s), \hat{\pi}) \leq \frac{\varepsilon}{H}.$$

Therefore using the performance difference lemma we have

$$\begin{aligned}V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\ &\leq \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\ &\leq \sum_{h=1}^{H} \frac{\varepsilon}{H} = \varepsilon.\end{aligned}$$

□

6.7.4 Proof of Theorem 70

Proof. Let $\tilde{\mathcal{S}}_h = \{s \in \mathcal{S}_h | \mu(s) \geq \varepsilon/2S\}$ be the set of "good" states that are reachable with probability at least $\varepsilon/(2S)$. Also let $\tilde{\mathcal{S}} = \bigcup_h \tilde{\mathcal{S}}_h$ be the set of all good states.

Now we show that the output policy $\hat{\pi}$ is ε-optimal. We first present the performance of EULER [201]:

Theorem 76 (Theorem 1, [201])**.** *Let Q^* be the value of the optimal policy. For any reward function r, the regret of EULER is at most*

$$R \leq O(\sqrt{Q^* SATL} + \sqrt{S}SAH^2L^3(\sqrt{S} + \sqrt{H}))$$

with probability $1 - \delta$, with $L = \log(HSAT/\delta)$.

For any state s, let N_s be the number of times that we reach s in Step 12. Using Theorem 76 with $T = HN_0$, we have for some constant C,

$$\mu(s)N_0 - N_s \leq C(\sqrt{\mu(s)SAHN_0L} + \sqrt{S}SAH^2L^3(\sqrt{S} + \sqrt{H})) \tag{6.1}$$

with probability $1 - \delta/4S$. By a union bound, suppose (6.1) holds for every state $s \in \mathcal{S}$, which happens with probability $1 - \delta/4$. Now consider any $s \in \tilde{\mathcal{S}}$. With $N_0 = \Omega(\frac{S^3AH^2\iota^3}{\varepsilon})$, we have $N_s \geq 1/2\mu(s)N_0$. Now also using $N_0 = \Omega\left(\frac{SH^\alpha\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^{\alpha+1}}\right)$, we have

$$N_s \geq \frac{1}{2}\mu(s)N_0 \geq \Omega\left(\frac{H^\alpha\Psi(A,\varepsilon/H,\delta/4S)}{\varepsilon^\alpha}\right).$$

By setting the constant in N_0 properly, from the definition of Ψ we know that with probability $1 - \delta/4S$, $\mathcal{M}$ returns a $\frac{C_K\varepsilon}{H}$ optimal arm; here c_K is a constant to be specified later.

Algorithm 15 loops over $h = 1, 2, ..., H$. At every step h for every state $s \in \mathcal{S}_h$, let $\tilde{a}_s^* = \arg\max_a V(s; a \circ \hat{\pi}_{h+1:H})$. From the guarantees of OPT-Maximize we have

$$v(s_h; \tilde{a}_s^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \leq \frac{1}{C_0}\phi_s(\tilde{a}_s^*, \hat{\pi}_{h+1:H}) \leq \frac{\varepsilon}{2H}.$$

The first inequality comes from Assumption 3; we set c_K small enough to satisfy the latter inequality.

Let P_h^* denote the state distribution of π^* at step h. We have

$$\begin{aligned}
V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^{H} E_{s_h \sim P_h^*} \left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \right] \\
&\leq \sum_{h=1}^{H} E_{s_h \sim P_h^*} \left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \right] \\
&\leq \sum_{h=1}^{H} \Pr_{\pi^*}[s_h \in \tilde{\mathcal{S}}] + \Pr[s_h \in \tilde{\mathcal{S}}] E_{s_h \sim P_h^*, s_h \in \tilde{\mathcal{S}}} \left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \right] \\
&\leq \varepsilon/(2S) \cdot S + \sum_{h=1}^{H} \varepsilon/(2H) \leq \varepsilon.
\end{aligned}$$

Here the first equality is the performance difference lemma (Lemma 13, [129]). The first inequality comes from the definition of $\tilde{\pi}^*$; the third inequality comes from definition of $\tilde{\mathcal{S}}$, and the guarantee of $\mathcal{M}$. Therefore we show that the output policy is ε-optimal.

Now we compute the sample complexity. It is obvious that the step complexity is HSN_0 since we iterate through all $s \in \mathcal{S}$, and each episode contains H steps. For comparison complexity, we only need to finish S instances of $\mathcal{M}$; therefore the comparison complexity is $O\left(\frac{H^\alpha S \Psi(A, \varepsilon/H, \delta/4S)}{\varepsilon^\alpha}\right)$.

□

6.7.5 Proof of Theorem 72

Proof. The proof follows most parts of that of 70. For any $s \in \tilde{\mathcal{S}}$, we have $N_s \geq \frac{1}{2}\mu(s)N_0$. Guarantee of Beat-the-Mean is as follows:

For simplicity, let $p = 1/\alpha$ and $q = (\alpha - 1)/\alpha$. For $s_h \in \tilde{\mathcal{S}}_h$, let $\varepsilon_{s_h} = \frac{\varepsilon}{2\mu^p(s_h) S^p H^q}$. For $N_0 = \Omega\left(\frac{H^{\alpha-1} S \Psi(A, \frac{\varepsilon}{HS}, \delta/4S)}{\varepsilon^\alpha}\right)$, we have

$$N_{s_h} \geq 1/2\mu(s_h)N_0 = \Omega\left(\frac{\mu(s_h) H^{\alpha-1} S \Psi(A, \frac{\varepsilon}{HS}, \delta/4S)}{\varepsilon^\alpha}\right) \geq \Omega\left(\Psi(A, \varepsilon_{s_h}, \delta/4S) \varepsilon_{s_h}^{-\alpha}\right).$$

Similar to proof of Theorem 70, we can set the constants properly to obtain that

$$v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H}) \leq \varepsilon_{s_h} \tag{6.2}$$

with probability $1 - \delta/4S$. Now suppose (6.2) holds for every $h \in [H]$ and $s_h \in \mathcal{S}_h$, which holds

with probability $1-\delta$. We have

$$\begin{aligned}
V^{\pi^*}(s_0) - V^{\hat{\pi}}(s_0) &= \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \pi_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\leq \sum_{h=1}^{H} E_{s_h \sim P_h^*}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\leq \sum_{h=1}^{H} \Pr_{\pi^*}[s_h \in \tilde{\mathcal{S}}] + \Pr_{\pi^*}[s_h \in \tilde{\mathcal{S}}] E_{s_h \sim P_h^*, s_h \in \tilde{\mathcal{S}}}\left[v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right] \\
&\leq \varepsilon/(2S) \cdot S + \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h)\left(v(s_h; \tilde{\pi}_h^* \circ \hat{\pi}_{h+1:H}) - v(s_h; \hat{\pi}_{h:H})\right) \\
&\leq \varepsilon/2 + \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h) \varepsilon_{s_h}.
\end{aligned}$$

Here the first equality is the performance different lemma (Lemma 13, [129]), and P_h^* is the state distribution of π^* on step h. The first inequality comes from the definition of $\tilde{\pi}^*$. Now note that $P_h^*(s_h) \leq \mu(s_h)$, and that $\sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h) = H$; we have

$$\begin{aligned}
\sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h) \varepsilon_{s_h} &= \frac{\varepsilon}{2S^p H^q} \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} P_h^*(s_h) \mu^{-p}(s_h) \\
&\leq \frac{\varepsilon}{2S^p H^q} \sum_{h=1}^{H} \sum_{s_h \in \tilde{\mathcal{S}}_h} (P_h^*(s_h))^{1-p} \leq \varepsilon/2.
\end{aligned}$$

The inequality holds from Hölder's inequality. Therefore we show that the output policy is ε-optimal.

Now we compute the sample complexity. The step complexity is simply $O(HSN_0)$. For comparison complexity, we roll out at most SN_0 trajectories, so the comparison complexity is at most SN_0.

□

6.7.6 Proof of Theorem 74

Proof. The proof of Theorem 74 is largely the same as Theorem 72. For every state s, let $\tilde{\mu}(s)$ be the probability that $\hat{\pi}_s$ visits s. Similar to Theorem 70 and 72, we have $N_s \geq 1/2\mu(s)N_0$ for $s \in \tilde{\mathcal{S}}$. Therefore by randomly pick a policy from the N_0 episodes, we have $\tilde{\mu}(s) \geq 1/2\mu(s)$.

Using Hoeffding's inequality and property of EULER we have that with probability $1-\delta/2$, for every state s we have

$$\hat{R}_{s_h}/N_1 \geq \tilde{\mu}(s) - \sqrt{\frac{\log(4S/\delta)}{N_0}} \geq 1/2\mu(s) - \sqrt{\frac{\log(4S/\delta)}{N_0}},$$

and therefore $\mu(s) \leq \hat{\mu}(s)$. On the other hand, we also have

$$\hat{\mu}(s) \leq 2\hat{R}_{s_h}/N_1 + 2\sqrt{\frac{\log(4S/\delta)}{N_0}} \leq 2\tilde{\mu}(s) + 4\sqrt{\frac{\log(4S/\delta)}{N_0}} \leq 2\mu(s) + 4\varepsilon/S \leq 6\mu(s).$$

Define $\varepsilon'_{s_h} \leftarrow \frac{\varepsilon}{2(\mu(s_h)SH^{\alpha-1})^{1/\alpha}}$ as in Theorem 72. Therefore we know that with probability $1-\delta/2$, we have $\varepsilon_{s_h} \leq \varepsilon'_{s_h}$ and that $\varepsilon_{s_h} = \Theta(\varepsilon'_{s_h})$. The rest of the proof follows the same process as Theorem 72.

□

6.8 Auxiliary Lemma

We present the performance difference lemma here for completeness. Here we use the version adapted to episode MDPs as in [129].

Lemma 77 (Lemma 13, [129])**.** *For any episode MDP with reward function r and two policies $\pi_{0:H-1}$ and $\pi'_{0:H-1}$, For any $h \in [H]$, let $Q_h(s)$ be the distribution of state s at step h induced by policy $\pi_{0:H-1}$. We have*

$$v_{\pi_{0:H-1}}(s_0) - v_{\pi'_{0:H-1}}(s_0) = \sum_{h=0}^{H-1} E_{s\sim Q_h(s)}[V_{\pi_h \circ \pi'_{h+1:H}}(s) - V_{\pi'_{h:H}}(s)].$$

Algorithm 16 PEPS2: Preferece-based Exploration and Policy Search with 2 phases

Input: Target Accuracy ε, Dueling Bandit algorithm $\mathcal{M}$, quantities N_0, N_1, N_2, success probability δ

for $h \in [H]$ **do**
  $\tilde{S}_h = \emptyset$
  for $s_h \in \mathcal{S}_h$ **do**
    Let $r_{s_h}(s,a) = 1_{s=s_h}$ for all $s \in S, a \in A$
    Obtain a policy $\hat{\pi}_{s_h}$ by using EULER$(r_{s_h}, N_0, \delta/(4S))$ to optimize $r_{s_h}(s,a)$
    Rollout $\hat{\pi}_{s_h}$ for N_1 times and record reward the total reward $\hat{R}_{s_h}$ under r_{s_h}
    Let $\hat{\mu}_{s_h} \leftarrow \min\{1, 2\hat{R}_{s_h}/N_1 + 2\sqrt{\frac{\log(4S/\delta)}{N}}\}$
  end for
end for
Initialize $\hat{\pi} \in \Pi^H$ randomly
for $h = H, H-1, ..., 1$ **do**
  Let $\varepsilon_{s_h} \leftarrow \frac{\varepsilon}{4(\hat{\mu}(s_h)SH^{\alpha-1})^{1/\alpha}}$
  for $s_h \in \mathcal{S}_h$ **do**
    Start an instance of $\mathcal{M}\,(\mathcal{A}, \varepsilon_{s_h}, \delta/(4S))$
    Get next query (a, a') from $\mathcal{M}$
    $U \leftarrow \emptyset$
    for $n \in [N_0]$ **do**
      Execute $\hat{\pi}_{s_h}$ until step h
      if current state is s_h **then**
        Let $\bar{\pi} = a \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a' \circ \hat{\pi}_{h+1:H}$
        Execute $\bar{\pi}$ till step H, obtain trajectory τ
        $U \leftarrow U \cup \tau$
      end if
      if $|U| = 2$ **then**
        Compare the two trajectories in U and return to $\mathcal{M}$
      end if
      If $\mathcal{M}$ has finished, break
    end for
    if $\mathcal{M}$ has finished **then**
      Update $\hat{\pi}_h(s_h)$ to the optimal action according to $\mathcal{M}$
    end if
  end for
end for

