UNIVERSITY OF CALIFORNIA

Los Angeles

Learning from Human Feedback: Ranking, Bandit, and Preference Optimization

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Yue Wu

2024

ABSTRACT OF THE DISSERTATION

Learning from Human Feedback: Ranking, Bandit, and Preference Optimization

by

Yue Wu
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Quanquan Gu, Chair

This dissertation investigates several challenges in artificial intelligence (AI) alignment and reinforcement learning (RL), particularly focusing on applications when only preference feedback is available. Learning from preference feedback has been one central problem across different fields such as ranking, recommendation systems, and social choice theory. Recently, reinforcement learning from human feedback (RLHF) has also shown its strong potential in utilizing weakly supervised human data (preference feedback) and its ability to encode human values into machine learning models accurately. This dissertation aims to comprehensively characterize preference-based statistical learning, focusing on the sample complexity of ranking and preference model estimation and fine-tuning large language models.

The first part of the dissertation explores novel methods for the learning-to-rank problem. I studied learning to rank under the strong stochastic transitivity (SST) condition, a prevalent model without assuming a score for each option. SST assumes that the accuracy of the comparison between two items increases as the disparity in their quality widens. I proposed one of the first adaptive approaches that can effectively aggregate the feedback from different human labelers and illustrated how the relationship between the number of human queries and resulting performance depends on the properties of the human labelers. I further follow up in this direction with my collaborators and provide active ranking algorithms that can work without strong stochastic transitivity. We developed novel algorithms under this practical yet harder setting. Our efficient algorithm requires fewer human queries compared with algorithms designed with stronger assumptions. The algorithm is provably optimal.

The second part focuses on preference learning without transitivity assumption. In reality, humans rarely make consistent comparisons and often demonstrate contradicting preferences such as a loop within the preference relations. I considered the most general setting where there is no transitivity at all. I proposed algorithms that identify the Borda winner, an optimal choice even when a true underlying rank does not exist. I showed the algorithm enjoys minimum regret, a notion that trades between exploration and exploitation. This result sheds light on the fundamental difficulty and cost of recovering human preferences under the fewest assumptions.

The third part also focuses on preference learning without transitivity assumption, but instead considers an alternative definition of learning objective, the von Neumann winner. I first formulate the general preference as a game environment where two players aim to win over each other and then present an algorithmic framework that can solve this game in a self-play manner asymptotically. The algorithm is then extended to the task of fine-tuning large language models and shows remarkable empirical performance.

The methods and techniques discussed in this dissertation cover a full spectrum of different assumptions and settings of preference learning. In each setting, the new algorithms are presented along with theoretical analysis ensuring a tight performance guarantee. Additionally, during the exploration of different settings, new research directions and open questions are identified, which could help promote the research of preference learning in terms of sample efficiency in the future.

The dissertation of Yue Wu is approved.

University of California, Los Angeles

2024

To my parents

who saw to it that I learned to program

when I was in elementary school

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Quanquan Gu, for his unwavering support and guidance throughout my Ph.D. journey. One moment that stands out is from our very first meeting at UCLA, when he told me (as I roughly recall), "You must believe that you are no less than anyone else. In fact, you can achieve even greater things." That scene still resonates with me to this day.

He has always been responsive and ready to offer thoughtful technical advice. Beyond that, he has mentored me in leadership and communication, shaping my skills in ways I hadn't anticipated. In many ways, he is a role model, exemplifying a remarkable sense of drive and determination, both in his personal achievements and in leading our team. Under his guidance, I have had the opportunity to explore research topics that I might never have ventured into on my own, such as reinforcement learning and, more recently, generative models like diffusion models and large language models, all of which I genuinely enjoy and find deeply fulfilling.

I am deeply thankful to the members of my doctoral committee—Mengdi Wang, Aditya Grover, Lieven Vandenberghe, Guy Van den Broeck, and Quanquan Gu—for their insightful suggestions and valuable feedback on my research and dissertation.

I would like to extend my heartfelt gratitude to Farzad Farnoud for his invaluable support and guidance throughout our collaborative projects on rank aggregation and dueling bandits, which constitute a significant part of this dissertation. I am also deeply grateful to Yuanzhi Li, Yiming Yang, and Dingxuan Zhou for their insights and suggestions during our collaborations. Furthermore, I want to express my appreciation to Yuan Cao, Pan Xu, and Dongruo Zhou, who were senior Ph.D. students or postdoctoral researchers in the lab, for their mentorship during the early stage of my Ph.D. journey.

I am immensely grateful to Wei Cheng at NEC Labs America; Yiqun Wang, Yuning Shen, and Lihao Wang at Bytedance AI Labs; and Bo Xiong, Rui Hou, and Liang Tan at Meta

Generative AI for their mentorship, support, and suggestions during my internships. These industry experiences have significantly broadened my understanding and vision of real-world research and applications.

Throughout my Ph.D. journey, I have been fortunate to collaborate with many talented individuals, both within and outside of Quanquan's research team. My deepest gratitude goes to my incredible team members: Yuan Cao, Jinghui Chen, Zixiang Chen, Qiwei Di, Zhiying Fang, Jiafan He, Kaixuan Ji, Xuheng Li, Lingxiao Wang, Pan Xu, Huizhuo Yuan, Weitong Zhang, Heyang Zhao, Dongruo Zhou, and Difan Zou.

I also want to acknowledge the extraordinary efforts of my fellow collaborators: Yuan Cao, Zixiang Chen, Yihe Deng, Qiwei Di, Jiafan He, Kaixuan Ji, Tao Jin, Hao Lou, Zhiqing Sun, Yan Wang, Pan Xu, Xianjun Yang, Huizhuo Yuan, Shuaicheng Zhang, Weitong Zhang, Heyang Zhao, and Dongruo Zhou. Without your contributions, these projects would not have reached their current high standard.

Furthermore, I would like to express my appreciation to a group of friends not previously mentioned. I am grateful to Changzhi Xie, Yuxiang Peng, Runzhou Tao, and others not named here, for their support throughout my five-year Ph.D. journey. I truly cherish the time we spent together, even when it was online.

Finally, I'd like to extend my deepest gratitude to my parents, Jun Wu and Xuemei Ding, for their understanding and unconditional support throughout my life. I am also profoundly thankful to Minni Sun for her love and care. I am incredibly fortunate to have you by my side throughout this journey.

VITA

2015–2019	Bachelor of Science in Machine Intelligence, Peking University
2019–2024	Research Assistant, Computer Science Department, University of California, Los Angeles
2020–2023	Teaching Assistant, Computer Science Department, University of California, Los Angeles

PUBLICATIONS

Publications most related to this dissertation

1. **Yue Wu**, Tao Jin, Hao Lou, Pan Xu, Farzad Farnoud, and Quanquan Gu. 2021. Adaptive sampling for heterogeneous rank aggregation from noisy pairwise comparisons. International Conference on Artificial Intelligence and Statistics (2021), PMLR, 11014–11036. **Presented in Chapter 2.**

2. Hao Lou, Tao Jin, **Yue Wu**, Pan Xu, Farzad Farnoud, and Quanquan Gu. 2022. Active Ranking without Strong Stochastic Transitivity. Advances in Neural Information Processing Systems (2022), 35, 297-309. **Presented in Chapter 3.**

3. **Yue Wu**, Tao Jin, Qiwei Di, Hao Lou, Farzad Farnoud, and Quanquan Gu. 2024. Borda Regret Minimization for Generalized Linear Dueling Bandits. Proceedings of the 41st International Conference on Machine Learning (2024), PMLR. **Presented in Chapter 4.**

4. **Yue Wu**, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-Play Preference Optimization for Language Model Alignment. ICML 2024 Workshop on Theoretical Foundations of Foundation Models. **Presented in Chapter 5.**

Other core publications

1. **Yue Wu**, Weitong Zhang, Pan Xu, and Quanquan Gu. 2020. A finite-time analysis of two time-scale actor-critic methods. Advances in Neural Information Processing Systems (2020).
2. **Yue Wu**, Dongruo Zhou, and Quanquan Gu. 2022. Nearly minimax optimal regret for learning infinite-horizon average-reward mdps with linear function approximation. International Conference on Artificial Intelligence and Statistics (2022).
3. **Yue Wu**, Shuaicheng Zhang, Wenchao Yu, Yanchi Liu, Quanquan Gu, Dawei Zhou, Haifeng Chen, and Wei Cheng. 2023. Personalized Federated Learning under Mixture of Distributions. Proceedings of the 40th International Conference on Machine Learning (2023), PMLR.
4. **Yue Wu**, Jiafan He, and Quanquan Gu. 2023. Uniform-PAC Guarantees for Model-Based RL with Bounded Eluder Dimension. Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence (2023), PMLR, 2304-2313.
5. Qiwei Di, Tao Jin, **Yue Wu**, Heyang Zhao, and Farzad Farnoud. 2023. Variance-Aware Regret Bounds for Stochastic Contextual Dueling Bandits. International Conference on Learning Representations (2024).

CHAPTER 1

Introduction

Reinforcement learning (RL) is a subfield of artificial intelligence and machine learning that focuses on teaching agents to make decisions in an environment by performing actions and receiving rewards. The goal of RL is to learn a policy that maps states to actions, maximizing the cumulative reward over time. The learning process in RL involves trial and error, as the agent receives feedback in the form of rewards and uses it to adjust its strategy accordingly. Due to its solid theoretical foundations, RL has been applied to various tasks, ranging from game playing to robotics. It has recently proven to be a powerful tool for solving diverse problems in the industrial and medical domains.

However, a significant limitation of the standard RL is that its success crucially depends on the reward function, the definition of which encodes the prior knowledge of the problem. The learned policy may be sensitive to small changes in the reward, possibly yielding very different behaviors, depending on the relative values of the rewards. Also, in many real-world applications such as drug discovery or even the training of chatbots (e.g. ChatGPT), reward functions might not be readily available or easy to design.

Preference-based Reinforcement Learning (PbRL), also known as Reinforcement Learning with Human Feedback (RLHF), is a paradigm for learning from non-numerical feedback in sequential domains. PbRL utilizes binary preference feedback instead of absolute utility values, making it a more natural and straightforward approach for many RL applications, especially those involving human evaluations.

Although several algorithms have been developed for PbRL, their performance guarantees are not well understood, and the existing theoretical guarantees are limited. This dissertation focuses on developing efficient and scalable RL algorithms that use preference feedback from humans and understanding both the capability and the fundamental limits of these preference-based learning algorithms. Specifically, I will study the statistical limits of PbRL algorithms, which describe the necessary number of samples for an agent to learn a near-optimal policy.

The first aspect of preference learning we would like to address is efficient learning-to-rank from preference feedback. When the preference is deterministic, the ranking problem is well-studied and can be solved by famous algorithms such as Selection Sort, Merge Sort, or Quick Sort. The solution is less straightforward when the feedback is assumed noisy, meaning that comparing two items can receive different outcomes with randomness. It is imaginable that in such cases, the algorithm needs to query multiple times and try to estimate the ground-truth ranking from the feedback it has received. With a decent estimation, the algorithm can adjust its further queries to recover the ranking more efficiently. This is the learning-to-rank problem.