Output: Policy $\hat{\pi}$

Algorithm 17 PEPS with Fixed Budget

Input: Budget N, dueling bandit algorithm $\mathcal{M}$, success probability δ

Initialize $\hat{\pi} \in \Pi^H$ randomly
Start an instance of $\mathcal{M}(\mathcal{A}, \delta/(4S))$ at every state $s \in \mathcal{S}$ (denote it by $\mathcal{M}_s$), and get first action (a_s, a'_s)
for $h = H, H-1, ..., 1$ **do**
    for $s_h \in \mathcal{S}_h$ **do**
        Let $r_{s_h}(s, a) = 1_{s=s_h}$ for all $s \in S, a \in A$
        Start an instance of EULER$(r_{s_h}, N_0, \delta/(4S))$
        $U \leftarrow \emptyset$
        for $n \in [N/S]$ **do**
            Obtain a policy $\hat{\pi}_n$ from EULER, and execute $\hat{\pi}_n$ until step h
            Return the trajectory and reward to EULER
            $\tilde{s} \leftarrow$ current state
            Let $\bar{\pi} = a_{\tilde{s}} \circ \hat{\pi}_{h+1:H}$ if $|U| = 0$, otherwise $a'_{\tilde{s}} \circ \hat{\pi}_{h+1:H}$
            Execute $\bar{\pi}$ till step H, obtain trajectory τ
            $U \leftarrow U \cup \tau$
            if $|U| = 2$ **then**
                Compare the two trajectories in U and return to $\mathcal{M}_{\tilde{s}}$
                Get next action $(a_{\tilde{s}}, a'_{\tilde{s}})$ from $\mathcal{M}_{\tilde{s}}$
                Update $\hat{\pi}_h(\tilde{s})$ to the optimal action according to $\mathcal{M}_{\tilde{s}}$
            end if
        end for
    end for
end for

Output: Policy $\hat{\pi}$

Part III

Natural Language Understanding from Multiple Domains

Chapter 7

Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension

Machine Reading Comprehension (MRC) has gained growing interest in the research community [140, 198]. In an MRC task, the machine reads a text passage and a question, and generates (or selects) an answer based on the passage. This requires the machine to possess strong comprehension, inference and reasoning capabilities. Over the past few years, there has been much progress in building end-to-end neural network models [149] for MRC. However, most public MRC datasets (e.g., SQuAD, MS MARCO, TriviaQA) are typically small (less than 100K) compared to the model size (such as SAN [115, 117] with around 10M parameters). To prevent over-fitting, recently there have been some studies on using pre-trained word embeddings [135] and contextual embeddings in the MRC model training, as well as back-translation approaches [198] for data augmentation.

Multi-task learning [43] is a widely studied area in machine learning, aiming at better model generalization by combining training datasets from multiple tasks. In this work, we explore a multi-task learning (MTL) framework to enable the training of one universal model across different MRC tasks for better generalization. Intuitively, this multi-task MRC model can be viewed as an implicit data augmentation technique, which can improve generalization on the target task by leveraging training data from auxiliary tasks. The first method simply adopts a sampling scheme, which randomly selects training data from the auxiliary tasks controlled by a ratio hyperparameter; The second algorithm incorporates recent ideas of data selection in machine translation [167]. It learns the sample weights from the auxiliary tasks automatically through language models.

Prior to this work, many studies have used upstream datasets to augment the performance of MRC models, including word embedding [135], language models (ELMo) [136] and machine translation [198]. These methods aim to obtain a robust semantic encoding of both passages and questions. Our MTL method is orthogonal to these methods: rather than enriching semantic embedding with external knowledge, we leverage existing MRC datasets across different domains, which help make the whole comprehension process more robust and universal. Our experiments show that MTL can bring further performance boost when combined with contextual

representations from pre-trained language models, e.g., ELMo [136].

We validate our MTL framework with two state-of-the-art models on four datasets from different domains. Experiments show that our methods lead to a significant performance gain over single-task baselines on SQuAD [140], NewsQA [163] and Who-Did-What [134], while achieving state-of-the-art performance on the latter two. For example, on NewsQA [163], our model surpassed human performance by **13.4** (46.5 vs 59.9) and **3.2** (72.6 vs 69.4) absolute points in terms of exact match and F1.

The contribution of this work is three-fold. First, we apply multi-task learning to the MRC task, which brings significant improvements over single-task baselines. Second, the performance gain from MTL can be easily combined with existing methods to obtain further performance gain. Third, the proposed sampling and re-weighting scheme can further improve the multi-task learning performance.

7.1 Related Works

Studies in machine reading comprehension mostly focus on architecture design of neural networks, such as bidirectional attention [149], dynamic reasoning [186], and parallelization [198]. Some recent work has explored transfer learning that leverages out-domain data to learn MRC models when no training data is available for the target domain [81]. In this work, we explore multi-task learning to make use of the data from other domains, while we still have access to target domain training data.

Multi-task learning [43] has been widely used in machine learning to improve generalization using data from multiple tasks. For natural language processing, MTL has been successfully applied to low-level parsing tasks [59], sequence-to-sequence learning [123], and web search [114]. More recently, [127] proposes to cast all tasks from parsing to translation as a QA problem and use a single network to solve all of them. However, their results show that multi-task learning hurts the performance of most tasks when tackling them together. Differently, we focus on applying MTL to the MRC task and show significant improvement over single-task baselines.

Our sample re-weighting scheme bears some resemblance to previous MTL techniques that assign weights to tasks [107]. However, our method gives a more granular score for each sample and provides better performance for multi-task learning MRC.

7.2 Model Architecture

We call our model Multi-Task-SAN (MT-SAN), which is a variation of SAN [115] model with two main differences: i) we add a highway network layer after the embedding layer, the encoding layer and the attention layer; ii) we use exponential moving average [149] during evaluation. The SAN architecture and our modifications are briefly described below and in Section 7.4.2, and detailed description can be found in [115].

7.2.1 Input Format

For most tasks we consider, our MRC model takes a triplet (Q, P, A) as input, where $Q = (q_1, ..., q_m), P = (p_1, ..., p_n)$ are the word index representations of a question and a passage, respectively , and $A = (a_{\text{begin}}, a_{\text{end}})$ is the index of the answer span. The goal is to predict A given (Q, P).

7.2.2 Lexicon Encoding Layer

We map the word indices of P and Q into their 300-dim Glove vectors [135]. We also use the following additional information for embedding words: i) 16-dim part-of-speech (POS) tagging embedding; ii) 8-dim named-entity-recognition (NER) embedding; iii) 3-dim exact match embedding: $f_{\text{exact\_match}}(p_i) = \mathbb{I}(p_i \in Q)$, where matching is determined based on the original word, lower case, and lemma form, respectively; iv) Question enhanced passage word embeddings: $f_{\text{align}}(p_i) = \sum_j \gamma_{i,j} h(\text{GloVe}(q_j))$, where

$$\gamma_{i,j} = \frac{\exp(h(\text{GloVe}(p_j)), h(\text{GloVe}(q_i)))}{\sum_{j'} \exp(h(\text{GloVe}(p_{j'})), h(\text{GloVe}(q_i)))} \tag{7.1}$$

is the similarity between word p_j and q_i, and $g(\cdot)$ is a 300-dim single layer neural net with Rectified Linear Unit (ReLU) $g(x) = \text{ReLU}(W_1 x)$; v) Passage-enhanced question word embeddings: the same as iv) but computed in the reverse direction. To reduce the dimension of the input to the next layer, the 624-dim input vectors of passages and questions are passed through a ReLu layer to reduce their dimensions to 125.

After the ReLU network, we pass the 125-dim vectors through a highway network [157], to adapt to the multi-task setting: $g_i = \text{sigmoid}(W_2 p_i^t), p_i^t = \text{ReLU}(W_3 p_i^t) \odot g_i + g_i \odot p_i^t$, where p_i^t is the vector after ReLU transformation. Intuitively, the highway network here provides a neuron-wise weighting, which can potentially handle the large variation in data introduced by multiple datasets.

7.2.3 Contextual Encoding Layer

Both the passage and question encodings go through a 2-layer Bidirectional Long-Short Term Memory (BiLSTM, Hochreiter and Schmidhuber, 1997) network in this layer. We append a 600-dim CoVe vector [126] to the output of the lexicon encoding layer as input to the contextual encoders. For the experiments with ELMo, we also append a 1024-dim ELMo vector. Similar to the lexicon encoding layer, the outputs of both layers are passed through a highway network for multi-tasking. Then we concatenate the output of the two layers to obtain $H^q \in \mathbb{R}^{2d \times m}$ for the question and $H^p = \mathbb{R}^{2d \times n}$ the passage, where d is the dimension of the BiLSTM.

7.2.4 Memory/Cross Attention Layer

We fuse H^p and H^q through cross attention and generate a working memory in this layer. We adopt the attention function from [169] and compute the attention matrix as

$$C = \text{dropout}\left(f_{\text{attention}}(\hat{H}^q, \hat{H}^p)\right) \in \mathbb{R}^{m \times n}.$$

We then use C to compute a question-aware passage representation as

$$U^p = \text{concat}(H^p, H^q C)$$

. Since a passage usually includes several hundred tokens, we use the method of [112] to apply self attention to the representations of passage to rearrange its information:

$$\hat{U}^p = U^p \text{drop}_{\text{diag}}(f_{\text{attention}}(U^p, U^p)),$$

where $\text{drop}_{\text{diag}}$ means that we only drop diagonal elements on the similarity matrix (i.e., attention with itself). Then, we concatenate U^p and $\hat{U}^p$ and pass them through a BiLSTM: $M = \text{BiLSTM}([U^p]; \hat{U}^p])$. Finally, output of the BiLSTM (after concatenating two directions) goes through a highway layer to produce the memory.

7.2.5 Answer Module

The base answer module is the same as SAN, which computes a distribution over spans in the passage. Firstly, we compute an initial state s_0 by self attention on H^q:

$$s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H_j^q)}{\sum_{j'} \exp w_4 H_{j'}^q} \cdot H_j^q\right).$$

The final answer is computed through T time steps. At step $t \in \{1, ..., T-1\}$, we compute the new state using a Gated Recurrent Unit (GRU, Cho et al., 2014) $s_t = \text{GRU}(s_{t-1}, x_t)$, where x_t is computed by attention between M and s_{t-1}: $x_t = \sum_j \beta_j M_j, \beta_j = \text{softmax}(s_{t-1} W_5 M)$. Then each step produces a prediction of the start and end of answer spans through a bilinear function: $P_t^{\text{begin}} = \text{softmax}(s_t W_6 M)$, $P_t^{\text{end}} = \text{softmax}(s_t W_7 M)$. The final prediction is the average of each time step: $P^{\text{begin}} = \frac{1}{T}\sum_t P_t^{\text{begin}}, P^{\text{end}} = \frac{1}{T}\sum_t P_t^{\text{end}}$. We randomly apply dropout on the step level in each time step during training, as done in [115]. During training, the objective is the log-likelihood of the ground truth: $l(Q, P, A) = \log P^{\text{begin}}(a_{\text{begin}}) + \log P^{\text{end}}(a_{\text{end}})$.

Answer module for choosing from candidates. For one of the tasks (WDW), we need to choose an answer from a list of candidates; the candidates are people names that have appeared in the passage. For this task we first use the same way to summary information in questions as in span-based models: $s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H_j^q)}{\sum_{j'} \exp w_4 H_{j'}^q} \cdot H_j^q\right)$. We then compute an attention score via simple dot product: $s = \text{softmax}(s_0^T M)$. The probability of a candidate being the true answer is the aggregation of attention scores for all appearances of the candidate:

$$\Pr(c|Q, P) \propto \sum_{1 \le i \le n} s_i \mathbb{I}(p_i \in C)$$

for each candidate C. Recall that n is the length of passage P, and p_i is the i-th word; therefore $\mathbb{I}(p_i \in C)$ is the indicator function of p_i appears in candidate C. The candidate with the largest probability is chosen as the predicted answer.

7.3 Algorithms

We describe our MTL training algorithms in this section. We start with a very simple and straightforward algorithm that samples one task and one mini-batch from that task at each iteration. To improve the performance of MTL on a target dataset, we propose two methods to re-weight samples according to their importance. The first proposed method directly lowers the probability of sampling from a particular auxiliary task; however, this probability has to be chosen using grid search. We then propose another method that avoids such search by using a language model.

Algorithm 18 Multi-task Learning of MRC

Input: K different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$

1: **for** epoch$= 1, 2, ...$ **do**
2: Divide each dataset $\mathcal{D}_k$ into N_k mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}$, $1 \leq k \leq K$
3: $S \leftarrow \bigcup_{k=1}^{K} \mathcal{D}_k$
4: Randomly shuffle minibatches in S to obtain a sequence $B = (b_1, ..., b_L)$, where $L = |S|$
5: **for** each mini-batch $b \in B$ **do**
6: Perform gradient update on model $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A)\in b} l(Q, P, A)$
7: **end for**
8: **end for**

Suppose we have K different tasks, the simplest version of our MTL training procedure is shown in Algorithm 18. In each epoch, we take all the mini-batches from all datasets and shuffle them for model training, and the same set of parameters is used for all tasks. Perhaps surprisingly, as we will show in the experiment results, this simple baseline method can already lead to a considerable improvement over the single-task baselines.