In **Chapter** 2, we study such a learning-to-rank problem, but with multiple preference oracles. This setting can be instantiated as a recommendation system: there are N items and M customers, each customer can provide preference feedback, and the goal of the system is to provide a ranking based on the (noisy) preference feedback. In this chapter, we seek to provide rigorous analysis and algorithmic design to solve this problem in a restricted setting, where all users agree on an underlying true ranking, but each user can be accurate, noisy, or even adversary, indicated by an unknown accuracy level. We design asymptotically optimal algorithms that can identify the accurate users gradually throughout the process and invest the query budget to the accurate users later. In **Chapter** 3, we further relax the strong stochastic transitivity to the weak stochastic transitivity and study the learning-to-rank problem with a single oracle under this setting. We propose a δ-correct algorithm, Probe-Rank, that actively learns the ranking from noisy pairwise comparisons. We prove a sample complexity upper bound for Probe-Rank, which only depends on the preference probabilities between items that are adjacent in the true ranking. This improves upon existing sample complexity results that depend on the preference probabilities for all pairs of items. Probe-Rank thus outperforms existing methods over many instances that do not satisfy Strong Stochastic Transitivity. Thorough numerical experiments in various settings are conducted, demonstrating that Probe-Rank is significantly more sample-efficient than the state-of-the-art active ranking method.

The learning-to-rank problem discussed above has an implicit assumption that the preference is transitive: if A is preferred over B and B is preferred over C, then A is preferred over C ("preferred" is in the probabilistic sense). In fact, this is usually not the case when human behavior is involved. One experiment on human behavior of gambling (Tversky, 1969) shows that "people chose between adjacent gambles according to the payoff and between the more extreme gambles according to probability or expected value". This shows humans often demonstrate contradicting preferences such as a loop within the preference relations. In **Chapter** 4 and **Chapter** 5, we will explore rich and general preference models, In **Chapter** 4, we consider the most general setting where the preference is determined by some rich context, and proposed algorithms that identify the Borda winner, an optimal choice even when a true underlying rank list does not exist. I showed the algorithm enjoys minimum regret, a notion that trades between exploration and exploitation. This result sheds light on the fundamental difficulty and cost of recovering human preferences under the fewest assumptions. In **Chapter** 5, we consider an alternative setting that allows general preference, the von Neumann winner. I first formulate the general preference as a game environment where two players aim to win over each other and then present an algorithmic framework that can solve this game in a self-play manner asymptotically. The algorithm is then extended to the task of fine-tuning large language models and shows remarkable empirical performance.

1.1 Organization of the Dissertation

The rest of this dissertation is organized as follows. In **Chapter** 2, we introduce the learning-to-rank problem and the setting of the rank aggregation problem. We then propose algorithms that can adaptively allocate query budget and recover the true ranking efficiently. In **Chapter** 3, we study the learning-to-rank problem under the Weak Stochastic Transitivity assumption and propose algorithms that actively learn the ranking from noisy pairwise comparisons and outperform existing methods over a large collection of instances that do not satisfy Strong Stochastic Transitivity. In **Chapter** 4, we move on to the topic of learning under general preference. We first introduce a new problem setting for identifying the Borda winner and propose new algorithms that can efficiently identify the Borda winner. We are also able to deliver matching upper bound and lower bound of the Borda regret for the proposed algorithm. In **Chapter** 5, we formulate another learning objective for learning under general preference. The learning objective is to identify the Nash equilibrium policy of a predefined two-player constant-sum gam. We adopt the classic online adaptive algorithm with multiplicative weights (Freund and Schapire, 1999) as a high-level framework that solves the two-player game and propose efficient self-play algorithms that show remarkable empirical performance when applied to fine-tuning large language models.

1.2 Notation System in this Dissertation

In this dissertation, scalars are denoted by lowercase letters. Vectors are denoted by lowercase boldface letters $\mathbf{x}$, and matrices by uppercase boldface letters $\mathbf{A}$. We denote by $[k]$ the set $\{1, 2, \cdots, k\}$ for positive integers k. We use $\log x$ to denote the logarithm of x to the base 2. For two nonnegative sequences $\{a_n\}, \{b_n\}$, $a_n \leqslant \mathcal{O}(b_n)$ means that there exists a positive constant C such that $a_n \leqslant Cb_n$. Notation $a_n \leqslant \widetilde{\mathcal{O}}(b_n)$ means that there exists a positive constant k such that $a_n \leqslant \mathcal{O}(b_n \log^k b_n)$. Notation $a_n \geqslant \Omega(b_n)$ means that there exists a positive constant C such that $a_n \geqslant Cb_n$. Notation $a_n \geqslant \widetilde{\Omega}(b_n)$ means there exists a positive constant k such that $a_n \geqslant \Omega(b_n \log^{-k} b_n)$. Notation $a_n \geqslant \omega(b_n)$ means that $\lim_{n\to\infty} b_n/a_n = 0$. The notations like $\widetilde{O}$ and $\widetilde{\Omega}$ are used to hide logarithmic factors. For a vector $\mathbf{x} \in \mathbb{R}^d$ and a positive semidefinite matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we define $\|\mathbf{x}\|_{\mathbf{A}}^2 = \mathbf{x}^\top \mathbf{A}\mathbf{x}$. For any set $\mathcal{C}$, we use $|\mathcal{C}|$ to denote its cardinality. We denote the identity matrix by $\mathbf{I}$ and the empty set by $\varnothing$. The remaining notations are defined before they are used in each chapter.

CHAPTER 2

Active Ranking from Heterogeneous Pairwise Comparisons

2.1 Introduction

To rank a set of items from noisy pairwise comparisons or preferences is a widely studied topic in machine learning (Braverman and Mossel, 2008; Weng and Lin, 2011; Ren et al., 2019; Jin et al., 2020). This is also referred to as rank aggregation, which has many applications in practice such as ranking online game players (Herbrich et al., 2006a), evaluating agents in games (Rowland et al., 2019), recommendation systems (Valcarce et al., 2017), etc.

In the above cases, all data used in inference shares the assumption that each comparison has the same credibility. However, in a heterogeneous setting, the providers of subsets of data may have varying unknown accuracy levels. Thus, it is natural to take advantage of the more accurate ones to obtain a more accurate ranking using a smaller number of queries.

Nowadays, it is common to collect large-scale datasets to facilitate the process of knowledge discovery. Due to its scale, such data collection is usually carried out by crowdsourcing (Kumar and Lease, 2011; Chen et al., 2013), where different entities with diverse backgrounds generate subsets of the data. While crowdsourcing makes it possible to scale up the size, it also brings new challenges when it comes to the cost of operation and cleanness of the data. For example, the optimal ranking algorithm in the single-user setting (Ren et al., 2019) may not be straightforwardly extended to the heterogeneous setting while maintaining optimality. In particular, if we know the most accurate user among the set of users providing comparisons, the best we can do is to apply optimal single-user ranking algorithms such as Iterative-Insertion-Ranking (IIR) (Ren et al., 2019) by querying only the most accurate user. Unfortunately, in practice, the accuracies of the users are often unknown. A naive solution may be to randomly select a user to query and use the comparisons provided by this user to insert an item into the ranked list per IIR. However, as we show later, this naive method usually bears a high sample complexity. Therefore, it is of great interest to design methods that can adaptively select a subset of users at each time to query pairwise comparisons to insert an item correctly into the ranked list.

In this chapter, we study the rank aggregation problem, where a heterogeneous set of users provides noisy pairwise comparisons for the items. We propose a novel algorithm that queries comparisons for pairs of items from a changing active user set. Specifically, we maintain a short history of user responses for a set of comparisons. When the inferred rank of these comparisons is estimated to be true with high confidence, it is then used to calculate a reward based on the recorded responses. Then an upper confidence bound (UCB)-style elimination process is performed to remove inaccurate users from the active user set. We theoretically analyze the sample complexity of the proposed algorithm, which reduces to the state-of-the-art ranking algorithm (Ren et al., 2019) for a single user. We conducted experiments on both synthetic and real-world datasets, which demonstrate that our adaptive sampling algorithm based on user elimination is much more sample efficient than baseline algorithms and can sometimes reach the performance of an oracle algorithm.

2.1.1 Organization of this Chapter

In this chapter, we propose a novel algorithm called Ada-IIR for heterogeneous rank aggregation, which uses a successive elimination subroutine to adaptively maintain a set of active users during the ranking process. The remainder of this chapter is organized as follows: we review the most relevant work in the literature to ours in Section 2.2. We present the preliminaries of ranking from noisy pairwise comparisons in Section 2.3 and discuss the challenge of extending single-user optimal ranking algorithms to the heterogeneous setting. In Section 2.4, we present our main algorithm and provide a detailed description of the method. Then in Section 2.5, we provide a theoretical analysis of the upper bound on the sample complexity of the proposed algorithm and compare the results with baseline algorithms. We conduct numerical experiments in Section 2.7 to demonstrate the empirical superiority of our method. Finally, we conclude the paper with Section 2.8. We defer the detailed proof of the theorems to Section 2.11.

2.2 Related Work

In this section, we discuss two closely related topics to our work, which cover the two facets of heterogeneous rank aggregation: active ranking to infer the rank and best arm identification to select a subset of accurate information sources (e.g., users). In addition to this, we also introduce similar work bearing the idea that ranking information can be heterogeneous.

Active ranking For passive ranking problems, a static dataset is given beforehand. Inference of the ranking often relies on models of ranked data, such as the Bradley-Terry-Luce (BTL) model (Bradley and Terry, 1952) and the Thurstone model (Thurstone, 1927b). In contrast to passive algorithms, active algorithms leverage assumptions embedded in the models to identify the most informative pairs to query, thus reducing the sample complexity of queries. For instance, in Maystre and Grossglauser (2017), under the assumption that the true scores for n items are generated by a Poisson process, with $O(n\text{poly}(\log(n))$ comparisons, an *approximate* ranking of n items can be found. Let the probability of making a correct comparison between item i and the most similar item to item i be $\frac{1}{2} + \Delta_i$ and let $\Delta_{\min} = \min_{i\in[n]} \Delta_i$. An instance-dependent sample complexity bound of $O(n \log(n)\Delta_{\min}^{-2} \log(n/(\delta\Delta_{\min}))$ is provided along with a QuickSort based algorithm by Szörényi et al. (2015). In Ren et al. (2019), an analysis for a distribution-agnostic active ranking scheme is provided. To achieve a δ-correct *exact* ranking, $O(\sum_{i\in[n]} \Delta_i^{-2}(\log\log(\Delta_i^{-1}) + \log(n/\delta)))$ comparisons are required. The exact inference requirement results in repeated queries of the same pair, which costs a constant overhead compared to approximate inference.

Best Arm Identification (BAI) BAI is a pure exploration method in multi-armed bandits (Audibert et al., 2010; Chen et al., 2017). In the crowdsourcing setting, every user can be queried with the same question. Noting that some users can provide more accurate answers than others, the goal is to identify the best user. We can regard the choice of which user to ask as an action and the correctness of the user's response as the reward (cost) of the taken action. A long line of research has explored the identification of the best action with stochastic feedback. Recently, Resler and Mansour (2019) studied cases when the observed binary action costs can be inverted with a probability that is less than half. With careful construction of the estimated cost despite the noise, the regret of the online algorithm suffers a constant order compared to the noiseless setting even without the knowledge of the inversion probability.

Heterogeneous Rank Aggregation An early work from Chen et al. (2013) explored the idea of user-specific accuracy through a model that is equivalent to adding noise to the comparisons produced by the BTL model. More recently, Jin et al. (2020) proposed a natural extension of BTL and Thurstone generative models to a heterogeneous population of users for pairwise comparisons and partial rankings. In addition to this line of work that assumes a global true ranking, mixture models (Zhao and Xia, 2019) were proposed for personal preference inference. These works output high-accuracy approximate rankings.