7.3.1 Mixture Ratio

One observation is that the performance of our model using Algorithm 18 starts to deteriorate as we add more and more data from other tasks into our training pool. We hypothesize that the external data will inevitably bias the model towards auxiliary tasks instead of the target task.

To avoid such adverse effect, we introduce a mixture ratio parameter during training. The training algorithm with the mixture ratio is presented in Algorithm 19, with $\mathcal{D}_1$ being the target dataset. In each epoch, we use all mini-batches from $\mathcal{D}_1$, while only a ratio α of mini-batches from external datasets are used to train the model. In our experiment, we use hyperparameter search to find the best α for each dataset combination. This method resembles previous methods in multi-task learning to weight losses differently (e.g., Kendall et al., 2017), and is very easy to implement. In our experiments, we use Algorithm 19 to train our network when we only use 2 datasets for MTL.

Algorithm 19 Multi-task Learning of MRC with mixture ratio, targeting $\mathcal{D}_1$

Input: K different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$, mixture ratio α

for epoch$= 1, 2, ...$ **do**
  Divide each dataset $\mathcal{D}_k$ into N_k mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}$, $1 \leq k \leq K$
  $S \leftarrow \{b_1^1, ..., b_{N_1}^1\}$
  Randomly pick $\lfloor \alpha N_1 \rfloor$ mini-batches from $\bigcup_{k=2}^{K} \mathcal{D}_k$ and add to S
  Randomly shuffle minibatches in S to obtain a sequence $B = (b_1, ..., b_L)$, where $L = |S|$
  for each mini-batch $b \in B$ **do**
    Perform gradient update on model $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A)\in b} l(Q, P, A)$
  end for
end for

7.3.2 Sample Re-Weighting

The mixture ratio (Algorithm 19) dramatically improves the performance of our system. However, it requires to find an ideal ratio by hyperparameter search which is time-consuming. Furthermore, the ratio gives the same weight to every auxiliary data, but the relevance of every data point to the target task can vary greatly.

We develop a novel re-weighting method to resolve these problems, using ideas inspired by data selection in machine translation [16, 167]. We use (Q^k, P^k, A^k) to represent a data point from the k-th task for $1 \leq k \leq K$, with $k = 1$ being the target task. Since the passage styles are hard to evaluate, we only evaluate data points based on Q^k and A^k. Note that only data from auxiliary task ($2 \leq k \leq K$) is re-weighted; target task data always have weight 1.

Our scores consist of two parts, one for questions and one for answers. For questions, we create language models (detailed in Section 7.4.2) using questions from each task, which we represent as LM_k for the k-th task. For each question Q^k from auxiliary tasks, we compute a cross-entropy score:

$$H_{C,Q}(Q^k) = -\frac{1}{m} \sum_{w \in Q^k} \log(LM_C(w)), \tag{7.2}$$

where $C \in \{1, k\}$ is the target or auxiliary task, m is the length of question Q^k, and w iterates over all words in Q^k.

It is hard to build language models for answers since they are typically very short (e.g., answers on SQuAD includes only one or two words in most cases). We instead just use the length of answers as a signal for scores. Let l_a^k be the length of A^k, the cross-entropy answer score is defined as:

$$H_{C,A}(A^k) = -\log \text{freq}_C(l_a^k), \tag{7.3}$$

where freq_C is the frequency of answer lengths in task $C \in \{1, k\}$.

The cross entropy scores are then normalized over all samples in task C to create a comparable

metric across all auxiliary tasks:

$$H'_{C,Q}(Q^k) = \frac{H_{C,Q}(Q^k) - \min(H_{C,Q})}{\max(H_{C,Q}) - \min(H_{C,Q})} \tag{7.4}$$

$$H'_{C,A}(A^k) = \frac{H_{C,A}(A^k) - \min(H_{C,A})}{\max(H_{C,A}) - \min(H_{C,A})} \tag{7.5}$$

for $C \in \{1, 2, ..., K\}$. For $C \in \{2, ..., K\}$, the maximum and minimum are taken over all samples in task k. For $C = 1$ (target task), they are taken over all available samples.

Intuitively, $H'_{C,Q}$ and $H'_{C,A}$ represents the similarity of text Q, A to task C; a low $H'_{C,Q}$ (resp. $H'_{C,A}$) means that Q^k (resp. A^k) is easy to predict and similar to C, and vice versa. We would like samples that are most similar from data in the target domain (low H'_1), and most different (informative) from data in the auxiliary task (high H'_k). We thus compute the following cross-entropy difference for each external data:

$$\begin{aligned} \text{CED}(Q^k, A^k) =& (H'_{1,Q}(Q^k) - H'_{k,Q}(Q^k)) + \\ & (H'_{1,A}(A^k) - H'_{k,A}(A^k)) \end{aligned} \tag{7.6}$$

for $k \in \{2, ..., K\}$. Note that a low CED score indicates high importance. Finally, we transform the scores to weights by taking negative, and normalize between $[0, 1]$:

$$\text{CED}'(Q^k, A^k) = 1 - \frac{\text{CED}(Q^k, A^k) - \min(\text{CED})}{\max(\text{CED}) - \min(\text{CED})}. \tag{7.7}$$

Here the maximum and minimum are taken over all available samples and task. Our training algorithm is the same as Algorithm 1, but for minibatch b we instead use the loss

$$l(b) = \sum_{(P,Q,A) \in b} \text{CED}'(Q, A) l(P, Q, A) \tag{7.8}$$

in step 6. We define $\text{CED}'(Q^1, A^1) \equiv 1$ for all target samples (P^1, Q^1, A^1).

7.4 Experiment Results

Our experiments are designed to answer the following questions on multi-task learning for MRC:
1. Can we improve the performance of existing MRC systems using multi-task learning?
2. How does multi-task learning affect the performance if we combine it with other external data?
3. How does the learning algorithm change the performance of multi-task MRC?
4. How does our method compare with existing MTL methods?

We first present our experiment details and results for MT-SAN. Then, we provide a comprehensive study on the effectiveness of various MTL algorithms in Section 7.4.4. At last, we provide some additional results on combining MTL with DrQA [48] to show the flexibility of our approach [1].

[1] We include the results in the appendix due to space limitations.

7.4.1 Datasets

We conducted experiments on SQuAD (Rajpurkar et al., 2016), NewsQA[163], MS MARCO (v1, Nguyen et al.,2016) and WDW [134]. Dataset statistics is shown in Table 7.1. Although similar in size, these datasets are quite different in domains, lengths of text, and types of task. In the following experiments, we will validate whether including external datasets as additional input information (e.g., pre-trained language model on these datasets) helps boost the performance of MRC systems.

Dataset	SQuAD(v1)	NewsQA	MS MARCO(v1)	WDW
# Training Questions	87,599	92,549	78,905	127,786
Text Domain	Wikipedia	CNN News	Web Search	Gigaword Corpus
Avg. Document Tokens	130	638	71	365
Answer type	Text span	Text span	Natural sentence	Cloze
Avg. Answer Tokens	3.5	4.5	16.4	N/A

Table 7.1: Statistics of the datasets. Some numbers come from [159].

7.4.2 Experiment Details

We mostly focus on span-based datasets for MT-SAN, namely SQuAD, NewsQA, and MS MARCO. We convert MS MARCO into an answer-span dataset to be consistent with SQuAD and NewsQA, following [115]. For each question, we search for the best span using ROUGE-L score in all passage texts and use the span to train our model. We exclude questions with maximal ROUGE-L score less than 0.5 during training. For evaluation, we use our model to find a span in all passages. The prediction score is multiplied with the ranking score, trained following Liu et al. [116]'s method to determine the final answer.

We train our networks using algorithms in Section 7.3, using SQuAD as the target task. For experiments with two datasets, we use Algorithm 19; for experiments with three datasets we find the re-weighting mechanism in Section 7.3.2 to have a better performance (a detailed comparison will be presented in Section 7.4.4).

For generating sample weights, we build a LSTM language model on questions following the implementation of Merity et al. [128] with the same hyperparameters. We only keep the 10,000 most frequent words, and replace the other words with a special out-of-vocabulary token.

Parameters of MT-SAN are mostly the same as in the original paper [115]. We utilize spaCy[2] to tokenize the text and generate part-of-speech and named entity labels. We use a 2-layer BiLSTM with 125 hidden units as the BiLSTM throughout the model. During training, we drop the activation of each neuron with 0.3 probability. For optimization, we use Adamax [108] with a batch size of 32 and a learning rate of 0.002. For prediction, we compute an exponential moving average (EMA, Seo et al. 2016) of model parameters with a decay rate of 0.995 and use it to compute the model performance. For experiments with ELMo, we use the model implemented by

[2]https://spacy.io

AllenNLP [3]. We truncate passage to contain at most 1000 tokens during training and eliminate those data with answers located after the 1000th token. The training converges in around 50 epochs for models without ELMo (similar to the single-task SAN); For models with ELMo, the convergence is much faster (around 30 epochs).

7.4.3 Performance of MT-SAN

In the following sub-sections, we report our results on SQuAD and MARCO development sets, as well as on the development and test sets of NewsQA [4]. All results are single-model performance unless otherwise noted.

The multi-task learning results of SAN on SQuAD are summarized in Table 7.2. By using MTL on SQuAD and NewsQA, we can improve the exact-match (EM) and F1 score by (2%, 1.5%), respectively, both with and without ELMo. The similar gain indicates that our method is orthogonal to ELMo. Note that our single-model performance is slightly higher than the original SAN, by incorporating EMA and highway networks. By incorporating with multi-task learning, it further improves the performance. The performance gain by adding MARCO is relatively smaller, with 1% in EM and 0.5% in F1. We conjecture that MARCO is less helpful due to its differences in both the question and answer style. For example, questions in MS MARCO are real web search queries, which are short and may have typos or abbreviations; while questions in SQuAD and NewsQA are more formal and well written.

Using 3 datasets altogether provides another marginal improvement. Our model obtains the best results among existing methods that do not use a large language model (e.g., ELMo). Our ELMo version also outperforms any other models which are under the same setting. We note that BERT [65] uses a much larger model than ours(around 20x), and we leave the performance of combining BERT with MTL as interesting future work.

The results of multi-task learning on NewsQA are in Table 7.3. The performance gain with multi-task learning is even larger on NewsQA, with over 2% in both EM and F1. Experiments with and without ELMo give similar results. What is worth noting is that our approach not only achieves new state-of-art results with a large margin but also surpasses human performance on NewsQA.

Finally we report MT-SAN performance on MS MARCO in Table 7.4. Multi-tasking on SQuAD and NewsQA provides a similar performance boost in terms of BLEU-1 and ROUGE-L score as in the case of NewsQA and SQuAD. Our method does not achieve very high performance compared to previous work, probably because we do not apply common techniques like yes/no classification or cross-passage ranking [176].

We also test the robustness of our algorithm by performing another set of experiments on SQuAD and WDW. WDW is much more different than the other three datasets (SQuAD, NewsQA, MS MARCO): WDW guarantees that the answer is always a person, whereas the percentage of such questions in SQuAD is 12.9%. Moreover, WDW is a cloze dataset, whereas in SQuAD and NewsQA answers are spans in the passage. We use a task-specific answer layer in this experiment

[3] https://allennlp.org/

[4] The official submission for SQuAD v1.1 and MARCO v1.1 are closed, so we report results on the development set. According to their leaderboards, performances on development and test sets are usually similar.

Model	Dev Set Performance
Single Model without Language Models	EM,F1
BiDAF [149]	67.7, 77.3
SAN [115]	76.24, 84.06
MT-SAN on SQuAD (single task, ours)	76.84, 84.54
MT-SAN on SQuAD+NewsQA(ours)	78.60, 85.87
MT-SAN on SQuAD+MARCO(ours)	77.79, 85.23
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**78.72, 86.10**
Single Model with ELMo	
SLQA+ [174]	80.0, 87.0
MT-SAN on SQuAD (single task, ours)	80.04, 86.54
MT-SAN on SQuAD+NewsQA(ours)	81.36, 87.71
MT-SAN on SQuAD+MARCO(ours)	80.37, 87.17
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**81.58, 88.19**
BERT [65]	**84.2, 91.1**
Human Performance (test set)	82.30, 91.22

Table 7.2: Performance of our method to train SAN in multi-task setting, competing published results, leaderboard results and human performance, on SQuAD dataset (single model). Note that BERT uses a much larger language model, and is not directly comparable with our results. We expect our test performance is roughly similar or a bit higher than our dev performance, as is the case with other competing models.

Model	Dev Set	Test Set
Model W/o ELMo	EM,F1	EM, F1
Match-LSTM[1]	34.4, 49.6	34.9, 50.0
FastQA[2]	43.7, 56.1	42.8, 56.1
AMANDA[3]	48.4, 63.3	48.4, 63.7
MT-SAN (Single task)	55.8, 67.9	55.6, 68.0
MT-SAN (S+N)	**57.8, 69.9**	**58.3, 70.7**
Model With ELMo		
MT-SAN (Single task)	57.7, 70.4	57.0, 70.4
MT-SAN (S+N)	**60.1, 72.5**	**59.9, 72.6**
Human Performance	-,-	46.5, 69.4

Table 7.3: Performance of our method to train SAN in multi-task setting, with published results and human performance on NewsQA dataset. All SAN results are from our models. "S+N" means jointly training on SQuAD and NewsQA References: [1]: implemented by Trischler et al. (2016). [2]:Weissenborn et al.(2017). [3]: Kundu and Ng(2018).

and use Algorithm 19; the WDW answer module is the same as in AS Reader [102], which we

Model	Scores
Single Model W/o ELMo	
FastQAExt[1] (test set)	33.99, 32.09
Reasonet++[2]	38.62, 38.01
V-Net[3]	-, 45.65
SAN[4]	43.85, 46.14
MT-SAN	34.13, 42.65
MT-SAN: SQuAD+MARCO	34.29, 43.47
MT-SAN: 3 datasets	**36.99, 43.64**
Single Model With ELMo	
MT-SAN	34.57, 42.88
MT-SAN: SQuAD+MARCO	37.02, 43.89
MT-SAN: 3 datasets	**37.12, 44.12**
Human Performance (test set)	48.02, 49.72

Table 7.4: Performance of our method to train SAN in multi-task setting, competing published results and human performance, on MS MARCO dataset. The scores stand for (BLEU-1, ROUGE-L) respectively. All SAN results are our results. "3 dataset" means we train using SQuAD+NewsQA+MARCO. References: [1]: [177]. [2]: implemented by [155]. [3]:[176]. [4]: [115]

Model	SQuAD	WDW
MT-SAN (Single Task)	76.8, 84.5	77.5
MT-SAN (S+W)	**77.6, 85.1**	**78.5**
SOTA	86.2, 92.2	71.7
Human Performance	82.3, 91.2	84

Table 7.5: Performance of MT-SAN on SQuAD Dev and WDW test set. Accuracy is used to evaluate WDW. "S+W" means jointly training on SQuAD and WDW. State of the art (STOA) result comes from [197].

Model	EM, F1	+/-
QANet	73.6, 82.7	0.0, 0.0
QANet + BT	75.1, 83.8	+1.5,+1.1
SAN	76.8, 84.5	0.0, 0.0
MT-SAN	**78.7, 86.0**	**+1.9,+1.5**
SAN + ELMo	80.0, 86.5	+3.2,+2.0
MT-SAN + ELMo	**81.6, 88.2**	**+4.8, +3.7**

Table 7.6: Comparison of methods to use external data. BT stands for back translation [198].

describe in the appendix for completeness. Despite these large difference between datasets, our results (Table 7.5) show that MTL can still provide a moderate performance boost when jointly training on SQuAD (around 0.7%) and WDW (around 1%).

Model	Performance
SQuAD + MARCO	EM,F1
Simple Combine (Alg. 18)	77.1, 84.6
Loss Uncertainty	77.3, 84.7
Mixture Ratio	77.8, 85.2
Sample Re-weighting	**77.9,85.3**
SQuAD + NewsQA + MARCO	
Simple Combine (Alg. 18)	77.6, 85.2
Loss Uncertainty	78.2, 85.6
Mixture Ratio	78.4, 85.7
Sample Re-weighting	**78.8**, **86.0**

Table 7.7: Comparison of different MTL strategies on MT-SAN. Performance is on SQuAD. Loss Uncertainty is from Kendall et al. [107].

	Samples/Groups	CED′	H_Q	H_A
Examples	(NewsQA) Q: Where is the drought hitting? A: Argentina	0.824	0.732	0.951
	(MARCO) Q: thoracic cavity definition A: is the chamber of the human body ... and fascia.	0.265	0.332	0.240
Averages	Samples in NewsQA	0.710	0.593	0.895
	Samples in MARCO	0.587	0.550	0.669
	MARCO Questions that start with “When” or “Who”	0.662	0.605	0.761
	All samples	0.654	0.573	0.791

Table 7.8: Scores for examples from NewsQA and MS MARCO and average scores for specific groups of samples. CED′ is as in (7.7), while H_Q and H_A are normalized version of question and sample scores. “Sum” are the actual scores we use, and “LM”, “Answer” are scores from language models and answer lengths.

Comparison of methods using external data. As a method of data augmentation, we compare our approach to previous methods for MRC in Table 7.6. Our model achieves better performance than back translation. We also observe that language models such as ELMo obtain a higher performance gain than multi-task learning, however, combining it with multi-task learning leads to the most significant performance gain. This validates our assumption that multi-task learning is more robust and is different from previous methods such as language modeling.

7.4.4 Comparison of Different MTL Algorithms

In this section, we provide ablation studies as well as comparisons with other existing algorithms on the MTL strategy. We focus on MT-SAN without ELMo for efficient training.

Table 7.7 compares different multi-task learning strategies for MRC. Both the mixture ratio (Sec 7.3.1) and sample re-weighting (Sec 7.3.2) improves over the naive baseline of simply com-

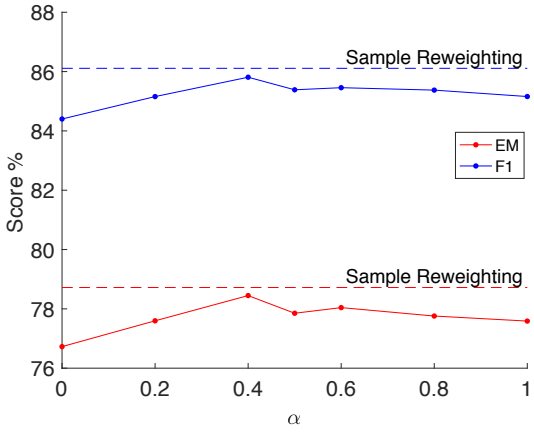


Figure 7.1: Effect of the mixture ratio on the performance of MT-SAN. Note that $\alpha = 0$ is equivalent to single task learning, and $\alpha = 1$ is equivalent to simple combining.

bining all the data (Algorithm 18). On SQuAD+MARCO, they provide around 0.6% performance boost in terms of both EM and F1, and around 1% on all 3 datasets. We note that this accounts for around a half of our overall improvement. Although sample re-weighting performs similar as mixture ratio, it significantly reduces the amount of training time as it eliminates the need for a grid searching the best ratio. Kendal et al., (2017) use task uncertainty to weight tasks differently for MTL; our experiments show that this has some positive effect, but does not perform as well as our proposed two techniques. We note that Kendal et al. (as well as other previous MTL methods) optimizes the network to perform well for all the tasks, whereas our method focuses on the target domain which we are interested in, e.g., SQuAD.

Sensitivity of mixture ratio. We also investigate the effect of mixture ratio on the model performance. We plot the EM/F1 score on SQuAD dev set vs. mixture ratio in Figure 7.1 for MT-SAN when trained on all three datasets. The curve peaks at $\alpha = 0.4$; however if we use $\alpha = 0.2$ or $\alpha = 0.5$, the performance drops by around 0.5%, well behind the performance of sample re-weighting. This shows that the performance of MT-SAN is sensitive to changes in α, making the hyperparameter search even more difficult. Such sensitivity suggests a preference for using our sample re-weighting technique. On the other hand, the ratio based approach is pretty straightforward to implement.

Analysis of sample weights. Dataset comparisons in Table 7.1 and performance in Table 7.2 suggests that NewsQA share more similarity with SQuAD than MARCO. Therefore, a MTL system should weight NewsQA samples more than MARCO samples for higher performance. We try to verify this in Table 7.8 by showing examples and statistics of the sample weights. We present the CED′ scores, as well as normalized version of question and answer scores (resp. $(H'_{1,Q} - H'_{k,Q})$ and $(H'_{1,A} - H'_{k,A})$ in (7.6), and then negated and normalized over all samples in NewsQA and MARCO in the same way as in (7.7)). A high H_Q score indicates high importance of the question, and H_A of the answer; CED′ is a summary of the two. We first show one example from NewsQA and one from MARCO. The NewsQA question is a natural question (similar to SQuAD) with a short answer, leading to high scores both in questions and answers. The MARCO question is a phrase, with a very long answer, leading to lower scores. From overall statistics, we

Setup	SQuAD (v1)	SQuAD (v2)	NewsQA	WDW
Single Dataset	69.5,78.8 (paper) 68.6, 77.8 (ours)	61.9, 65.2	51.9, 64.6	**75.8**
MT-DrQA on Sv1+NA	70.2, 79.3	-,-	52.8, 65.8	-
MT-DrQA on Sv1+W	69.2, 78.4	-,-	-,-	75.7
MT-DrQA on Sv1+N+W	**70.2, 79.3**	-,-	**53.1, 65.7**	75.4
MT-DrQA on Sv2+N	-,-	**63.6, 66.7**	52.7, 65.7	-
MT-DrQA on Sv2+W	-,-	63.5, 66.3	-,-	75.4
MT-DrQA on Sv2+N+W	-,-	63.1, 66.3	52.5, 65.6	75.3
SOTA (Single Model)	80.0, 87.0	72.3, 74.8	48.4, 63.7 (test)	71.7 (test)
MT-DrQA (Best)	**70.2, 79.3**	**63.6, 66.7**	**53.0, 66.2(test)**	**75.4 (test)**
Human Performance (test)	82.3, 91.2	86.8, 89.5	46.5, 69.4	84

Table 7.9: Single model performance of our method to train DrQA on multi-task setting, as well as state-of-the-art (SOTA) results and human performance. SQuAD and NewsQA performance are measured by (EM, F1), and WDW by accuracy percentage. All results are on development set unless otherwise noted by "test". Published SOTA results come from [92, 111, 174, 197] respectively.

also find samples in NewsQA have a higher score than those in MARCO. However, if we look at MARCO questions that start with "when" or "who" (i.e., probability natural questions with short answers), the scores go up dramatically.

7.4.5 Additional Experiments on DrQA

To demonstrate the flexibility of our approach, we also adapt DrQA [48] into our MTL framework. We only test DrQA using the basic Algorithm 19, since our goal is mainly to test the MTL framework.

Model Architecture. Similar to MT-SAN, we add a highway network after the lexicon encoding layer and the contextual encoding layer and use a different answer module for each dataset. We apply MT-DrQA to a broader range of datasets. For span-detection datasets such as SQuAD, we use the same answer module as DrQA. For cloze-style datasets like Who-Did-What, we use the attention-sum reader [102] as the answer module. For classification tasks required by SQuAD v2.0 [141], we apply a softmax to the last state in the memory layer and use it as the prediction.

Performance of MT-DrQA. We apply MT-DrQA to SQuAD (v1.1 and v2.0), NewsQA, and WDW. We follow the setup of [48] for model architecture and hyperparameter setup. We use Algorithm 18 to train all MT-DrQA models. Different than [141], we do not optimize the evaluation score by changing the threshold to predict unanswerable question for SQuAD v2.0; we just use the argmax prediction. As a result, we expect the gap between dev and test performance to be lower for our model. The results of MT-DrQA are presented in Table 7.9. The results of combining SQuAD and NewsQA obtain similar performance boost as our SAN experiment, with a performance boost between 1-2% in both EM and F1 for the two datasets. The results of MTL including WDW is different: although adding WDW to SQuAD still brings a marginal

performance boost to SQuAD, the performance on WDW drops after we add SQuAD and NewsQA into the training process. We conjecture that this negative transfer phenomenon is probably because of the drastic difference between WDW and SQuAD/NewsQA, both in their domain, answer type, and task type; and DrQA might not be capable of caputuring all these features using just one network. We leave the problem of further preventing such negative transfer to future work.

7.5 Conclusion

We proposed a multi-task learning framework to train MRC systems using datasets from different domains and developed two approaches to re-weight the samples for multi-task learning on MRC tasks. Empirical results demonstrated our approaches outperform existing MTL methods and the single-task baselines as well. Interesting future directions include combining with larger language models such as BERT, and MTL with broader tasks such as language inference [119] and machine translation.

Chapter 8

Multi-Source Transfer Learning for Natural Language Understanding in the Medical Domain

Recent advancement in NLP such as BERT [65] has facilitated great improvements in many Natural Language Understanding (NLU) tasks [119]. BERT first trains a language model on an unsupervised large-scale corpus, and then the pretrained model is fine-tuned to adapt to downstream NLU tasks. This fine-tuning process can be seen as a form of transfer learning, where BERT learns knowledge from the large-scale corpus and transfer it to downstream tasks.