2.3 Preliminaries

2.3.1 Ranking from Noisy Pairwise Comparisons

Suppose there are N items that we want to rank and M users to be queried. An item is indexed by an integer $i \in [N]$. We assume there is a unique *true ranking* of the N items. A user is also indexed by an integer $u \in [M]$. For a subset of users, we use $\mathcal{U} \subseteq [M]$ to denote the index subset. In each time step, we can pick a pair of items i and j and ask a user u whether item i is better than item j. The comparison returned by the user may be noisy. We assume that for any pair of items (i, j) with true ranking $i \succ j$, the probability that user u answers the query correctly is $p_u(i, j) = \Delta_u + 1/2$, where $\Delta_u \in (0, \frac{1}{2}]$ is referred to as the accuracy level of user u. When some of the Δ_u's are different from the others, we call the set of users *heterogeneous*. We assume comparison results for item pairs, regardless of the queried user, are mutually independent. While this independence assumption may not always hold for real datasets, it is commonly adopted in the literature as it facilitates the analysis (Falahatgar et al., 2017b, 2018; Jin et al., 2020).

In this paper, we aim to achieve the exact ranking for a ranking problem defined as follows.

Definition 2.3.1 (Exact Ranking with Multiple Users)**.** Given N items, M users, and $\delta \in (0, 1)$, our goal is to identify the true ranking among the N items with a probability of at least $1 - \delta$. An algorithm $\mathcal{A}$ is δ-correct if, for any instance of the input, it will return the correct result in finite time with probability at least $1 - \delta$.

To actively eliminate the users in the user pool, we define an α-optimal user as follows.

Definition 2.3.2. Let $\mathcal{U} \subseteq [M]$ be an arbitrary subset of users. If a user $x \in \mathcal{U}$ satisfies $\Delta_x + \alpha \geqslant \max_{u \in \mathcal{U}} \Delta_u$, then x is called an *α-optimal user in $\mathcal{U}$*. If a user is α-optimal among all M users, then it is called a (global) α-optimal user.

2.3.2 Iterative Insertion Ranking with a Single User

When there is only one user u to be queried ($M = 1$), the problem defined in Section 2.3.1 reduces to the exact ranking problem with a single user, for which Ren et al. (2019) proposed the Iterative-Insertion-Ranking (IIR) algorithm. The sample complexity (i.e., the total number of queries) to achieve exact ranking with probability $1 - \delta$ is characterized by the following proposition:

Proposition 2.3.3 (Adapted from Theorems 2 and 12 in Ren et al. (2019))**.** Given $\delta \in (0, 1/12)$ and an instance of N items, the number of comparisons used by any δ-correct algorithm $\mathcal{A}$ on this instance is at least

$$\Theta\big(N\Delta_u^{-2}\big(\log\log\Delta_u^{-1} + \log(N/\delta)\big)\big). \tag{2.3.1}$$

Moreover, the IIR algorithm proposed by Ren et al. (2019) can output the exact ranking using this number of comparisons, with probability $1 - \delta$.

The complexity above can be decomposed into the complexity of inserting each item into a constructed sorting tree.

In this paper, we consider a more challenging ranking problem, where multiple users with heterogeneous levels of accuracy can be queried each time. In the multi-user setting, the optimal sample complexity in (2.3.1) can be achieved only if we know which user is the best, i.e., $u^* = \arg\max_{u\in[M]} \Delta_u$. The optimal sample complexity can then be written as

$$\mathcal{C}_{u^*}(N) = \Theta\big(N\Delta_{u^*}^{-2}\big(\log\log\Delta_{u^*}^{-1} + \log(N/\delta)\big)\big). \tag{2.3.2}$$

However, with no prior information on the users' comparison accuracies, it is unclear whether we can achieve a sample complexity close to (2.3.2). In this scenario, the most primitive route is to perform no inference on the users' accuracy and randomly choose users to query.

This leads to an equivalent accuracy of $\bar{\Delta}_0 = \frac{1}{M}\sum_{u\in[M]}\Delta_u$ and a sample complexity given as

$$\mathcal{C}_{\text{ave}}(N) = \Theta\big(N\bar{\Delta}_0^{-2}\big(\log\log\bar{\Delta}_0^{-1} + \log(N/\delta)\big)\big). \tag{2.3.3}$$

Compared with the best possible complexity (2.3.2), the sample complexity (2.3.3) is larger by a factor (ignoring logarithmic factors) up to M^2, because the ratio between Δ_{u*} and $\bar{\Delta}_0$ could vary a lot for different set of users and can be as large as M. This is certainly undesirable, especially when there are a large number of items to be ranked. Therefore, an immediate question is: Can we design an algorithm that has a smaller multiplicative factor in its sample complexity compared with the optimal sample complexity? What we will propose in the following section is an algorithm that can achieve a sublinear regret, where the regret is defined as the difference between the sample complexity of the proposed algorithm and the optimal sample complexity.

2.4 Adaptive Sampling and User Elimination

The main framework of our procedure is derived based on the Iterative-Insertion-Ranking algorithm proposed in Ren et al. (2019), which, to the best of our knowledge, is the first algorithm that has matching instance-dependent upper and lower sample complexity bounds for active ranking problems in the single-user setting. We assume that the strong stochastic transitivity (SST) assumption defined in Falahatgar et al. (2017b, 2018) holds in our setting. The ranking algorithm comprises the following four hierarchical parts and operates on a Preference Interval Tree (PIT) (Feige et al., 1994a; Ren et al., 2019), which stores the currently inserted and sorted items (the detailed definition is presented in Section 2.9):

1. Adaptive Iterative-Insertion-Ranking (Ada-IIR): the main procedure that calls IAI to insert an item into a PIT with a high probability of correctness. It is displayed in Algorithm 1.

2. Iterative-Attempting-Insertion (IAI): the subroutine which calls ATI to insert the current item $z \in [N]$ into the ranked list with an error ϵ, and iteratively calls ATI by decreasing the error until the probability that item z is inserted to the correct position is high enough. It is displayed in Algorithm 5.

3. Attempting-Insertion (ATI): the subroutine that traverses the Preference Interval Tree using binary search (Feige et al., 1994b) to find the node where the item should be inserted with error ϵ. To compare the current item and any node in the tree, it calls ATC to obtain the comparison result. It is displayed in Algorithm 6.

4. Attempting-Comparison (ATC): the subroutine that adaptively samples queries from a subset of users for a pair of items (z, j), where z is the item currently being inserted and j is any other item. ATC records the number of queries each user provides and the results of the comparisons. It is displayed in Algorithm 2.

In the heterogeneous rank aggregation problem, each user may have a different accuracy level from the others. Therefore, we adaptively sample the comparison data from a subset of users. In particular, we maintain an active set $\mathcal{U} \subseteq [M]$ of users, which contains the potentially most accurate users from the entire group. We add a user elimination phase to the main procedure (Algorithm 1) based on the elimination idea in multi-armed bandits (Slivkins et al., 2019; Lattimore and Szepesvári, 2020) to update this active set. In particular, we view each user as an arm in a multi-armed bandit, where the reward is 1 if the answer from a certain user is correct and 0 if wrong. After an item is successfully inserted by IAI, we call Algorithm 3 (EliminateUser) to eliminate users with low accuracy levels before we proceed to the next item.

To estimate the accuracy levels of users, a vector $\mathbf{s}_z \in \mathbb{R}^M$, recording the counts of responses from each user for item z is maintained during the whole period of inserting item z. We further keep track of two matrices $A_z, B_z \in \mathbb{R}^{N \times M}$. When a pair (z, j) (where z refers to the item currently being inserted and j to an arbitrary item) is compared by user $u \in \mathbb{R}^M$

in Algorithm 2, we increase $A[j,u]$ by 1 if user u thinks z is better than j and increase $B[j,u]$ by 1 otherwise. We use w to record the total number of times that item z is deemed better by any users and use the average $\widehat{p} = w/t$ to provide an estimation of the average accuracy $|\mathcal{U}|^{-1} \sum_{u \in \mathcal{U}} p_{ij}^{u}$. The variables A_z, B_z, and $\mathbf{s}_z$ are global variables, shared by different subroutines throughout the process. After an item z is successfully inserted, A_z, B_z will be discarded and the space allocated can be used for A_{z+1}, B_{z+1} (See Line 3 of Algorithm 1).

We use the 0/1 reward for each user to indicate whether the provided pairwise comparison is correct. Nevertheless, this reward is not known immediately after each arm-pull since the correctness depends on the ranking of items which is also unknown. But when IAI returns *inserted*, the item recently inserted has a high probability of being in the right place. Our method takes advantage of this fact by constructing a fairly accurate prediction of pairwise comparison for the item with all other already inserted items in the PIT. Then an estimate of the reward $\boldsymbol{n}_z$ can be obtained with the help of recorded responses A_z and B_z, which are updated in ATC as described in the preceding paragraph. At last, in Algorithm 3 a UCB-style condition is imposed on estimated accuracy levels $\boldsymbol{\mu} = \boldsymbol{n}_z / \boldsymbol{s}_z$.

Due to the space limit, we omit here the IAI and ATI routines that are proposed in Ren et al. (2019). We include them for completeness and ease of reference in Section 2.9.

2.5 Theoretical Analysis

In this section, we analyze the sample complexity of the proposed algorithm and compare it with other baselines mentioned in Section 2.3.

Algorithm 1 Main Procedure: ADAPTIVE ITERATIVE-INSERTION-RANKING (Ada-IIR)

Global Variables:

$z \in \mathbb{N}$: the index of the item being inserted into the ranked list.

$A_z \in \mathbb{R}^{N\times M}$: $A_z[j,u]$ is the number of times that user u thinks item z is better than item j.

$B_z \in \mathbb{R}^{N\times M}$: $B_z[j,u]$ is the number of times that user u thinks item z is worse than item j.

$\mathbf{s}_z \in \mathbb{R}^M$: total number of responses by each user so far.

Input parameters: Items to rank $S = [N]$ and confidence δ

Initialize: $\boldsymbol{n}_1 = \boldsymbol{s}_1 = \mathbf{0}$

```
1: $Ans \leftarrow$ the list containing only $S[1]$
2: for $z \leftarrow 2$ to $|S|$ do
3:     $\boldsymbol{n}_z = \boldsymbol{n}_{z-1}, \boldsymbol{s}_z = \boldsymbol{s}_{z-1}, A_z = \mathbf{0}, B_z = \mathbf{0}$
4:     IAI$(S[z], Ans, \delta/(n-1))$ ▻Algorithm 5 (global variables $A_z, B_z, \boldsymbol{s}_z$ are updated here)
5:     for $j \in [z-1]$ do
6:         if $S[z] > S[j]$ in PIT then
7:             $\boldsymbol{n}_z = \boldsymbol{n}_z + A_z[j,*]$
8:         else
9:             $\boldsymbol{n}_z = \boldsymbol{n}_z + B_z[j,*]$
10:        end if
11:    end for
12:    $\mathcal{U}_z \leftarrow$ ELIMINATEUSER$(\mathcal{U}_{z-1}, \mathbf{n}_z, \mathbf{s}_z, \delta/(n-1))$     ▻Algorithm 3
13: end for
14: return $Ans$;
```

Algorithm 2 Subroutine: ATTEMPT-TO-COMPARE (ATC) $(z, j, \mathcal{U}, \epsilon, \delta)$

Input: items (z, j) to be compared, set of users $\mathcal{U}$, confidence parameter ϵ, δ. M is the number of users originally.

```
1: $m = |\mathcal{U}|, \widehat{p} = 0, w = 0, \widehat{y} = 1$. Number of rounds $r = 1$. $r_{\max} = \lceil \frac{1}{2}\epsilon^{-2} \log \frac{2}{\delta} \rceil$.
2: while $r \leqslant r_{\max}$ do
3:     Choose $u$ uniformly at random from $\mathcal{U}$
4:     Obtain comparison result from user $u$ as $y_{ij}^u$
5:     Increment the counter of responses collected from this user $\boldsymbol{s}_z[u] \leftarrow \boldsymbol{s}_z[u] + 1$
6:     if $y_{ij}^u > 0$ then
7:         $A_z[j, u] \leftarrow A_z[j, u] + 1$, $w \leftarrow w + 1$
8:     else
9:         $B_z[j, u] \leftarrow B_z[j, u] + 1$
10:    end if
11:    $\widehat{p} \leftarrow w/r$, $r \leftarrow r + 1$, $c_r \leftarrow \sqrt{\frac{1}{2t} \log(\frac{\pi^2 r^2}{3\delta})}$
12:    if $|\widehat{p} - \frac{1}{2}| \geqslant c_r$ then
13:        break
14:    end if
15: end while
16: if $\widehat{p} \leqslant \frac{1}{2}$ then
17:     $\widehat{y} = 0$
18: end if
19: return: $\widehat{y}$
```