We investigate NLU in the medical (scientific) domain, during the MEDIQA 2019 shared tasks competition.The MEDIQA 2019 shared tasks [34] aim to improve the current state-of-the-art systems for textual inference, question entailment and question answering in the medical domain. This ACL-BioNLP 2019 shared task is motivated by a need to develop relevant methods, techniques and gold standards for inference and entailment in the medical domain and their application to improve domain-specific information retrieval and question answering systems. The shared task consists of three parts: i) natural language inference (NLI) on MedNLI, ii) Recognizing Question Entailment (RQE), and iii) Question Answering (QA).

To fit BERT to medical domain NLU, we need to adapt to i) The change from general domain corpus to scientific language; ii) The change from low-level language model tasks to complex NLU tasks. Although there is limited training data in NLU in the medical domain, we fortunately have pre-trained models from two intermediate steps:

- General NLU embeddings: We use MT-DNN [119] trained on GLUE benchmark[172]. MT-DNN is trained on 10 tasks including NLI, question equivalence, and machine comprehension. These tasks correspond well to the target MEDIQA tasks but in different domains.
- Scientific embeddings: We use SciBERT [33], which is a BERT model, but trained on SemanticScholar scientific papers. Although SciBERT obtained state-of-the-art results on several single-sentence tasks, it lacks knowledge from other NLU tasks such as GLUE.

In this chapter, we investigate different methods to combine and transfer the knowledge from the two different sources and illustrate our results on the MEDIQA shared task. We name our method

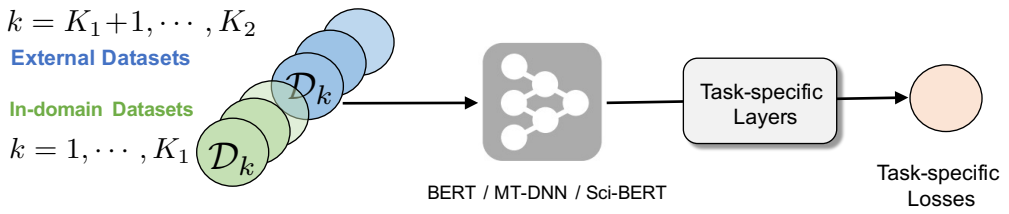


Figure 8.1: Illustration of the proposed multi-source multi-task learning method.

as DoubleTransfer, since it transfers knowledge from two different sources. Our method is based on fine-tuning both MT-DNN and SciBERT using multi-task learning, which has demonstrated the efficiency of knowledge transformation [43, 113, 119, 191], and integrating models from both domains with ensembles.

8.1 Related Works

Transfer learning has been widely used in training models in the medical domain. For example, Romanov and Shivade [145] leveraged the knowledge learned from SNLI to MedNLI; a transfer from general domain NLI to medical domain NLI. They also employed word embeddings trained on MIMIC-III medical notes, which can be seen as a language model in the scientific domain. SciBERT [33] studies transferring knowledge from SciBERT pretrained model to single-sentence classification tasks. Our problem is unique because of the prohibitive cost to train BERT: Either BERT or SciBERT requires a very long time to train, so we only explore how to combine the existing embeddings from SciBERT or MT-DNN. Transfer learning is also widely used in other tasks of NLP, such as machine translation [22] and machine reading comprehension [191].

8.2 Methods

We propose a multi-task learning method for the medical domain data. It employs datasets/tasks from both medical domain and external domains, and leverage the pre-trained model such as MT-DNN and SciBERT for fine-tuning. An overview of the proposed method is illustrated in Figure 8.1. To further improve the performance, we propose to ensemble models trained from different initialization in the evaluation stage. Below we detail our methods for fine-tuning and ensembles.

8.2.1 Fine-tuning details

Algorithm. We fine-tune the two types of pre-trained models on all the three tasks using multi-task learning. As suggested by MEDIQA paper, we also fine-tune our model on MedQuAD [1], a medical QA dataset. We will provide details for fine-tuning on these datasets in Section 8.2.3. We additionally regularize the model by also training on MNLI [179]. To prevent the negative transfer from MNLI, we put a larger weight on MEDIQA data by sampling MNLI data with less probability. Our algorithm is presented in Algorithm 20 and illustrated as Figure 8.1, which is a

Algorithm 20 Multi-task Fine-tuning with External Datasets

Input: In-domain datasets $\mathcal{D}_1, ..., \mathcal{D}_{K_1}$, External domain datasets $\mathcal{D}_{K_1+1}, ..., \mathcal{D}_{K_2}$, max_epoch, mixture ratio α

1: Initialize the model $\mathcal{M}$
2: **for** epoch$= 1, 2, ...,$ max_epoch **do**
3: Divide each dataset $\mathcal{D}_k$ into N_k mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}$, $1 \leq k \leq K_2$
4: $S \leftarrow \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_{K_1}$
5: $N \leftarrow N_1 + N_2 + \cdots + N_{K_1}$
6: Randomly pick $\lfloor \alpha N \rfloor$ mini-batches from $\bigcup_{k=K_1}^{K_2} \mathcal{D}_k$ and add to S
7: Assign mini-batches in S in a random order to obtain a sequence $B = (b_1, ..., b_L)$, where $L = N + \lfloor \alpha N \rfloor$
8: **for** each mini-batch $b \in B$ **do**
9: Perform gradient update on $\mathcal{M}$ with loss $l(b) = \sum_{(s_1,s_2)\in b} l(s_1, s_2)$
10: **end for**
11: Evaluate development set performance on $\mathcal{D}_1, ..., \mathcal{D}_{K_1}$
12: **end for**

Output: Model with best evaluation performance

mixture ratio method for multi-task learning inspired by Xu et al. [191]. We start with in-domain datasets $\mathcal{D}_1, ...\mathcal{D}_{K_1}$ (i.e., the MEDIQA tasks, $K_1 = 3$) and external datasets $\mathcal{D}_{K_1+1}, ..., \mathcal{D}_{K_2}$ (in this case MNLI). We cast all the training samples as sentence pairs $(s_1, s_2) \in \mathcal{D}_k, k = 1, 2, ..., K_2$. In each epoch of training, we use all mini-batches from in-domain data, while only a small proportion (controlled by α) of mini-batches from external datasets are used to train the model. In our experiments, the mixture ratio α is set to 0.5. We use MedNLI, RQE, QA, and MedQuAD in medical domain as in-domain data and MNLI as external data. For MedNLI, we additionally find that using MedNLI as in-domain data and RQE, QA, MedQuAD as external data can also help boost performance. We use models trained using both setups of external data for ensembling.

Pre-trained Models. We use three different types of initialization as the starting point for fine-tuning: i) the uncased MT-DNN large model from Liu et al. [119], ii) the cased knowledge-distilled MT-DNN model from Liu et al. [118], and iii) the uncased SciBERT model [33]. We add a simple softmax layer (or linear layer for QA and MedQuAD tasks) atop BERT as the answer module for fine-tuning. For initialization in step 1 in Algorithm 20, we initialize all BERT weights with the pretrained weights, and randomly initialize the answer layers. After multi-task fine-tuning, the joint model is further fine-tuned on each specific task to get better performance. We detail the training loss and fine-tuning process for each task in Section 8.2.3.

Objectives. MedNLI and RQE are binary classification tasks, and we use a cross-entropy loss. Specifically, for a sentence pair X we compute the loss

$$\mathcal{L}(X) = -\sum_c \mathbb{1}(X, c) \log(P_r(c|X)),$$

where c iterates over all possible classes, $\mathbb{1}(X, c)$ is the binary indicator (0 or 1) if class label c is the correct classification for X, and $P_r(c|X)$ is the model prediction for probability of class c for sample X.

We formulate QA and MedQuAD as regression tasks, and thus a MSE loss is used. Specifically, for a question-answer pair (Q, A) we compute the MSE loss as

$$\mathcal{L}(Q, A) = (y - \texttt{score}(Q, A))^2,$$

where y is the target relevance score for pair (Q, A), and $\texttt{score}(Q, A)$ is the model prediction for the same pair.

8.2.2 Model Ensembles

After fine-tuning, we ensemble models trained from MT-DNN and SciBERT, and using different setups of in-domain and external datasets. The traditional methods typically fuse models by averaging the prediction probability of different models. For our setting, the in-domain data is very limited and it tends to overfit; this means the predictions can be arbitrarily close to 1, favoring to more over-fitting models. To prevent over-fitting, we ensemble the models by using a majority vote on their predictions, and resolving ties using sum of prediction probabilities. Suppose we have M models, and the m-th model predicts the answer $\hat{p}_m$ for a specific question. For the classification task (MedNLI and RQE), we have $\hat{p}_m \in \mathbb{R}^C$, where C is the number of categories. Let $\hat{y}_m = \arg\max_i \hat{p}_m^{(i)}$ be the prediction of model m, where $\hat{p}_m^{(i)}$ is the i-th dimension of $\hat{p}_m$. The final prediction is chosen as

$$\hat{y}_{\text{ensemble}} = \underset{y \in \text{maj}(\{\hat{y}_m\}_{m=1}^M)}{\arg\max} \sum_{m=1}^{M} \hat{p}_m^{(y)}.$$

In other words, we first obtain the majority of predictions by computing the majority $\text{maj}(\{\hat{y}_m\}_{m=1}^M)$, and resolve the ties by computing the sum of prediction probabilities $\sum_{m=1}^M \hat{p}_m^{(y)}$. For QA tasks (QA and MedQuAD), the task is cast as a regression problem, where a positive number means correct answer, and negative otherwise. We have $\hat{p}_m \in \mathbb{R}$. We first compute the average score $\hat{p}_{\text{ensem}} = \frac{1}{M}\sum_{m=1}^M \hat{p}_m$. We also compute the prediction as $\hat{y}_m = I(\hat{p}_m \geq 0)$, where I is the indicator function. We compute the ensemble prediction through a similar majority vote as the classification case:

$$\hat{y}_{\text{ensem}} = \begin{cases} 1, & \text{if } \sum_{m=1}^M \hat{y}_m > M/2 \\ 0, & \text{if } \sum_{m=1}^M \hat{y}_m < M/2 \\ I(\hat{p}_{\text{ensem}} > 0), & \text{otherwise.} \end{cases}$$

To be precise, we predict the majority if a tie does not exist, or the sign of $\hat{p}_{\text{ensem}}$ otherwise. The final ranking of answers is carried out by first rank the (predicted) positive answers, and then the (predicted) negative answers.

8.2.3 Dataset-Specific Details

MedNLI: Since the MEDIQA shared task uses a different test set than the original MedNLI dataset, we merge the original MedNLI development set into the training set and use evaluation performance on the original MedNLI test set. Furthermore, MedNLI and MNLI are the same NLI

tasks, thus, we shared final-layer classifiers for these two tasks. For MedNLI, we find that each consecutive 3 samples in all the training set contain the same premise with different hypothesizes, and contains exactly 1 entail, 1 neutral and 1 contradiction. To the end, in our prediction, we constrain the three predictions to be one of each kind, and use the most likely prediction from the model prediction probabilities.

RQE: We use the clinical question as the premise and question from FAQ as the hypothesis. We find that the test data distribution is quite different from the train data distribution. To mitigate this effect, we randomly shuffle half of the evaluation data into the training set and evaluate on the remaining half.

QA: We use the answer as the premise and the question as the hypothesis. The QA task is cast as both a ranking task and a classification task. Each question is associated with a relevance score in $\{1, 2, 3, 4\}$, and an additional rank over all the answers for a specific question is given. We use a modified score to incorporate both information: suppose there are m questions with relevance score $s \in \{1, 2, 3, 4\}$. Then the i-th most relevant answer in these m questions get modified score $s - \frac{i-1}{m}$. In this way the scores are uniformly distributed in $(s - 1, s]$. We shift all scores by -2 so that a positive score leads to a correct answer and vice versa. We also tried pairwise losses to incorporate the ranking but did not find it to boost the performance very much.

We find that the development set distribution is inconsistent with test data - the training and test set consist of both LiveQAMed and Alexa questions, whereas the development set seems to only contain LiveQAMed questions. We shuffle the training and development set to make them similar: We use the last 25 questions in original development set (LiveQAMed questions) and the last 25 Alexa questions (from the original training set) as our development set, and use the remaining questions as our training set. This results in 1,504 training pairs and 431 validation pairs. Due to the limited size of the QA dataset, we use cross-validation that divides all pairs into 5 slices and train 5 models by using each slice as a validation set. We train MT-DNN and SciBERT on both these 5 setups and obtain 10 models, and ensemble all the 10 models obtained.

MedQuAD: We use 10,109 questions from MedQuAD because the remaining questions are not available due to copyright issues. The original MedQuAD dataset only contains positive question pairs. We add negative samples to the dataset by randomly sampling an answer from the same web page. For each positive QA pair, we add two negative samples. The resulting 30,327 pairs are randomly divided into 27,391 training pairs and 2,936 evaluation pairs. Then we use the same method as QA to train MedQuAD; we also share the same answer module between QA and MedQuAD.