2.5.1 Sample Complexity of Algorithm 1

We first present an upper bound on the sample complexity of the proposed algorithm. Define $\bar{\Delta}_z = \frac{1}{\mathcal{U}_z} \sum_{u \in \mathcal{U}_z} \Delta_u$ to be the average accuracy of all users in the current active set. Denote

$$F(x) = x^{-2}(\log \log x^{-1} + \log(N/\delta)). \tag{2.5.1}$$

Algorithm 3 Subroutine: ELIMINATEUSER

```
Input parameters: ($\mathcal{U}$, $\mathbf{n}$, $\mathbf{s}$, $\delta$).
1: Set $S = \sum_{u\in[M]} \mathbf{s}_u$, $\mathbf{s}_{\min} = \min_{u\in\mathcal{U}} \mathbf{s}_u$, $\boldsymbol{\mu}_u = \mathbf{n}_u/\mathbf{s}_u$, $r = \sqrt{\log(2|\mathcal{U}|/\delta)/(2\mathbf{s}_{\min})}$
2: Set $\mathbf{LCB} = \boldsymbol{\mu} - r\mathbf{1}$ and $\mathbf{UCB} = \boldsymbol{\mu} + r\mathbf{1}$.
3: if $S \geqslant 2M^2 \log(NM/\delta)$ then
4:     for $u \in \mathcal{U}$ do
5:         Remove user $u$ from $\mathcal{U}$ if $\exists u' \in \mathcal{U}, \mathbf{UCB}_u < \mathbf{LCB}_{u'}$.
6:     end for
7: end if
8: return $\mathcal{U}$
```

Although $F(x)$ depends on N and δ^{-1}, the dependence is only logarithmic, and it does not affect the validity of reasoning via big-O notations.

Theorem 2.5.1. For any $\delta > 0$, with probability at least $1 - \delta$, Algorithm 1 returns the exact ranking of the N items, and it makes at most $\mathcal{C}_{\text{Alg}}(N)$ queries, where $\mathcal{C}_{\text{Alg}}(N) = O(\sum_{z=2}^{N} \bar{\Delta}_z^{-2}(\log\log \bar{\Delta}_z^{-1} + \log(N/\delta))) = O(\sum_{z=2}^{N} F(\bar{\Delta}_z))$.

Proof. The analysis of the sample complexity follows a similar route as Ren et al. (2019) due to the similarity in algorithm design. Since we randomly choose a user from $\mathcal{U}_t$ and query it for feedback, it is equivalent to querying a single user with the averaged accuracy $\frac{1}{2} + \bar{\Delta}_z$, where $\bar{\Delta}_z := \frac{1}{|\mathcal{U}_z|}\sum_{u\in\mathcal{U}_z} \Delta_u$. This means most of the theoretical results from Ren et al. (2019) can also apply to our algorithm. In Section 2.11.1, we present more detailed reasoning. □

2.5.2 Sample Complexity Comparison of Different Algorithms

While Theorem 2.5.1 characterizes the sample complexity of Algorithm 1 explicitly, the result therein is not directly comparable with the sample complexity of the oracle algorithm that only queries the best user $\mathcal{C}_{u^*}(N)$ or the complexity of the naive random-query algorithm $\mathcal{C}_{\text{ave}}(N)$. Based on Theorem 2.5.1, we can derive the following more elaborate sample complexity for Algorithm 1.

Theorem 2.5.2. Suppose there are N items and M users initially. Denote $S_z = \sum_{u\in[M]}(\mathbf{s}_z)_u$ to be the number of all queries made before inserting item z (Line 4 in Algorithm 1). The proposed algorithm has the following sample complexity upper bound:

$$\begin{aligned}\mathcal{C}_{\mathrm{Alg}}(N,M) = \Theta(NF(\Delta_{u^*})) &+ O\left(\sum_{z=2}^{N}\mathbb{1}\left\{S_z < 2M^2\log(NM/\delta)\right\}\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\right)\\ &+ O\left(L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\mathbb{1}\{S_z \geqslant 2M^2\log(NM/\delta)\}\sqrt{\frac{M}{S_z}}\right), \end{aligned} \tag{2.5.2}$$

where $L(\mathcal{U}_0) = \frac{F(c\Delta_{u^*}^3)-F(\Delta_{u^*})}{\Delta_{u^*}-c\Delta_{u^*}^3}$ is an instance-dependent factor, with only logarithmic dependence on N and δ^{-1}(through F), and where $c = 1/25$ is a global constant.

Proof. The detailed proof can be found in Section 2.11.2. □

A few discussions are necessary to show the meaning of the result. First, if the number of users $M \gg N$, then no user is eliminated because each user will be queried so few times that no meaningful inference can be made. Since the goal is to achieve the accuracy of the best user, more inaccurate users only make the task more difficult. Therefore, it is necessary to impose assumptions on M with respect to N.

This intuition can be made more precise. Suppose we loosely bound S_t as $S_t \geqslant t\log(t/\delta)$, which is reasonable since for a very accurate user the algorithm will spend roughly no more than $O(\log(t/\delta))$ comparisons to insert one item. This means the complexity can be bounded (ignoring log factors)

$$\mathcal{C}_{\mathrm{Alg}}(N,M) = O\big(NF(\Delta_{u^*})\big) + \widetilde{O}\big(M^2\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)\big) + \widetilde{O}\big(L(\mathcal{U}_0)\big(\sqrt{M}(\sqrt{N}-M)\big)\big). \tag{2.5.3}$$

If $M = \Omega(\sqrt{N})$, then this is not ideal because our algorithm won't eliminate any user until $\Omega(N)$ items are inserted with accuracy $\bar{\Delta}_0$, which already leads to a gap linear in N compared with the best complexity $\mathcal{C}_{u^*}$. In this case, our algorithm roughly makes the same amount of queries as $\mathcal{C}_{\text{ave}}$.

To avoid the bad case, it is necessary to assume $M = o(\sqrt{N})$ so that the last two terms become negligible (notice that $L(\mathcal{U}_0)$ is an instance-dependent constant). Now we restate Theorem 2.5.2 with the additional assumption and compare it with the baselines.

Proposition 2.5.3. Suppose we have M users and N items to rank exactly, with $M = o(\sqrt{N})$. We have the following complexity along with (2.3.2) and (2.3.3):

$$\begin{aligned}
\mathcal{C}_{u^*}(N, M) &= \Theta(NF(\Delta_{u^*})),\\
\mathcal{C}_{\text{ave}}(N, M) &= \Theta(NF(\bar{\Delta}_0)),\\
\mathcal{C}_{\text{Alg}}(N, M) &= \Theta(NF(\Delta_{u^*})) + o(N\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big)) + o\big(N\big).
\end{aligned}$$

The last two terms of $\mathcal{C}_{\text{Alg}}(N, M)$ are negligible when compared with the first term. Therefore, our algorithm can perform comparably efficiently as if the best user were known while enjoying an advantage over the naive algorithm with sample complexity $\mathcal{C}_{\text{ave}}(N, M)$.

Remark 2.5.4. Note that if we set $\mathcal{U}_0 = \{u^*\}$ for our algorithm, it will achieve the same complexity as (2.3.2) indicates. Similarly, if we construct a new user $\bar{u}$ where $\Delta_{\bar{u}} = \bar{\Delta}_0$ and set $\mathcal{U}_0 = \{\bar{u}\}$, our algorithm will recover exactly (2.3.3). By this argument and the fact that Big-O notations hide no M, the first term in each equation has the same absolute constant factor. Therefore, our algorithm is indeed comparable with the best user.

Remark 2.5.5. Notice that $F(x) \to +\infty$ when $x \to 0$. This means $\mathcal{C}_{\text{ave}}$ is very sensitive to the initial average accuracy margin $\bar{\Delta}_0$. In the case where there is only one best user u^* and all other users have a near-zero margin $\Delta_u \to 0$, $\mathcal{C}_{\text{ave}}$ can be very large compared with $\mathcal{C}_{u^*}$.

Remark 2.5.6. In the experiments, we notice that even with $N = 10$ and $M = 9$, after inserting the first item, each user has already been queried enough times so that $S_2 \geqslant 2M^2 \log(NM/\delta)$, which makes the second term in (2.5.2) vanish.

2.6 A Two-Stage Algorithm

In this section, we present an alternative simple scheme, called two-stage ranking with a heterogeneous set of users. This provides another baseline with which we can compare Ada-IIR. Additionally, it can be useful in situations with a large number of users, i.e., $M = \Omega(\sqrt{N})$, where Ada-IIR is less effective.

Two-stage ranking first performs user selection and then item ranking. In the user-selection stage, we search for an α-optimal user for some small α. Specifically, we first take an arbitrary pair of items (i, j) and then run the Iterative-Insertion-Ranking (IIR) algorithm (see Proposition 2.3.3) on them to determine the order, e.g., $i \succ j$, with high probability. Note that at this point, users have not been distinguished yet. So we take each query from a randomly chosen user. As discussed in Section 2.3.2, this is equivalent to querying the user $\bar{u}$ whose accuracy is $\bar{\Delta}_0$. Given $i \succ j$, the problem of finding an α-optimal user is reduced to pure exploration of an α-optimal arm in the context of multi-armed bandit: making queries about the pre-determined item pair from user u is the same as generating outcomes from an arm with Bernoulli($\frac{1}{2} + \Delta_u$) reward, e.g., if user u returns the answer $i \succ j$ then we get reward 1, otherwise we get reward 0. Hence, an α-optimal user is equivalent to an α-optimal arm. For determining an α-optimal arm, we can adopt the Median-Elimination (ME) algorithm from Even-Dar et al. (2002a). After ME returns an α-optimal user u_α, we discard all other users and rank items by only querying u_α. Ranking with a single user can again be done by the IIR algorithm.

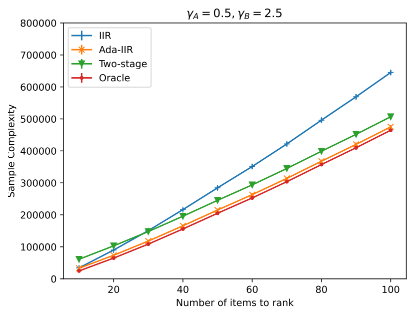
Two-stage ranking is composed of three procedures: IIR for determining the order of i and j, ME for obtaining an α-optimal user, and IIR again for producing the final ranking. The complexity of two-stage ranking is therefore the sum of the complexities of the three procedures.

Theorem 2.6.1. thmtsr For any $\alpha \in (0, \Delta_{u*})$, two-stage ranking outputs the exact ranking with probability at least $1-\delta$ using at most $\mathcal{C}_{\text{tsr}}(N,M)$ comparisons, where

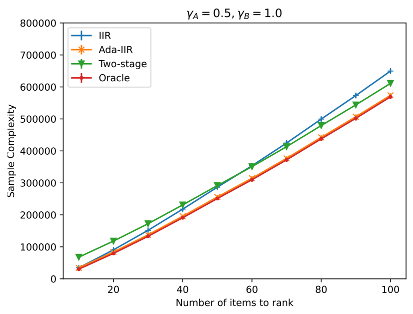
$$\begin{aligned}\mathcal{C}_{\text{tsr}}(N,M) = \Theta\bigg(& \frac{1}{\bar{\Delta}_0^2}\left(\log\log\frac{1}{\bar{\Delta}_0}+\log\frac{1}{\delta}\right)+\frac{M}{\alpha^2}\log\frac{1}{\delta} \\ & +\frac{N}{(\Delta_{u*}-\alpha)^2}\left(\log\log\frac{1}{\Delta_{u*}-\alpha}+\log\frac{N}{\delta}\right)\bigg).\end{aligned}$$

A more formal statement of two-stage ranking as well as the proof of Theorem 2.6.1 are presented in Section 2.10.1.