8.2.4 Implementation and Hyperparameters

We implement our method using PyTorch[1] and Pytorch-pretrained-BERT[2], as an extension to MT-DNN[3]. We also use the pytorch-compatible SciBERT pretrained model provided by AllenNLP[4]. Each training example is pruned to at most 384 tokens for MT-DNN models and 512 tokens for SciBERT models. We use a batch size of 16 for MT-DNN, and 40 for SciBERT. For fine-tuning,

[1] `https://pytorch.org/`
[2] `https://github.com/huggingface/pytorch-pretrained-BERT`
[3] `https://github.com/namisan/mt-dnn`
[4] `https://github.com/allenai/scibert`

Model	Dev Set	Test Set
WTMed	-	**98.0**
PANLP	-	96.6
Ours	**91.7**	**93.8**
Sieg	-	91.1
SOTA	76.6	-

Table 8.1: The leaderboard for MedNLI task (link). Scores are accuracy(%). Our method ranked the 3rd on the leaderboard. Previous SOTA method was from [145], on the original MedNLI test set (used as dev set here).

Model	Dev Set	Test Set
PANLP	-	**74.9**
Sieg	-	70.6
IIT-KGP	-	68.4
Ours	**91.7**	**66.2**

Table 8.2: The leaderboard for RQE task (link). Scores are accuracy(%). Our method ranked the 7th on the leaderboard.

we train the models for 20 epochs using a learning rate of 5×10^{-5}. After that, we further fine-tune the model from the best multi-task model for 6 epochs for each dataset, using a learning rate of 5×10^{-6}. We ensemble all models with an accuracy larger than 87.7 for MedNLI, 83.5 for shuffled RQE, and 83.0 for QA. We ensemble 4 models for MedNLI, 14 models for RQE. For QA, we ensemble 10 models from cross-validation and 7 models using the normal training-validation approach.

8.3 Experiment Results

In this section, we provide the leaderboard performance and conduct an analysis of the effect of ensemble models from different sources.

8.3.1 Test Set Performance and LeaderBoards

The results for MedNLI dataset is summarized in Table 8.1. Our method ends up the 3rd place on the leaderboard and substantially improving upon previous state-of-the-art (SOTA) methods.

The results for RQE dataset is summarized in Table 8.2. Our method ends up the 7th place on the leaderboard. Our method has a very large discrepancy between the dev set performance and test set performance. We think this is because the test set is quite different from dev set, and that the dev set is very small and easy to overfit to.

The results for QA dataset is summarized in Table 8.3. Our method reaches the first place on the leaderboard based on accuracy and precision score and 3rd-highest MRR. We note that the Spearman score is not consistent with other scores in the leaderboard; actually, the Spearman

Model	Acc	Spearman	Precision	MRR
Ours	**78.0**	0.238	**81.91**	0.937
PANLP	77.7	0.180	78.1	0.938
Pentagon	76.5	0.338	77.7	**0.962**
DUT-BIM	74.5	0.106	74.7	0.906

Table 8.3: The leaderboard for QA task (link). Our method ranked #1 on the leaderboard in terms of Acc (accuracy). The Spearman score is not consistent with other scores in the leaderboard.

score is computed just based on the predicted positive answers, and a method can get very high Spearman score by never predict positive labels.

8.3.2 Ensembles from Different Sources

We compare the effect of ensembling from different sources in Table 8.4. We train 6 different models with different randomizations, with initializations from MT-DNN (#1,#2,#3) and SciBERT (#4, #5,#6) respectively. If we ensemble models with the same MT-DNN architecture, the resulting model only has around 1.5% improvement in accuracy, compared to the numerical average of the ensemble model accuracies (#1+#2+#3 and #4+#5+#6 in Table 8.4). On the other hand, if we ensemble three models from different sources (#1+#2+#5 and #1+#5+#6 in Table 8.4), the resulting model gains more than 3% in accuracy compared to the numerical average. This shows that ensembling from different sources has a great advantage than ensembling from single-source models.

8.3.3 Single-Model Performance

For completeness, we report the single-model performance on the MedNLI development set under various multi-task learning setups and initializations in Table 8.5. (1) The *Naïve* approach denotes only MedNLI, RQE, QA, MedQuAD is considered as in-domain data in Algorithm 20 without any external data; (2) The *Ratio* approach denotes that we consider MedNLI as in-domain data, and RQE, QA, MedQuAD as external data in Algorithm 20; (3) The *Ratio+MNLI* approach denotes that we consider MedNLI, RQE, QA, MedQuAD as in-domain data and MNLI as external data in Algorithm 20. Note that MNLI is much larger than the medical datasets, so if we use RQE, QA, MedQuAD, MNLI as external data, the performance is very similar to the third setting. We did not conduct experiments on single-dataset settings, as previous works have suggested that multi-task learning can obtain much better results than single-task models [119, 191].

Overall, the best results are achieved via using SciBERT as the pre-trained model, and multi-task learning with MNLI. The models trained by mixing in-domain data (the second setup) is also competitive. We therefore use models from both setups for ensemble.

Model	Avg. Acc	Esm. Acc
Single Model		
#1, MT-DNN	-	88.61
#2, MT-DNN	-	88.33
#3, MT-DNN	-	87.84
#4, SciBERT	-	88.19
#5, SciBERT	-	87.70
#6, SciBERT	-	87.21
Ensemble Model		
#1+#2+#3, MT-DNN	88.26	89.7
#4+#5+#6, SciBERT	87.70	89.2
#1+#2+#5, MultiSource	88.21	91.6
#1+#5+#6, MultiSource	87.84	90.4
#1-6, MultiSource	87.98	91.3

Table 8.4: Comparison of ensembles from different sources. Avg.Acc stands for average accuracy, the numerical average of each individual model's accuracy. Esm.Acc stands for ensemble accuracy, the accuracy of the resulting ensemble model. For ensembles, MT-DNN means all the three models are from MT-DNN, and similarly for SciBERT; MultiSource denotes the ensemble models come from two different sources.

Init Model	Naïve	Ratio	Ratio+MNLI
MT-DNN	86.9	86.2	87.8
MT-DNN-KD	87.5	88.2	**88.8**
SciBERT	87.1	87.0	**89.4**

Table 8.5: Single model performance on MedNLI developlment data. *Naiïve* means simply integrating all medical-domain data; *Ratio* means using MedNLI as in-domain data and other medical domain data as external data; *Ratio+MNLI* means using medical domain data as in-domain and MNLI as external.

8.4 Conclusion

We present new methods for multi-source transfer learning for the medical domain. Our results show that ensembles from different sources can improve model performance much more greatly than ensembles from a single source. Our methods are proved effective in the MEDIQA2019 shared task.

Chapter 9

Conclusion and Discussion

Throughout this thesis, we develop various ways to incorporate diverse forms of information into the machine learning and decision making process. The methodology in this thesis is both theoretical and practical: We prove performance guarantees and minimax optimality for our algorithms, while also demonstrating competitive performance on real-world datasets. The contribution of the current thesis covers two main forms of such additional information, preferences and multi-task learning.

- **Learning from Preferences.**We used comparisons to help accelerate the learning process for classification, regression, multi-armed bandits and reinforcment learning problems. In the applications that we consider, comparisons are available at a much cheaper cost than direct queries; this includes many scenarios in clinical trials, material science and information retrieval. Comparisons represents a tradeoff between label accuracy and amount of information in the labels: Comparisons are cheaper and usually more accuracy than direct labels, but a exact comparison carries less information than a exact direct label, since it is binary. Through a effcient link between comparisons and direct labels, we show that we are able to achieve the same performance using either comparisons or direct queries. Therefore, the overall learning cost is reduced by replacing direct queries with comparisons.

  We take two ways to incorporate comparisons into the learning process. In the first way, we use comparisons to infer direct labels, and use inferred labels for the learning process. This is the case for classification (Chapter 2), regression (Chapter 3) and thresholding bandits (Chapter 5). In the second way, we use comparisons to directly optimize for the optimal action in decision making, and direct labels are not necessarily needed. This is the case for multi-armed bandits (Chapter 4) and reinforcement learning (Chapter 6).
- **Multi-Task and Transfer Learning.** In multi-task and transfer learning, the additional information comes from a similar from as the main task, but is from a different domain. Our method to use sample-reweighting and multi-source transfer for learning from multiple domains represents a outstanding example of how diverse forms of information are able to help machine learning models gain additional performance boost.

In summary, we give an affirmative answer to the question we raised in Chapter 1: We show that the overall learning cost can be significantly reduced by learning with diverse forms of information. Human beings are able to understand and answer various kinds of questions, and

we believe that machines should not only ask direct queries on the very same domain. With the increasing need of data from modern machine learning, our methods and insights can make machine learning more *accessible and practical* to the human society. By incorporating preference feedback and multi-task data into the machine learning algorithms, we increase the amount of information that a machine can obtain for the task at hand.

Future Directions. While we have explored a few ways to incorporate diverse information into machine learning, we believe that many future directions remain to further improve the amount of information that a machine can take in. For learning from comparisons, one important question is how to incorporate comparisons for even more complicated applications, e.g., reinforcement learning with infinite state space, language generation or image generation tasks, and machine translation. These tasks also have natural preference queries that can be elicited in an easier way than direct queries.

More broadly, an important future direction is to find more intuitive ways to elicit the knowledge from human labelers. In addition to comparisons and multiple domains, people have considered learning from features[63], from corrections[62], from games [170], and many more. An important question is to understand what effect do these diverse forms of information play in the learning process, and how they can be transformed into guaranteed on the original learning problem.

Bibliography

[1] Asma Ben Abacha and Dina Demner-Fushman. A Question-Entailment Approach to Question Answering. *arXiv preprint arXiv:1901.08079*, 2019. 8.2.1

[2] Yasin Abbasi-Yadkori, Peter Bartlett, Xi Chen, and Alan Malek. Large-Scale Markov Decision Problems with KL Control Cost and its Application to Crowdsourcing. In *In Proceedings of the International Conference on Machine Learning (ICML)*, 2016. 5

[3] Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38, 2017. 4.6

[4] Shivani Agarwal and Partha Niyogi. Stability and generalization of bipartite ranking algorithms. In *International Conference on Computational Learning Theory*, pages 32–47. Springer, 2005. 2.2

[5] Shivani Agarwal and Partha Niyogi. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, 10:441–474, 2009. 2.2

[6] E Agustsson, R Timofte, S Escalera, X Baro, I Guyon, and R Rothe. Apparent and real age estimation in still images with deep residual regressors on APPA-REAL database. In *12th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2017*. IEEE, 2017. 3.5.3

[7] Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. *arXiv preprint arXiv:0710.2889*, 2007. 2, 2.4.2, 2.4.2, 2.9.1, 2.9.1

[8] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In *International Conference on Machine Learning*, pages 856–864, 2014. 4.3, 4.5.2

[9] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016. 6

[10] Raman Arora, Teodor V Marinov, and Mehryar Mohri. Corralling stochastic bandit algorithms. *arXiv preprint arXiv:2006.09255*, 2020. 4.6

[11] Josh Attenberg, Prem Melville, and Foster Provost. A unified approach to active dual supervision for labeling features and examples. In *Machine Learning and Knowledge Discovery in Databases*, pages 40–55. Springer, 2010. 2

[12] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2014. 6.1

[13] Pranjal Awasthi, Maria Florina Balcan, and Philip M Long. The power of localization for

efficiently learning linear separators with noise. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 449–458. ACM, 2014. 3.4.1, 3.1, 3.4.3, 3.5, 41, 3.8.8

[14] Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Hongyang Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In *Annual Conference on Learning Theory*, pages 152–192, 2016. 2, 2.2, 2.2, 2.4.1, 2.6, 3.4.1, 3.1, 3.6

[15] Pranjal Awasthi, Maria-Florina Balcan, and Philip M Long. The Power of Localization for Efficiently Learning Linear Separators with Noise. *Journal of the ACM*, 63(6):50, 2017. 2, 2.2, 2.4.1, 2.6, 2.6, 2.9.4, 2.9.4, 15, 2.9.4, 17, 2.9.4, 2.9.4

[16] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*, pages 355–362. Association for Computational Linguistics, 2011. 7.3.2

[17] Mohammad Gheshlaghi Azar, Remi Munos, M Ghavamzadaeh, and Hilbert J Kappen. Speedy Q-learning. 2011. 6.3

[18] Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012. 6.3

[19] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013. 6.3

[20] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. *arXiv preprint arXiv:1703.05449*, 2017. 6, 6.2, 6.4

[21] J Andrew Bagnell, Sham M Kakade, Jeff G Schneider, and Andrew Y Ng. Policy search by dynamic programming. In *Advances in neural information processing systems*, pages 831–838, 2004. 6.3