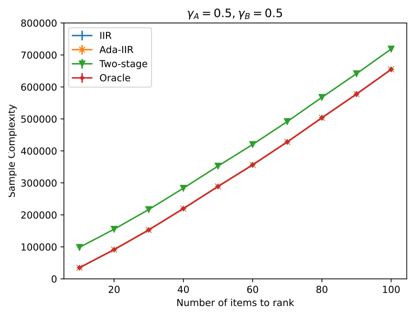
Recall $F(x)=x^{-2}(\log\log x^{-1}+\log(N/\delta))$ defined in (2.5.1). From the preceding theorem, it is clear that for any constant α, when $M=o(N\log N)$, $\mathcal{C}_{\text{tsr}}(N,M)$ is dominated by $\Theta\left(NF\left(\Delta_{u*}-\alpha\right)\right)$. From Proposition 2.5.3, when $M=o(\sqrt{N})$, $\mathcal{C}_{\text{ave}}(N,M)=\Theta\left(NF\left(\bar{\Delta}_0\right)\right)$ and $\mathcal{C}_{\text{Alg}}(N,M)=\Theta(NF(\Delta_{u*}))$. Therefore, as long as α is properly chosen, the two-stage ranking has lower complexity than the non-adaptive ranking. However, since $\alpha>0$, there is a linear gap between the two-stage ranking of the proposed algorithm Ada-IIR. On the other hand, two-stage ranking is less constrained than Ada-IIR as it has an advantage over the non-adaptive scheme when the number of users M is in the regime $\Omega(\sqrt{N})$ while Ada-IIR does not. A more detailed analysis of the two-stage ranking is presented in Section 2.10.2 and Section 2.10.3.



(a) $\gamma_A=0.5$, $\gamma_B=2.5$

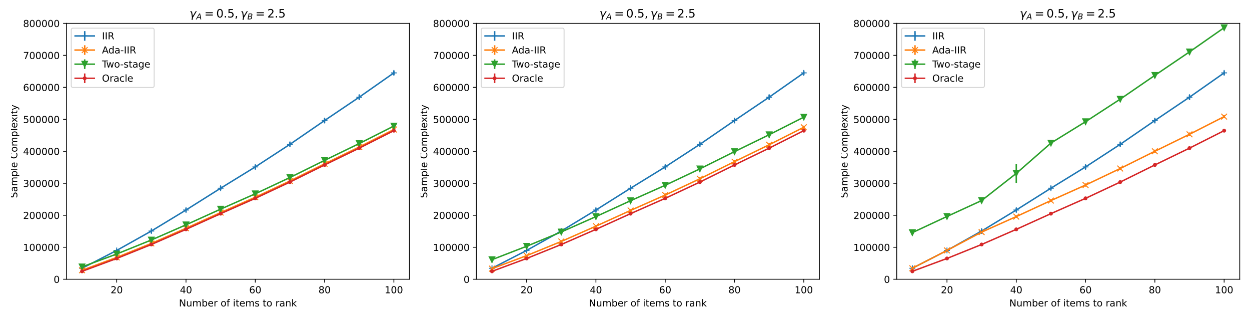


(b) $\gamma_A=0.5$, $\gamma_B=1.0$



(c) $\gamma_A=0.5$, $\gamma_B=0.5$

Figure 2.1: Sample complexities v.s. number of items for all algorithms. (a) (b) and (c) are different heterogeneous user settings where the accuracy of two groups of users differs.



(a) $M = 18$ (b) $M = 36$ (c) $M = 72$

Figure 2.2: Sample complexities v.s. number of items for all algorithms. (a) (b) and (c) are different settings where the number of users differs. The accuracy of two groups of users are $\gamma_A = 0.5$, $\gamma_B = 2.5$.

2.7 Experiments

In this section, we study the empirical performance of the following algorithms on both synthetic and real-world datasets:

- **IIR** (Ren et al., 2019): The original single-user algorithm adapted to the multi-user case by querying a user selected uniformly at random.
- **Ada-IIR**: The proposed method.
- **Two-stage ranking**: A simple method described in Section 2.6.
- **Oracle**: Query only the best user as if it is known.

Confidence parameter $\delta = 0.25$, $\alpha = 0.05$ is set if required by the algorithm.

2.7.1 Synthetic Experiment

In our experiment, we use a similar setup as that of Jin et al. (2020), except that every pair has the same distance. In particular, we consider a set of users $[M]$, whose accuracies are set by $p_u(i,j) = (1 + \exp(\gamma_u(s_j - s_i)))^{-1}$, for $u \in [M]$ and items $i, j \in [N]$, where parameter γ_u determines the user accuracy and s_i, s_j are the utility scores of the corresponding items in the BTL model. Larger values of γ_u lead to more accurate users. We set $s_i - s_j = 3$ if $i \prec j$ and $s_i - s_j = -3$ otherwise. Note that here we assume that the accuracy of user u is the same for all pairs of items (i, j) as long as $i \prec j$. We assume that there are two distinct groups of users: the high-accuracy group in which the users have the same accuracy $\gamma_u = \gamma_B \in \{0.5, 1.0, 2.5\}$ in three different settings; and the low-accuracy group in which the users have the same accuracy $\gamma_u = \gamma_A = 0.5$ in all settings. This set of γ_u, s_i, s_j is chosen so that $p_u(i,j)$ for accurate users ranges from 0.55 to 0.99 and inaccurate users have a value close to 0.55.

The number of items to be ranked ranges from 10 to 100. Each setting is repeated 100 times with randomly generated data. To showcase the effectiveness of active user selection, we tested a relatively adverse situation where only 12 out of $M = 36$ users are highly accurate.

The average sample complexity and standard deviation over 100 runs are plotted in Figure 2.1. Note that the standard deviation is hard to see, given that it is small compared to the average. In most cases, the proposed method achieves nearly identical performance to the oracle algorithm, with only a small overhead. For two-stage ranking, we observe a constant overhead regardless of the accuracy of the users. It may outperform the non-adaptive one (IIR) if there exist enough highly accurate users such as in Figure 2.1(a). However, the situation is less favorable for the two-stage algorithm when the cost of finding the best user overwhelms the savings of queries due to increased accuracy as shown in Figure 2.1(b). It may even have an adverse effect when accuracies are similar, as shown in Figure 2.1(c).

When we increase the total number of users and keep their accuracy the same, as shown in Figure 2.2, the Ada-IIR algorithm is able to tackle the increasing difficulty in finding more accurate users within a larger pool. Although, the overhead increases, our proposed method can adapt to each case and deliver near-optimal performance.

In our experiments, every algorithm can recover the exact rank to the ground truth, which is reasonable since the IIR algorithm is designed to output an exact ranking. And due to the union bounds used to guarantee a high probability of correct output, the algorithms tend to request more than enough queries so we did not see a case in which a non-exact ranking was produced.

2.7.2 Real-world Experiment

The above synthetic experiments serve as a proof of concept. We add one more experiment based on the real data, the setting is from the "Country Population" dataset from Jin et al. (2020). In this dataset, the population of 15 countries was ranked by workers. Since the ground-truth Δ_u is not available, we first used the method described in the same work to infer the user accuracy and item parameters. During the simulation, the responses are generated according to their model with these parameters. As we have discussed in 2.5.2, the number of users should fall in a reasonable range. Thus, we randomly sub-sample a set of 25 users since the set of users provided by the dataset is excessive. The results, shown in Table 2.1, suggest that the Ada-IIR provides a moderate improvement over the non-adaptive algorithm.

METHOD	SAMPLE COMPLEXITY
IIR	59223 ± 3183
Two-stage	85027 ± 2619
Ada-IIR	52693 ± 2739
Oracle	43855 ± 2365

Table 2.1: Experiments on Country Population with 15 items and 25 users.

2.8 Conclusions

In this paper, we study the heterogeneous rank aggregation problem, where noisy pairwise comparisons are provided by a group of users with different accuracy levels. We propose a new ranking algorithm based on the idea of arm elimination from multi-armed bandits. The algorithm can identify the best user and utilize this information to efficiently perform the ranking. Under the Bernoulli setting, we provide theoretical guarantees that our algorithm is comparable with the oracle algorithm that knows the best user, and the gap between the sample complexities of these two methods is only sublinear in the number of items. We conduct thorough experiments and show that the proposed algorithm can perform as well as the oracle algorithm and is significantly more sample-efficient than all baseline algorithms. One immediate and interesting future direction may be to extend our adaptive sampling algorithm to more complicated models such as the heterogeneous Bradley-Terry-Luce model and the heterogeneous Thurstone Case V model (Jin et al., 2020).

2.9 More Details About the Proposed Algorithm

We borrow the definition of Preference Interval Tree (PIT) (Feige et al., 1994a; Ren et al., 2019) based on which we can insert items to a ranked list. Specifically, given a list of ranked items S the PIT can be constructed using the following Algorithm 4.

For the completeness of our paper, we also present the subroutines Iterative-Attempting-Insertion (IAI) and Attempting-Insertion (ATI) in this section, which are omitted in Section 2.4 due to space limit. In particular, IAI is displayed in Algorithm 5, and ATI is displayed in Algorithm 6. Both algorithms are proposed by Ren et al. (2019) for adaptive sampling in the single-user setting.

Algorithm 4 Build PIT

Input parameters: S

Data structure: NODE = $\{left, mind, right, lchild, rchild, parent\}$, $left, mid, right$ holds index values, $lchild, rchild, parent$ points to any other NODE.

Initialize: $N = |S|$

```
 1: $X$ = CreateEmptyNode
 2: $X$.left = $-1$
 3: $X$.right = $|S|$
 4: $X$.mid = $\lfloor (X.left + X.right)/2 \rfloor$
 5: queue = [X]
 6: while queue.NotEmpty do
 7:     X = queue.PopFront
 8:     $X$.mid = $\lfloor (X.left + X.right)/2 \rfloor$
 9:     if X.right - X.left $> 1$ then
10:         lnode = CreateEmptyNode
11:         lnode.left = X.left
12:         lnode.right = mid
13:         X.lchild = lnode
14:         rnode = CreateEmptyNode
15:         queue.append(lnode)
16:         rnode.left = X.mid
17:         rnode.right = X.right
18:         X.rchild = rnode
19:         queue.append(rnode)
20:     end if
21: end while
22: replace $-1$ with $-\infty$, $|S|$ with $\infty$ in each Node.left and Node.right.
```

Algorithm 5 Subroutine: ITERATIVE ATTEMPT TO INSERT(IAI)

Input parameters: (i, S, δ)

Initialize: For all $\tau \in \mathbb{Z}^+$, set $\epsilon_\tau = 2^{-(\tau+1)}$ and $\delta_\tau = \frac{6\delta}{\pi^2\tau^2}$; $t \leftarrow 0$; $Flag \leftarrow$ *unsure*;

```
1: **repeat**
2:     $t \leftarrow t + 1$;
3:     $Flag \leftarrow$ATI$(i, S, \epsilon_\tau, \delta_\tau)$;
4: **until** $Flag =$ *inserted*
```

2.10 Two-Stage Ranking

In this section, we formally state and analyze the two-stage ranking presented in Section 2.6.