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 8.1

[23] Maria-Florina Balcan and Steve Hanneke. Robust Interactive Learning. In *COLT*, pages 20–21, 2012. 2, 2.2

[24] Maria-Florina Balcan and Philip M Long. Active and passive learning of linear separators under log-concave distributions. In *Annual Conference on Learning Theory*, pages 288–316, 2013. 15

[25] Maria-Florina Balcan and Hongyang Zhang. Noise-Tolerant Life-Long Matrix Completion via Adaptive Sampling. In *Advances in Neural Information Processing Systems*, pages 2955–2963, 2016. 2

[26] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72. ACM, 2006. 2, 2.4.1

[27] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In

International Conference on Computational Learning Theory, pages 35–50. Springer, 2007. 2.2

[28] Maria-Florina Balcan, Ellen Vitercik, and Colin White. Learning Combinatorial Functions from Pairwise Comparisons. *arXiv preprint arXiv:1605.09227*, 2016. 2, 2.2

[29] R E Barlow. *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. J. Wiley, 1972. 3.2, 3.3.2

[30] Andrew R Barron. Complexity Regularization with Application to Artificial Neural Networks. chapter 7, pages 561–576. Springer Netherlands, 1991. 3.3.3

[31] Pierre C Bellec. Sharp oracle inequalities for least squares estimators in shape restricted regression. *The Annals of Statistics*, 46(2):745–780, 2018. 3.6

[32] Pierre C Bellec and Alexandre B Tsybakov. Sharp oracle bounds for monotone and convex regression through aggregation. *Journal of Machine Learning Research*, 16:1879–1892, 2015. 3.6

[33] Iz Beltagy, Arman Cohan, and Kyle Lo. SciBERT: Pretrained Contextualized Embeddings for Scientific Text. *arXiv preprint arXiv:1903.10676*, 2019. 1.1.3, 8, 8.1, 8.2.1

[34] Asma Ben Abacha, Chaitanya Shivade, and Dina Demner-Fushman. Overview of the MEDIQA 2019 Shared Task on Textual Inference, Question Entailment and Question Answering. In *Proceedings of the BioNLP 2019 workshop, Florence, Italy, August 1, 2019*. Association for Computational Linguistics, 2019. 1.1.3, 8

[35] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016. 6

[36] Alina Beygelzimer, Daniel J Hsu, John Langford, and Chicheng Zhang. Search Improves Label for Active Learning. In *Advances in Neural Information Processing Systems*, pages 3342–3350, 2016. 2, 2.2

[37] Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005. 2.9.1

[38] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 3.2, 4.5, 4.5.2, 5.5, 6.5

[39] Mark Braverman and Elchanan Mossel. Sorting from noisy information. *arXiv preprint arXiv:0910.1191*, 2009. 3.1, 3.2, 3.3.4, 28, 2, 3.5.1

[40] Sebastian Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple Identifications in Multi-Armed Bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. 5, 5.1

[41] Adam D Bull. Convergence Rates of Efficient Global Optimization Algorithms. *Journal of Machine Learning Research*, 12:2879–2904, nov 2011. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1953048.2078198`. 4.3

[42] Robert Busa-Fekete, Eyke Hüllermeier, and Adil El Mesaoudi-Paul. Preference-based

Online Learning with Dueling Bandits: A Survey. *arXiv preprint arXiv:1807.11398*, 2018. 4.3, 6.1

[43] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 7, 7.1, 8

[44] Rui M Castro and Robert D Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008. 2, 2.1, 1, 3.8.7

[45] Sabyasachi Chatterjee, Adityanand Guntuboyina, and Bodhisattva Sen. On risk bounds in isotonic and other shape restricted regression problems. *Ann. Statist.*, 43(4):1774–1800, 2015. 3.6

[46] Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *Advances in Neural Information Processing Systems*, pages 343–351, 2010. 3.8.1

[47] Kamalika Chaudhuri, Sham M Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence rates of active learning for maximum likelihood estimation. In *Advances in Neural Information Processing Systems*, pages 1090–1098, 2015. 3.2, 3.4.1

[48] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. *arXiv preprint arXiv:1704.00051*, 2017. 7.4, 7.4.5, 7.4.5

[49] Hung Chen. Lower rate of convergence for locating a maximum of a function. *The Annals of Statistics*, pages 1330–1334, 1988. 4.3

[50] Jiecao Chen, Xi Chen, Qin Zhang, and Yuan Zhou. Adaptive Multiple-{A}rm Identification. In *In Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 5

[51] Lijie Chen, Jian Li, and Mingda Qiao. Nearly Instance Optimal Sample Complexity Bounds for Top-k Arm Selection. In *Proceedings of the AISTATS*, 2017. 5

[52] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 379–387, 2014. 5.1, 5.2.1, 5.3.2, 5.6.1

[53] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM, 2013. 5.6.1

[54] Xi Chen, Qihang Lin, and Dengyong Zhou. Statistical decision making for optimal budget allocation in crowd labeling. *The Journal of Machine Learning Research*, 16(1):1–46, 2015. 5, 5.1

[55] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 7.2.5

[56] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. *arXiv preprint arXiv:1704.00445*, 2017. 4.5, 4.5.1, 4.5.1, 4.5.3, 4.5.3, 4.7.3, 52

[57] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017. 6

[58] Alon Cohen, Tamir Hazan, and Tomer Koren. Tight bounds for bandit combinatorial

optimization. *arXiv preprint arXiv:1702.07539*, 2017. 6.1, 6.2

[59] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011. 7.1

[60] Cecil C Craig. On the {T}chebychef inequality of {B}ernstein. *The Annals of Mathematical Statistics*, 4(2):94–102, 1933. 3.9

[61] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015. 6.2

[62] Sanjoy Dasgupta and Michael Luby. Learning from partial correction. *arXiv preprint arXiv:1705.08076*, 2017. 3.2, 9

[63] Sanjoy Dasgupta, Akansha Dey, Nicholas Roberts, and Sivan Sabato. Learning from discriminative feature feedback. In *Advances in Neural Information Processing Systems*, pages 3955–3963, 2018. 9

[64] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13:2655–2697, 2012. 2

[65] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 7.4.3, 8

[66] Persi Diaconis and Ronald L Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977. 3.8.3

[67] Simon S Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dud\'\ik, and John Langford. Provably efficient rl with rich observations via latent state decoding. *arXiv preprint arXiv:1901.09018*, 2019. 6, 6.4.1, 6.4.2

[68] Dheeru Dua and Casey Graff. {UCI} Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`. 4.5.6

[69] Miroslav Dud\'\ik, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. *arXiv preprint arXiv:1502.06362*, 2015. 4.1

[70] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105, 2006. 1.1.2, 4.4.2

[71] Felix A Faber, Alexander Lindmaa, O Anatole Von Lilienfeld, and Rickard Armiento. Machine Learning Energies of 2 Million Elpasolite (A B C 2 D 6) Crystals. *Physical review letters*, 117(13):135502, 2016. 4.1

[72] Felix A Faber, Alexander Lindmaa, O Anatole von Lilienfeld, and Rickard Armiento. Machine Learning Energies of 2 Million {E}lpasolite{(ABC2D6)}Crystals. *Physical Review Letters*, 117(13), 2016. 3

[73] Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and Ranking with Few Assumptions. In *Advances in Neural Information Processing Systems*, pages 7060–7070, 2017. 6.1, 6.2, 6.2, 6.3, 6.4.1

[74] Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. *arXiv preprint arXiv:1705.05366*, 2017. 4.3, 6.1, 6.2, 6.2, 6.2.1

[75] Peter C Fishburn. Binary choice probabilities: on the varieties of stochastic transitivity. *Journal of Mathematical psychology*, 10(4):327–352, 1973. 5.2

[76] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan 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. 4.3

[77] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference learning*, pages 65–82. Springer, 2010. 2, 2.2

[78] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2):123–156, 2012. 6.2.1

[79] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015. 4.3

[80] Edgar N Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31 (3):504–522, 1952. 3.8.2

[81] David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. Two-stage synthesis networks for transfer learning in machine comprehension. *arXiv preprint arXiv:1706.09789*, 2017. 7.1

[82] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006. 3.3.1, 3.3.2, 3.8.1, 35, 3.8.4

[83] Qiyang Han, Tengyao Wang, Sabyasachi Chatterjee, and Richard J Samworth. Isotonic regression in general dimensions. *arXiv preprint arXiv:1708.09468*, 2017. 3.6

[84] Steve Hanneke. Adaptive Rates of Convergence in Active Learning. In *COLT*. Citeseer, 2009. 2.1, 2.2, 2.3

[85] Steve Hanneke. *Theoretical foundations of active learning*. ProQuest, 2009. 3.2, 3.1, 3.6

[86] Steve Hanneke. Theory of active learning, 2014. 2, 2.1, 2.2, 2.1, 2.2, 2.5, 2.7, 2.9.3, 13, 2.9.6, 2.9.7

[87] Steve Hanneke and Liu Yang. Surrogate losses in passive and active learning. *arXiv preprint arXiv:1207.3772*, 2012. 2.7

[88] Wolfgang Karl Härdle, Marlene Müller, Stefan Sperlich, and Axel Werwatz. *Nonparametric and semiparametric models*. Springer Science & Business Media, 2012. 3.5.2

[89] Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter Optimization: A Spectral Approach, 2017. 4.3

[90] Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright. Active Ranking from Pairwise Comparisons and when Parametric Assumptions Don't Help, 2016. 2, 4.5.2, 5.6.1, 5.8.2

[91] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 7.2.3

[92] Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, and Others. Read+ Verify: Machine Reading Comprehension with Unanswerable Questions. *arXiv preprint arXiv:1808.05759*, 2018. 7.9

[93] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *Advances in neural information processing systems*, pages 575–583, 2013. 6

[94] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015. 6

[95] Kevin G Jamieson and Robert Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems*, pages 2240–2248, 2011. 2

[96] Kevin G Jamieson, Robert Nowak, and Ben Recht. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2012. 4.3

[97] Nan Jiang and Alekh Agarwal. Open problem: The dependence of sample complexity lower bounds on planning horizon. In *Conference On Learning Theory*, pages 3395–3398, 2018. 6.2

[98] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning Provably Efficient? *arXiv preprint arXiv:1807.03765*, 2018. 6, 6.4, 6.4.4

[99] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-Free Exploration for Reinforcement Learning. *arXiv preprint arXiv:2002.02794*, 2020. 6, 6.4.1, 6.4.2

[100] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002. 3.5.1

[101] D R Jones, C D Perttunen, and B E Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, oct 1993. ISSN 1573-2878. doi: 10.1007/BF00941892. URL `https://doi.org/10.1007/BF00941892`. 4.5.6

[102] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016. 7.4.3, 7.4.5

[103] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity gaussian process bandit optimisation. *arXiv preprint*

arXiv:1603.06288, 2016. (document), 4.3, 4.5.1, 4.5.3, 4.5.5, 4.5.6, 53, 4.7.4, 57, 4

[104] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian Optimisation with Continuous Approximations. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1799–1808. JMLR.org, 2017. URL `http://dl.acm.org/citation.cfm?id=3305381.3305567`. 4.3

[105] Daniel M Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. *arXiv preprint arXiv:1704.03564*, 2017. 2, 4.3, 5.1, 5.3.2

[106] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17(1):1–42, 2016. 5.8.2, 65

[107] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 3, 2017. 7.1, 7.3.1, 7.7, 7.4.4

[108] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7.4.2

[109] Akshay Krishnamurthy. Interactive Algorithms for Unsupervised Machine Learning. Technical report, DTIC Document, 2015. 2.2

[110] Wataru Kumagai. Regret Analysis for Continuous Dueling Bandit. In *Advances in Neural Information Processing Systems*, pages 1489–1498, 2017. 4.3

[111] Souvik Kundu and Hwee Tou Ng. A Question-Focused Multi-Factor Attention Network for Question Answering. *arXiv preprint arXiv:1801.08290*, 2018. 7.3, 7.9

[112] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017. 7.2.4

[113] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, 2015. 8

[114] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015. 7.1

[115] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*, 2017. 7, 7.2, 7.2.5, 7.4.2, 7.4

[116] Xiaodong Liu, Kevin Duh, and Jianfeng Gao. Stochastic Answer Networks for Natural Language Inference. *arXiv preprint arXiv:1804.07888*, 2018. 7.4.2

[117] Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. Stochastic Answer Networks for SQuAD 2.0. *arXiv preprint arXiv:1809.09194*, 2018. 7

[118] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding, 2019. 8.2.1

[119] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-Task Deep Neural Networks for Natural Language Understanding. *arXiv preprint arXiv:1901.11504*, 2019. 1.1.3, 7.5, 8, 8.2.1, 8.3.3

[120] Andrea Locatelli, Maurilio Gutzeit, and Alexandra Carpentier. An optimal algorithm for the thresholding bandit problem. In *Proceedings of the 33rd International Conference on Machine Learning-Volume 48*, pages 1690–1698. JMLR. org, 2016. 5, 5.1, 5.2.1, 5.6.1