2.10.1 Algorithm Outline

We present two-stage ranking in Algorithm 7. As described in Section 2.6, an arbitrary pair of items is first fed to the IIR algorithm for determining the order with the 'average' user. Next, the Median-Elimination (ME) algorithm from Even-Dar et al. (2002a) is used to find an α-optimal user. After that, the total ranking can be obtained by applying the IIR algorithm to the selected user only. IIR takes a set of items, the confidence level, and a user as inputs and outputs a ranked version of the items. ME takes a set $\mathcal{U}$ of users, real numbers α, δ, and two ranked items as inputs and outputs an α-optimal user in $\mathcal{U}$ with probability at least $1 - \delta$.

Proof of Theorem 2.6.1. It is clear that two-stage ranking is composed of three procedures: IIR for determining the order of i, j, ME for obtaining an α-optimal user, and IIR again for producing the final true ranking. Therefore, the probability guarantee of two-stage ranking follows from applying the union bound on the three procedures.

Algorithm 6 Subroutine: ATTEMPT TO INSERT(ATI).

Input parameters: (i, S, ϵ, δ)

Initialize: Let z be a PIT constructed from S, $h \leftarrow \lceil 1 + \log_2(1 + |S|) \rceil$, the depth of z

For all leaf nodes u of z, initialize $c_u \leftarrow 0$; Set $t^{\max} \leftarrow \lceil \max\{4h, \frac{512}{25} \log \frac{2}{\delta}\} \rceil$ and $q \leftarrow \frac{15}{16}$

```
1:  $X \leftarrow$ the root node of $z$;
2:  for $t \leftarrow 1$ to $t^{\max}$ do
3:      if $X$ is the root node then
4:          if ATC$(i, X.\text{mid}, \epsilon, 1-q) = i$ then
5:              $X \leftarrow X.\text{rchild}$
6:          else
7:              $X \leftarrow X.\text{lchild}$
8:          end if
9:      else if $X$ is a leaf node then
10:         if ATC$(i, X.\text{left}, \epsilon, 1-\sqrt{q}) = i \wedge$ ATC$(i, X.\text{right}, \epsilon, 1-\sqrt{q}) = X.\text{right}$ then
11:             $c_X \leftarrow c_X + 1$
12:             if $c_X > b^t := \frac{1}{2}t + \sqrt{\frac{t}{2}\log\frac{\pi^2 t^2}{3\delta}} + 1$ then
13:                 Insert $i$ into the corresponding interval of $X$ and
14:                 return inserted
15:             end if
16:         else if $c_X > 0$ then
17:             $c_X \leftarrow c_X - 1$
18:         else
19:             $X \leftarrow X.\text{parent}$
20:         end if
21:     else
22:         if ATC$(i, X.\text{left}, \epsilon, 1-\sqrt[3]{q}) = X.\text{left} \vee$ ATC$(i, X.\text{right}, \epsilon, 1-\sqrt[3]{q}) = i$ then
23:             $X \leftarrow X.\text{parent}$
24:         else if ATC$(i, X.\text{mid}, \epsilon, 1-\sqrt[3]{q}) = i$ then
25:             $X \leftarrow X.\text{rchild}$
26:         else
27:             $X \leftarrow X.\text{lchild}$
28:         end if
29:     end if
30: end for
31: if there is a leaf node $u$ with $c_u \geqslant 1 + \frac{5}{16}t^{\max}$ then
32:     Insert $i$ into the corresponding interval of $u$ and
33:     return inserted
34: else
35:     return unsure
36: end if
```

Algorithm 7 TWO-STAGE RANKING($\mathcal{N}$,$\mathcal{U}$, α, δ)

```
input: set of items $\mathcal{N}$, set of users $\mathcal{U}$, desired near-optimal level $\alpha$, confidence level $\delta$.
Let $i, j$ be two arbitrary items. Let $\bar{u}$ be the 'average' user.
$[i', j'] \leftarrow$ Iterative-Insertion-Ranking($\{i, j\}, \frac{\delta}{3}, \bar{u}$).
$u_\alpha \leftarrow$ Median-Elimination($\mathcal{U}, \alpha, \frac{\delta}{3}, [i', j']$)
output: Iterative-Insertion-Ranking($\mathcal{N}, \frac{\delta}{3}, u_\alpha$)
```

From Proposition 2.3.3, Iterative-Insertion-Ranking$\big(\{i, j\}, \frac{\delta}{3}, \bar{u}\big)$ takes number of queries

$$\Theta\left(\frac{2}{\bar{\Delta}_0^2}\left(\log\log\frac{1}{\bar{\Delta}_0} + \log\left(\frac{6}{\delta}\right)\right)\right), \tag{2.10.1}$$

Iterative-Insertion-Ranking$\big(\mathcal{N}, \frac{\delta}{3}, u_\alpha\big)$ takes number of queries at most

$$\Theta\left(\frac{N}{(\Delta_{u^*} - \alpha)^2}\left(\log\log\frac{1}{\Delta_{u^*} - \alpha} + \log\left(\frac{3N}{\delta}\right)\right)\right) \tag{2.10.2}$$

by noting that the accuracy of u_α is at least $\Delta_{u^*} - \alpha$. Moreover, it is shown in Even-Dar et al. (2002a) that ME outputs an α-optimal user using at most

$$\Theta\left(\frac{M}{\alpha^2}\log\frac{1}{\delta}\right) \tag{2.10.3}$$

comparisons.

The desired complexity bound thus follows from summing up (2.10.1), (2.10.2) and (2.10.3). $\square$

2.10.2 Complexity Analysis

In this subsection, we provide a more detailed discussion on the complexity of the two-stage algorithm described in Algorithm 7. Recall that we define

$$F(x) = x^{-2}\left(\log\log x^{-1} + \log\left(N/\delta\right)\right).$$

When the average accuracy of users $\bar{\Delta}_0$ is a constant that reflects population accuracy (e.g., all user accuracies follow some probability distribution), the following propositions can be made.

Proposition 2.10.1. When $M = \omega(N \log N)$ or $\alpha = o\left(\sqrt{\frac{M}{N \log N}}\right)$,

$$\mathcal{C}_{\text{tsr}}(\alpha) = \omega(N \log N) + \Theta\left(NF(\Delta_{u^*} - \alpha)\right).$$

When $M = \omega(N \log N)$ or $\alpha = o\left(\sqrt{\frac{M}{N \log N}}\right)$, the number of comparisons it takes in the user-selection stage can be more costly than ranking items. In particular, when the number of users M is too large, even asking each user one question becomes affordable. When α is chosen too small, the improvement in the accuracy of the selected user leads to limited savings on the ranking. Both cases are undesirable.

Proposition 2.10.2. If $M = O(N)$ and $\alpha = \omega\left(\sqrt{\frac{M}{N \log N}}\right) \cap o(1)$, then

$$\mathcal{C}_{\text{tsr}}(\alpha) = \Theta\left(NF(\Delta_{u^*})\right) + o(N \log N) + O(1).$$

In the preceding proposition, the dominating term in $\mathcal{C}_{\text{tsr}}$ is $\Theta\left(NF\left(\Delta_{u^*}\right)\right)$, which is of the same order as $\mathcal{C}_{u^*}$. Therefore, when the number of users M is not much larger than the number of items N, two-stage ranking can achieve order optimal by choosing α small enough.

Proposition 2.10.3. If α is a constant,

$$\mathcal{C}_{\text{tsr}}(\alpha) = \Theta\left(NF(\Delta_{u^*} - \alpha)\right) + O(M).$$

In particular, when $M = o(N \log N)$, $\mathcal{C}_{\text{tsr}}\left(N, M\right)$ is dominated by the term $\Theta\left(NF\left(\Delta_{u^*} - \alpha\right)\right)$. This is equivalent to performing ranking using a single user with accuracy $\Delta_{u^*} - \alpha$.

2.10.3 User Selection in a Subset

As shown in Proposition 2.10.1, when M is much larger than $N \log N$, even querying each user once costs time linear in M which could be higher than the ranking complexity. Therefore, instead of selecting a global α-optimal user, we devise a subroutine Subset-User-Selection

(SUS) that randomly picks without replacement L ($L \leqslant M$) users and only searches for an α-optimal user among them (see Algorithm 8). We use $\mathcal{L}$ to denote this L-subset of users.

Algorithm 8 Subroutine: SUBSET-USER-SELECTION($\mathcal{U}, L, \alpha, \delta_i, \delta_m, i, j$)

```
input: set of users $\mathcal{U}$, user subset size $L$, desired near-optimal level $\alpha$, confidence level $\delta_i$ of initial ranking, confidence level $\delta_m$ of user selection, two items $i, j \in \mathcal{N}$.
$[i', j'] \leftarrow$ Iterative-Insertion-Ranking($\{i, j\}, \delta_i, \bar{u}$).
Randomly choose a subset $\mathcal{L}$ of $L$ users from $\mathcal{U}$.
output: Median-Elimination($\mathcal{L}, \alpha, \delta_m, [i', j']$)
```

The main procedure of the two-stage algorithm is also modified, shown in Algorithm 9.

Algorithm 9 MODIFIED-TWO-STAGE-RANKING($\mathcal{N}, \mathcal{U}, L, \alpha, \delta_i, \delta_m, \delta_r$)

```
input: set of items $\mathcal{N}$, set of users $\mathcal{U}$, user subset size $L$, desired near-optimal level $\alpha$, confidence level $\delta_i$ of initial ranking, confidence level $\delta_m$ of user selection, confidence level $\delta_r$ of final ranking.
Let $i, j$ be two arbitrary items.
$u_\alpha \leftarrow$ Subset-User-Selection($\mathcal{U}, L, \alpha, \delta_i, \delta_m, i, j$).
output: Iterative-Insertion-Ranking($\mathcal{N}, \delta_r, u^\alpha$)
```

Generally, no guarantee can be made on how close is a subset α-optimal user to the global optimal user. So analysis on the two-stage algorithm will be done under the assumption that the M user accuracies are iid samples drawn from a probability distribution $F(x)$ over the interval $(0, \frac{1}{2}]$ ($F(x)$ is independent of any other quantities). Let $b = \inf_x\{x : F(x) = 1\}$. In the following, we use the cdf $F(x)$ to represent this distribution.

Since $\Delta_1, \Delta_2, \ldots, \Delta_M$ are iid samples from $F(x)$ and $\mathcal{L}$ is drawn randomly, we assume WOLOG that $\mathcal{L}$ contains the first L users, i.e., $\mathcal{L} = \{1, 2, \ldots, L\}$. Let $\Delta^\circ = \max_{u \in \mathcal{L}} \Delta_u$. Recall that $\Delta_{u*} = \max_{u \in \mathcal{U}} \Delta_u$. We first show in the following lemma that $\Delta_{u*} - \Delta^\circ$ is independent of M.

Lemma 2.10.4. For any $\delta' \in (0, \frac{1}{2}), \alpha \in (0, b)$, if $L \geqslant \log(\delta')/\log\left(F(b-\alpha)\right)$, then with probability at least $1-\delta'$,

$$\Delta^{\circ} \geqslant \Delta_{u*} - \alpha.$$

Proof. Note that the claim becomes trivial when $M \leqslant \frac{\log \delta'}{\log(F(b-\alpha))}$. We consider the case when $\frac{\log \delta'}{\log(F(b-\alpha))} \leqslant L \leqslant M$.

Since $\Delta_1, \Delta_2, \ldots, \Delta_L$ are iid samples from $F(x)$, with probability $\left(F\left(b-\alpha\right)\right)^L$,

$$\Delta_i \leqslant b - \alpha \text{ for all } 1 \leqslant i \leqslant L.$$

Hence, $\left(F\left(b-\alpha\right)\right)^L \leqslant \delta'$ gives

$$\Delta^{\circ} = \max_{1 \leqslant i \leqslant L} \Delta_i \geqslant b - \alpha \geqslant \Delta_{u*} - \alpha$$

with probability at least $1-\delta'$, where the last inequality follows from $\Delta_{u*} \leqslant b$ with probability 1. $\square$

The preceding lemma states that when user accuracies follow a fixed distribution, at least one of the L users we select randomly will be close to the global best user as long as L is large enough (but still independent of M). Thus, even when the number of users M is huge, we do not need to collect information from every one of them. A randomly chosen subset is able to accurately reflect the characteristics of the larger group.

Next, we compute the number of comparisons needed for user selection. Our goal is to show that the complexity of user selection becomes negligible compared with item ranking. In the following analysis, for simplification, we assign the confidence levels $\delta_i, \delta_m, \delta_r$ in Two-Stage-Ranking as well as the confidence level δ' for the existence of an α-optimal user equal values. Specifically, we let $\delta' = \delta_i = \delta_m = \delta_r = \frac{\delta}{4}$ for some $\delta \in (0, 1)$.

Theorem 2.10.5. For any $\delta \in (0, \frac{1}{2}), \alpha \in (0, b), L = \min\left(\lceil \frac{\log(\delta/4)}{\log(F(b-\alpha/2))} \rceil, M\right)$, with probability at least $1 - \frac{3\delta}{4}$, subroutine Subset-User-Selection$(\mathcal{U}, L, \frac{\alpha}{2}, \frac{\delta}{4}, \frac{\delta}{4}, i, j)$ outputs a global

α-optimal user after

$$\Theta\left(\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1}+\log\frac{4}{\delta}\right)+\frac{4L}{\alpha^2}\log\frac{4}{\delta}\right)$$

comparisons.

Proof. By Lemma 2.10.4, letting $L = \min\left(\lceil\frac{\log(\delta/4)}{\log(F(b-\alpha/2))}\rceil, M\right)$ gives $\Delta^\circ \geqslant \Delta_{u*} - \frac{\alpha}{2}$ with probability at least $1-\frac{\delta}{4}$. Moreover, IIR finds the correct order of items i, j with probability at least $1-\frac{\delta}{4}$ and given that Median-Elimination outputs an $\frac{\alpha}{2}$-optimal user in the L-subset with probability at least $1-\frac{\delta}{4}$. Therefore, by the union bound, with probability $1-\frac{3\delta}{4}$, the $\frac{\alpha}{2}$-optimal user found is a global α-optimal user.

The complexity is a sum of two terms: the complexity of IIR ranking two items and the complexity of Median-Elimination outputting an $\alpha/2$-optimal user among L users. □

Theorem 2.10.6. For any $\delta \in (0,\frac{1}{2}), \alpha \in (0,b), L = \min\left(\lceil\frac{\log(\delta/4)}{\log(F(b-\alpha/2))}\rceil, M\right)$, with probability at least $1-\delta$, Modified-Two-Stage-Ranking($\mathcal{N}$,$\mathcal{U}$, $L, \alpha, \frac{\delta}{4}, \frac{\delta}{4}, \frac{\delta}{4}$) outputs the exact ranking of $\mathcal{N}$, and consumes

$$\mathcal{C}_{\mathrm{mtsr}}(\alpha) = \Theta\left(\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1}+\log\frac{4}{\delta}\right)+\frac{4L}{\alpha^2}\log\frac{4}{\delta}+NF\left(\Delta_{u*}-\alpha\right)\right)$$

comparisons.

Proof. Modified-Two-Stage-Ranking being able to output the exact ranking of $\mathcal{N}$ is guaranteed by the algorithm IIR.

It remains to compute the complexity. By Theorem 2.10.5, with probability at least $1-\frac{3}{4}\delta$, Subset-User-Selection outputs a global α-optimal user. With a global α-optimal user, IIR outputs the exact ranking of $\mathcal{N}$ after

$$\Theta\left(NF\left(\Delta_{u*}-\alpha\right)\right),$$

comparisons with probability at least $1-\frac{\delta}{4}$. Therefore, the desired complexity follows from applying the union bound and summing up the complexities of SUS and IIR. □

By noting that for M sufficiently large, $\bar{\Delta}$ equals the mean of $F(x)$ with probability 1 and thus $\bar{\Delta}^{-2}\left(\log\log\bar{\Delta}^{-1} + \log\frac{4}{\delta}\right) = O(1)$, we have the following proposition.

Proposition 2.10.7. If $\alpha = \Omega(N^{-\frac{1}{2}}) \cap o(1)$, then

$$\mathcal{C}_{\text{mtsr}}(\alpha) = \Theta\left(NF(\Delta_{u*})\right) + O(N).$$

Comparing the preceding proposition with Proposition 2.10.2, we can see that by Subset-User-Selection, the two-stage algorithm can perform efficiently even with a large number of users.

2.11 Omitted Proofs

2.11.1 Query Complexity of the Proposed Algorithm

The following lemmas characterize the performance of each subroutine:

Lemma 2.11.1 (Lemma 9 in Ren et al. (2019))**.** For any input pair (i, j) and a set of users $\mathcal{U}$, Algorithm 2 terminates in $\lceil r_{\max}\rceil = \lceil \epsilon^{-2}\log(2/\delta)\rceil$ queries. If $\epsilon \leqslant \bar{\Delta}$, then the returned $\widehat{y}$ indicates the preferable item with probability at least $1-\delta$.

Lemma 2.11.2 (Lemma 10 in Ren et al. (2019))**.** Algorithm 6 returns after $O(\epsilon^2\log(|S|/\delta)$ queries and, with probability $1-\delta$, correctly insert or return unsure. Additionally, if $\epsilon \leqslant \bar{\Delta}$, Algorithm 6 will insert correctly with probability $1-\delta$.

Lemma 2.11.3 (Lemma 11 in Ren et al. (2019))**.** With probability $1-\delta$, Algorithm 5 correctly insert the item and makes $O(\bar{\Delta}^{-2}(\log\log\bar{\Delta}^{-1} + \log(N/\delta)))$ queries at most.

Proof of Theorem 2.5.1. When inserting the z-th item, we makes at most $\bar{\Delta}_z^{-2}(\log\log\bar{\Delta}_z^{-1} + \log(N/\delta))$ queries, for $z = 2, 3, \ldots, N$.

The number of total queries can be obtained by summing up the term above, which is

$$\mathcal{C}_{\text{Alg}}(N) = O\Bigg(\sum_{z=2}^{N}\bar{\Delta}_z^{-2}(\log\log\bar{\Delta}_z^{-1} + \log(N/\delta))\Bigg).$$

□

2.11.2 Complexity Gap Analysis

The first lemma we will introduce is about the confidence interval:

Lemma 2.11.4. With probability $1-\delta$, it holds for any $z \in [N]\backslash\{1\}$ and $u \in \mathcal{U}_z$,

$$\frac{1}{2} + \Delta_u \in \Big[(\mathbf{LCB}_z)_u, (\mathbf{UCB}_z)_u\Big].$$

This also indicates that when inserting the z-th item, for any $u \in \mathcal{U}_z$,

$$\Delta_{u*} - \Delta_u \leqslant 4r_z.$$

Proof of Lemma 2.11.4. Recall that $(\boldsymbol{\mu}_z)_u$ is the empirical mean of the Bernoulli variable with parameter $\frac{1}{2} + \Delta_u$. For a given z and u, by Hoeffding's inequality we have

$$\mathbb{P}\left(\left|(\boldsymbol{\mu}_z)_u - \left(\frac{1}{2} + \Delta_u\right)\right| > r_z\right) \leqslant 2e^{-2(\mathbf{s}_z)_u r_u^2} \leqslant 2e^{-2(\mathbf{s}_z)_{\min} r_u^2} \leqslant \frac{\delta}{|\mathcal{U}_z|N},$$

and applying union bound over $z = 2, 3, \ldots, N$ and $u \in \mathcal{U}_z$ gives the claim.

Under this event, we have

$$\begin{aligned}
\Delta_{u*} - \Delta_u &= \left(\frac{1}{2} + \Delta_{u*}\right) - \left(\frac{1}{2} + \Delta_u\right) \\
&\leqslant (\mathbf{UCB}_z)_{u*} - (\mathbf{LCB}_z)_u \\
&\leqslant (\mathbf{UCB}_z)_{u*} - (\mathbf{LCB}_z)_{u*} + (\mathbf{UCB}_z)_u - (\mathbf{LCB}_z)_u \\
&= 4r_z,
\end{aligned}$$

where the first inequality is clearly from the confidence interval, and the second inequality holds because the two confidence intervals should intersect. □

Next, we will introduce another lemma concerning the growth of $(\mathbf{s}_z)_u$ for each $u \in \mathcal{U}_z$.

Lemma 2.11.5. Denote S_z as all queries made till inserting the z-th item and $M = |\mathcal{U}_0|$. Suppose $S_z \geqslant 2M^2 \log(NM/\delta)$. With probability $1 - \delta$, we have for any $z \in \{2, 3, \ldots, N\}$,

$$(\mathbf{s}_z)_{\min} \geqslant \frac{S_z}{2M}.$$

Proof of Lemma 2.11.5. For fixed z and $u \in \mathcal{U}_z$, by Hoeffding's inequality we have

$$\begin{aligned}\mathbb{P}\left(\frac{(\mathbf{s}_z)_u}{S_z} - \frac{1}{M} < -\frac{1}{2M}\right) &\leqslant \mathbb{P}\left(\frac{(\mathbf{s}_z)_u}{S_z} - \mathbb{E}\left[\frac{(\mathbf{s}_z)_u}{S_z}\right] < -\frac{1}{2M}\right)\\ &\leqslant \exp\left(-\frac{S_z}{2M^2}\right) \leqslant \frac{\delta}{NM}.\end{aligned}$$

Applying union bound we know that with probability $1 - \delta$,

$$(\mathbf{s}_z)_u \geqslant \frac{S_z}{2M}, \forall z \in \{2, 3, \ldots, N\}, \forall u \in \mathcal{U}_z.$$

Since $(\mathbf{s}_z)_{\min} := \min_{u \in \mathcal{U}_z} (\mathbf{s}_z)_u$, we have

$$(\mathbf{s}_z)_{\min} \geqslant \frac{S_z}{2M}, \forall z \in \{2, 3, \ldots, N\}.$$

□

With the two lemmas above, we can control the accuracy gap as follows:

Lemma 2.11.6. Denote $\bar{\Delta}_z = \frac{1}{|\mathcal{U}_z|} \sum_{u \in \mathcal{U}_z} \Delta_u$. Suppose $S_z \geqslant 2|M|^2 \log(NM/\delta)$. With probability $1 - 2\delta$, we have for any $t \in [N]$,

$$\Delta_{u*} - \bar{\Delta}_z \leqslant \operatorname{polylog}(N, M, \delta^{-1}) \cdot \sqrt{\frac{M}{S_z}}.$$

Proof of Lemma 2.11.6. The proof has two steps:

From Lemma 2.11.5 we know that with probability $1 - \delta$,

$$(\mathbf{s}_z)_{\min} \geqslant \frac{S_z}{2M}, \forall t \in [N], \forall u \in \mathcal{U}_z.$$

From Lemma 2.11.4, we know with probability $1-\delta$(recall that $(\mathbf{r}_z)_u = \sqrt{\frac{\log(2|\mathcal{U}_z|N/\delta)}{2(\mathbf{s}_z)_{\min}}}$),

$$\begin{aligned}\Delta_{u^*} - \Delta_u &\leqslant 4r_z \\ &\leqslant 4\sqrt{\frac{M\log(2MN/\delta)}{S_z}} \\ &= 4\sqrt{\log(2MN/\delta)} \cdot \sqrt{\frac{M}{S_z}}.\end{aligned}$$