[121] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358, 2007. 15, 3.4.1, 3.4.2, 41

[122] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005. 3.6

[123] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015. 7.1

[124] Subhransu Maji and Gregory Shakhnarovich. Part and attribute discovery from relative annotations. *International Journal of Computer Vision*, 108(1-2):82–96, 2014. 2, 2.2

[125] Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning. *arXiv Mathematics e-prints*, page math/0702683, feb 2007. 5.5

[126] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017. 7.2.3

[127] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The Natural Language Decathlon: Multitask Learning as Question Answering. *arXiv preprint arXiv:1806.08730*, 2018. 7.1

[128] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*, 2017. 7.4.2

[129] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic State Abstraction and Provably Efficient Rich-Observation Reinforcement Learning. *arXiv preprint arXiv:1911.05815*, 2019. 6, 6.4.1, 6.4.2, 6.7.4, 6.7.5, 6.8, 77

[130] Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top-k rank aggregation from noisy comparisons. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2488–2497. JMLR. org, 2017. 5

[131] Subhojyoti Mukherjee, Naveen Kolar Purushothama, Nandan Sudarsanam, and Balaraman Ravindran. Thresholding bandits with augmented UCB. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2515–2521. AAAI Press, 2017. 5.1, 5.6.1

[132] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. 7.4.1

[133] Ellen R Novoseller, Yanan Sui, Yisong Yue, and Joel W Burdick. Dueling Posterior Sampling for Preference-Based Reinforcement Learning. 2019. 6, 6.2.1, 6.5

[134] Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457*, 2016. 7, 7.4.1

[135] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`. 7, 7.2.2

[136] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. 7

[137] Robin L Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975. 3.6

[138] Stefanos Poulis and Sanjoy Dasgupta. Learning with Feature Feedback: from Theory to Practice. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017. 3.2

[139] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 43–52. ACM, 2008. 4.1

[140] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 7, 7.4.1

[141] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. *arXiv preprint arXiv:1806.03822*, 2018. 7.4.5, 7.4.5

[142] Siddartha Y Ramamohan, Arun Rajkumar, and Shivani Agarwal. Dueling bandits: Beyond condorcet winners to general tournament solutions. In *Advances in Neural Information Processing Systems*, pages 1253–1261, 2016. 4.1

[143] Idan Rejwan and Yishay Mansour. Top-$ k $ Combinatorial Bandits with Full-Bandit Feedback. In *Algorithmic Learning Theory*, pages 752–776, 2020. 6.1

[144] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952. 4.1

[145] Alexey Romanov and Chaitanya Shivade. Lessons from Natural Language Inference in the Clinical Domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, 2018. 8.1, 8.1

[146] Sivan Sabato and Tom Hess. Interactive algorithms: from pool to stream. In *Annual Conference On Learning Theory*, pages 1419–1439, 2016. 2

[147] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 3.5.3

[148] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Multi-Fidelity Black-Box Optimization with Hierarchical Partitions. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4538–4547, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL `http://proceedings.mlr.press/v80/sen18a.html`. 4.3

[149] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. 7, 7.1, 7.2, 7.4.2

[150] Nihar Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *Journal of Machine Learning Research*, 2016. 3, 3.5.4, 3.6

[151] Nihar Shah, Sivaraman Balakrishnan, Aditya Guntuboyina, and Martin Wainwright. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. *{IEEE} {T}ransactions on {I}nformation {T}heory*, 2016. 3.2, 5.2

[152] Nihar B Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. When is it Better to Compare than to Score? *arXiv preprint arXiv:1406.6618*, 2014. 2, 2.2

[153] Nihar B Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin J Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *The Journal of Machine Learning Research*, 17(1): 2049–2095, 2016. 1.1.1, 5, 5.2

[154] Nihar B Shah, Sivaraman Balakrishnan, and Martin J Wainwright. A Permutation-based Model for Crowd Labeling: Optimal Estimation and Robustness, 2016. 3

[155] Yelong Shen, Xiaodong Liu, Kevin Duh, and Jianfeng Gao. An Empirical Analysis of Multiple-Turn Reasoning Strategies in Reading Comprehension Tasks. *arXiv preprint arXiv:1711.03230*, 2017. 7.4

[156] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 1.1.2, 4.3, 4.5, 4.5.1, 4.5.1, 4.5.6

[157] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 7.2.2

[158] Neil Stewart, Gordon D A Brown, and Nick Chater. Absolute identification by relative judgment. *Psychological review*, 112(4):881, 2005. 2.2

[159] Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. Evaluation Metrics for Machine Reading Comprehension: Prerequisite Skills and Readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 806–817, 2017. 7.1

[160] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Multi-dueling bandits with

dependent arms. *arXiv preprint arXiv:1705.00253*, 2017. 4.2, 4.3, 4.5.6

[161] Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. Online rank elicitation for Plackett-Luce: A dueling bandits approach. In *Advances in Neural Information Processing Systems*, pages 604–612, 2015. 5.2

[162] Louis L Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927. 3.2, 3.6, 4.5, 4.5.2, 5.5

[163] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016. 7, 7.4.1, 7.3

[164] Kristi Tsukida and Maya R Gupta. How to analyze paired comparison data. Technical report, DTIC Document, 2011. 1.1.1, 3

[165] Alexandre B Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, pages 135–166, 2004. 2, 3.6, 5.5

[166] Alexandre B Tsybakov. Introduction to nonparametric estimation. Revised and extended from the 2004 French original. Translated by Vladimir Zaiats, 2009. 3.3.1, 3.8.2, 3.8.5, 42

[167] Marlies van der Wees, Arianna Bisazza, and Christof Monz. Dynamic data s election for neural machine translation. *arXiv preprint arXiv:1708.00712*, 2017. 7, 7.3.2

[168] R R Varshamov. Estimate of the number of signals in error correcting codes. In *Dokl. Akad. Nauk SSSR*, 1957. 3.8.2

[169] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 7.2.4

[170] Luis Von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM, 2006. 9

[171] Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity comparisons for interactive fine-grained categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2014. 2, 2.2

[172] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. 2019. 8

[173] Ruosong Wang, Simon S Du, Lin F Yang, and Sham M Kakade. Is Long Horizon Reinforcement Learning More Difficult Than Short Horizon Reinforcement Learning? *arXiv preprint arXiv:2005.00527*, 2020. 6.2

[174] Wei Wang, Ming Yan, and Chen Wu. Multi-Granularity Hierarchical Attention Fusion Networks for Reading Comprehension and Question Answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714, 2018. 7.9

[175] Yining Wang, Sivaraman Balakrishnan, and Aarti Singh. Optimization of smooth func-

tions with noisy observations: Local minimax rates. In *Advances in Neural Information Processing Systems*, pages 4338–4349, 2018. 4.3

[176] Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. Multi-Passage Machine Reading Comprehension with Cross-Passage Answer Verification. *arXiv preprint arXiv:1805.02220*, 2018. 7.4.3, 7.4

[177] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making Neural QA as Simple as Possible but not Simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, 2017. 7.3, 7.4

[178] Rebecca Willett, Robert Nowak, and Rui M Castro. Faster rates in regression via active learning. In *Advances in Neural Information Processing Systems*, pages 179–186, 2006. 3.2

[179] Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`. 8.2.1

[180] Christian Wirth and Johannes Fürnkranz. First steps towards learning from game annotations. 2012. 6

[181] Christian Wirth and Johannes Fürnkranz. EPMC: Every visit preference Monte Carlo for reinforcement learning. In *Asian Conference on Machine Learning*, pages 483–497, 2013. 6.5

[182] Christian Wirth and Johannes Fürnkranz. On learning from game annotations. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(3):304–316, 2015. 6

[183] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017. 6, 6.1

[184] Shifeng Xiong, Peter Z G Qian, and C F Jeff Wu. Sequential Design and Analysis of High-Accuracy and Low-Accuracy Computer Codes. *Technometrics*, 55(1):37–46, 2013. doi: 10.1080/00401706.2012.723572. URL `https://doi.org/10.1080/00401706.2012.723572`. 4.5.6

[185] Y. Xu, H. Muthakana, S. Balakrishnan, A. Dubrawski, and A. Singh. Nonparametric regression with comparisons: Escaping the curse of dimensionality with ordinal information. In *35th International Conference on Machine Learning, ICML 2018*, volume 12, 2018. ISBN 9781510867963. 1.1.1, 4.3, 5.1, 5.3.2

[186] Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. Dynamic Fusion Networks for Machine Reading Comprehension. *arXiv preprint arXiv:1711.04964*, 2017. 7.1

[187] Yichong Xu, Hongyang Zhang, Kyle Miller, Aarti Singh, and Artur Dubrawski. Noise-tolerant interactive learning using pairwise comparisons. In *Advances in neural information*

processing systems, volume 2017-Decem, pages 2431—-2440, 2017. 1.1.1, 3.6, 4.3, 5.1, 5.3.2

[188] Yichong Xu, Sivaraman Balakrishnan, Aarti Singh, and Artur Dubrawski. Interactive Linear Regression with Pairwise Comparisons. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, volume 2018-Octob, pages 636–640. IEEE, 2018. ISBN 9781538692189. doi: 10.1109/ACSSC.2018.8645081. 1.1.1

[189] Yichong Xu, Aparna Joshi, Aarti Singh, and Artur Dubrawski. Zeroth Order Non-convex optimization with Dueling-Choice Bandits. *Proceedings of UAI*, 2019. 1.1.2, 6.1

[190] Yichong Xu, Xiaodong Liu, Chunyuan Li, Hoifung Poon, and Jianfeng Gao. DoubleTransfer at MEDIQA 2019: Multi-Source Transfer Learning for Natural Language Understanding in the Medical Domain. *arXiv preprint arXiv:1906.04382*, 2019. 1.1.3

[191] Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension. In *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2644–2655, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N19-1271`. 1.1.3, 8, 8.1, 8.2.1, 8.3.3

[192] Yichong Xu, Xi Chen, Aarti Singh, and Artur Dubrawski. Thresholding Bandit Problem with Both Duels and Pulls. In *Proceedings of AISTATS*, 2020. 1.1.2

[193] Yichong Xu, Ruosong Wang, Lin F Yang, Aarti Singh, and Artur Dubrawski. Preference-based Reinforcement Learning with Finite-Time Guarantees. *arXiv preprint arXiv:2006.08910*, 2020. 1.1.2

[194] Dezhen Xue, Prasanna V Balachandran, John Hogden, James Theiler, Deqing Xue, and Turab Lookman. Accelerated search for materials with targeted properties by adaptive design. *Nature communications*, 7:11241, 2016. 3, 4.1

[195] Songbai Yan and Chicheng Zhang. Revisiting Perceptron: Efficient and Label-Optimal Active Learning of Halfspaces. *arXiv preprint arXiv:1702.05581*, 2017. 2.6, 2.9.4, 3.1

[196] Liu Yang and Jaime G Carbonell. Cost complexity of proactive learning via a reduction to realizable active learning. Technical report, CMU-ML-09-113, 2009. 2

[197] Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*, 2016. 7.5, 7.9

[198] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *arXiv preprint arXiv:1804.09541*, 2018. 7, 7.1, 7.6

[199] 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. 1.1.2, 4.3, 4.4.2, 1, 4.4.3, 50, 6.1, 6.2, 6.2, 6.2.1, 6.4.1

[200] 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. 1.1.2, 4.1, 4.5, 5.2, 6.1, 6.2

[201] Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *arXiv preprint arXiv:1901.00210*, 2019. 6, 6.4, 6.4.1, 6.7.4, 76

[202] Chicheng Zhang and Kamalika Chaudhuri. Beyond disagreement-based agnostic active learning. In *Advances in Neural Information Processing Systems*, pages 442–450, 2014. 2.2

[203] Cun-Hui Zhang. Risk bounds in isotonic regression. *The Annals of Statistics*, 30(2): 528–555, 2002. 3.2, 3.8.1, 3.8.3, 38

[204] Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4): 1422–1433, 2011. 6

[205] Yuan Zhou, Xi Chen, and Jian Li. Optimal {PAC} Multiple Arm Identification with Applications to Crowdsourcing. In *In Proceedings of the International Conference on Machine Learning (ICML)*, 2014. 5

[206] Masrour Zoghi, Zohar S Karnin, Shimon Whiteson, and Maarten De Rijke. Copeland dueling bandits. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 307–315. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/6023-copeland-dueling-bandits.pdf`. 4.1, 4.3, 4.5.2

[207] James Y Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. Crowdsourcing Feature Discovery via Adaptively Chosen Comparisons. In *Proceedings of the Third {AAAI} Conference on Human Computation and Crowdsourcing, {HCOMP} 2015, November 8-11, 2015, San Diego, California.*, page 198, 2015. 3.2