□

Define function $F(x) = x^{-2}(\log\log(x^{-1}) + \log(N/\delta))$ with $x \in (0, 1/2]$. We care about the following term GAP which characterize the query complexity gap between our algorithm and the optimal user.

$$\mathrm{GAP}(N, M, \delta) = \sum_{z=2}^{N} F(\bar{\Delta}_z) - F(\Delta_{u^*}).$$

The following lemma provides a way to linear bound the gap between function values:

Lemma 2.11.7. $F(x) = x^{-2}(\log\log(x^{-1}) + \log(N/\delta))$ with $x \in (0, 1/2]$ is a convex function over $(0, 1/2]$, and for any $\Delta \in [a, b]$, we have

$$F(\Delta) - F(b) \leqslant \frac{F(a) - F(b)}{b - a} \cdot (b - \Delta) = L(a, b) \cdot (b - \Delta).$$

Furthermore, under the event of Lemma 2.11.6, for any $z \in [N]$ such that $S_z > 2M^2\log(NM/\delta)$, we have $\bar{\Delta}_z \in [c\Delta_{u^*}^3, \Delta_{u^*}]$ and therefore

$$F(\bar{\Delta}_z) - F(\Delta_{u^*}) \leqslant \frac{F(c\Delta_{u^*}^3) - F(\Delta_{u^*})}{\Delta_{u^*} - c\Delta_{u^*}^3} \cdot (\Delta_{u^*} - \bar{\Delta}_z) = L(\mathcal{U}_0) \cdot (\Delta_{u^*} - \bar{\Delta}_z).$$

Here we use $L(\mathcal{U}_0) = \frac{F(c\Delta_{u^*}^3) - F(\Delta_{u^*})}{\Delta_{u^*} - c\Delta_{u^*}^3}$ is indeed an instance-dependent factor, with only logarithmic dependent in N and δ^{-1}(in F). c is a global constant and in fact $c = 1/25$.

Proof. Differentiate $F(x)$ twice and it can be verified that $F''(x) > 0$. For any $\Delta \in [a, b]$, the inequality above is easy to prove via convexity.

The rest is to prove that $\forall t \in [N]$, we have $\bar{\Delta}_z \in [\Delta_{u^*}/M, \Delta_{u^*}]$. It is clear that the upper bound holds because $\Delta_{u^*} := \max_{u \in \mathcal{U}_0} \Delta_u$.

The lower bound is proved as follows: We still have $\bar{\Delta}_z > \Delta_{u^*}/M$ because at any time u^* always remains in the user set and by the assumption $\Delta_u > 0$.

Also, since $S_z > 2M^2 \log(NM/\delta)$, by Lemma 2.11.6, we have

$$\begin{aligned}\Delta_{u^*} - \bar{\Delta}_z &\leqslant 4\sqrt{\frac{M \log(2MN/\delta)}{S_z}} \\ &\leqslant 4\sqrt{\frac{M \log(2MN/\delta)}{2M^2 \log(NM/\delta)}} \\ &\leqslant \frac{4}{\sqrt{M}}.\end{aligned}$$

Now we will prove that

$$\max\left\{\frac{\Delta_{u^*}}{M}, \Delta_{u^*} - \frac{4}{\sqrt{M}}\right\} \geqslant c\Delta_{u^*}^3.$$

Suppose $\frac{\Delta_{u^*}}{M} < c\Delta_{u^*}^3$, then we have $M > c^{-1}\Delta_{u^*}^{-2}$, this means

$$\Delta_{u^*} - \frac{4}{\sqrt{M}} \geqslant \Delta_{u^*} - 4\sqrt{c}\Delta_{u^*} \geqslant c\Delta_{u^*}^3.$$

The last inequality is due to $\Delta_{u^*} \leqslant 1/2$ and $c = 1/25$. □

Now we are ready to prove the main result:

Proof of Theorem 2.5.2. Based on our algorithmic design, we will not eliminate any user until the cumulative number of queries S_z reaches the threshold $S_z \geqslant 2M^2 \log(NM/\delta)$. We have

$$\begin{aligned}\mathrm{GAP}(N, M, \delta) &= \sum_{z=2}^{N} F(\bar{\Delta}_z) - F(\Delta_{u^*}) \\ &= \underbrace{\sum_{z=2}^{N} \mathbb{1}\{S_z < 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_z) - F(\Delta_{u^*})\big)}_{I_1} \\ &\quad + \underbrace{\sum_{z=2}^{N} \mathbb{1}\{S_z \geqslant 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_z) - F(\Delta_{u^*})\big)}_{I_2}.\end{aligned}$$

For I_1, no elimination is performed, so $\mathcal{U}_z = \mathcal{U}_0$, and we have

$$I_1 = \sum_{z=2}^{N} \mathbb{1}\{S_z < 2M^2 \log(NM/\delta)\}\big(F(\bar{\Delta}_0) - F(\Delta_{u^*})\big).$$

For each term in I_2, we have $F(\bar{\Delta}_z) - F(\Delta_{u^*}) \leqslant L(\mathcal{U}_0) \cdot 4\sqrt{\log(2MN/\delta)} \cdot \sqrt{\frac{M}{S_z}}$ due to Lemma 2.11.7 and Lemma 2.11.6. Therefore,

$$I_2 \leqslant L(\mathcal{U}_0) 4\sqrt{\log(2MN/\delta)} \sum_{z=2}^{N} \mathbb{1}\{S_z \geqslant 2M^2 \log(NM/\delta)\}\sqrt{\frac{M}{S_z}}.$$

□

2.11.3 Proof and Discussions of Proposition 2.5.3

Suppose $M = o(N^{1/2})$, since $S_z \geqslant z\log(z/\delta) \geqslant z$(at least one comparison for an item), from (2.5.2) we have

$$\sum_{z=2}^{N} \mathbb{1}\left\{S_z < 2M^2 \log(NM/\delta)\right\} \leqslant \sum_{z=2}^{N} \mathbb{1}\left\{z < 2M^2 \log(NM/\delta)\right\} = o(N).$$

The third term can be bounded with the fact $\mathbb{1}\left\{z < 2M^2\log(NM/\delta)\right\} \leqslant 1$,

$$\begin{aligned} & L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\mathbb{1}\{S_z \geqslant 2M^2\log(NM/\delta)\}\sqrt{\frac{M}{S_z}} \\ & \leqslant L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\sqrt{\frac{M}{S_z}} \\ & \leqslant L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sum_{z=2}^{N}\sqrt{\frac{M}{z}} \\ & \leqslant 2L(\mathcal{U}_0)\sqrt{\log(2MN/\delta)}\sqrt{MN} \\ & = O(L(\mathcal{U}_0)\sqrt{\log(MN/\delta)}\sqrt{MN}). \end{aligned}$$

$L(\mathcal{U}_0)$ is actually dominated by the minimal mean accuracy $\min_z \bar{\Delta}_z$ throughout the algorithm. In practice, $L(\mathcal{U}_0)$ is usually a constant, related to all users' accuracy. In the worst theoretical case, $L(\mathcal{U}_0)$ will be dominated by $F(\Delta_{u*}/M) = \widetilde{O}(M^2)$, which further turns the last term into $\widetilde{O}(M^{5/2}N^{1/2})$, and requires $M = o(N^{1/5})$ so that this term becomes negligible.

CHAPTER 3

Active Ranking without Strong Stochastic Transitivity

3.1 Introduction

Ranking from noisy comparisons has a wide range of applications including voting Caplin and Nalebuff (1991); Conitzer and Sandholm (2005), identifying the winner/full ranking of teams in sport leagues, ranking players in online gaming systems Herbrich et al. (2006b), crowdsourcing services Chen et al. (2013), web search Dwork et al. (2001), and recommendation systems Baltrunas et al. (2010); Piech et al. (2013). In practice, comparisons usually contain certain levels of "noise". For example, duels in a game are not always won by the more proficient player, and preferences between movies/restaurants can also vary among different individuals. The presence of noise is commonly studied using a probabilistic comparison model Falahatgar et al. (2018); Ren et al. (2019), where an item has a certain probability to win the comparison over another (pairwise) or a group of items (listwise).

We are interested in estimating the total ranking. To guarantee that the ranking is consistent with the preference probabilities, it is often assumed Feige et al. (1994b); Mohajer et al. (2017); Falahatgar et al. (2018); Ren et al. (2019) that if i ranks higher than j, then i wins a comparison between j with probability $p_{i,j} > \frac{1}{2}$. It is clear that the closer $p_{i,j}$ is to 1/2, the more difficult it becomes to compare i and j. This assumption is referred to as *Weak Stochastic Transitivity (WST)*. A more strict assumption, *Strong Stochastic Transitivity (SST)*, is also often made Falahatgar et al. (2017a); Ren et al. (2018); Saha and Gopalan (2019). SST requires items that have closer ranks to be more difficult to compare, i.e., if

$i \succ j \succ k$, then $p_{i,k} \geqslant \max(p_{i,j}, p_{j,k}) > \frac{1}{2}$. Formal definitions of WST and SST are stated in Section 3.2.

However, SST can be too strong for scenarios where preference probabilities are not based on comparing a single quantifiable attribute. For instance, in sports, match outcomes are usually affected by team tactics. Team k may play a tactic that counters team i, resulting in a higher winning rate against team i compared with team j. Furthermore, items usually have multidimensional features and people may compare different pairs based on different features. A close pair in the overall ranking is thus not necessarily harder to compare than a pair that has a large gap. For example, when comparing cars, people might compare a given pair based on their interior design and another pair based on performance. As another example, in an experiment with games of chance with different probabilities of winning and payoffs Tversky (1969), it was observed that "people chose between adjacent gambles according to the payoff and between the more extreme gambles according to probability, or expected value."

Motivated by such applications, in this chapter, we are interested in the problem of recovering the full ranking of n items under a more general setting, where only WST holds, while SST is not assumed to hold. We focus on only pairwise queries as they are easier to obtain and less prone to error in practice. Furthermore, as many applications Chen et al. (2013); Pfeiffer et al. (2012) allow interactions between users/annotators, we consider comparisons collected in an adaptive manner. Our goal is to use as few comparisons as possible and achieve a high confidence.

Existing algorithms Mohajer et al. (2017); Ren et al. (2019) cannot avoid comparing every item i with the item i^* that is the most similar to i, i.e., $\left|p_{i,i^*} - \frac{1}{2}\right| = \min_{j \neq i}\{\left|p_{i,j} - \frac{1}{2}\right|\}$. Further, Ren et al. (2019) pointed out that comparing item pairs that are adjacent in the true ranking are necessary. When SST holds, adjacent pairs are also the most difficult pairs to distinguish, existing methods thus achieve sample-efficiency. For example, the Iterative-Insertion-Ranking (IIR) algorithm proposed in Ren et al. (2019) maintains a preference tree and performs ranking by inserting items one after another. During the insertion process, every item is possible to be compared with every other items (and thus the most similar one), depending on the relative order of insertion and the true ranking. IIR was shown to be sample complexity optimal under SST and some other conditions.

However, when SST does not hold, comparing nonadjacent items harms the performance. Consider an extreme scenario where the true ranking is $1 \succ 2 \succ 3$ and $p_{1,2} = p_{2,3} = 0.8, p_{1,3} = \frac{1}{2} + 2^{-10}$. If item 1 is directly compared to item 3, then it takes $\Theta\left(2^{20}\right)$ comparisons[1]. For instance, in IIR, this can happen during the insertion process of item 3 when item 1 happens to be the root of the preference tree. On the other hand, a simple fix exists as we can let the three pairs to be compared simultaneously. The comparisons between items 1 and 2, items 2 and 3 will terminate much earlier and provide us with the information $1 \succ 2$, $2 \succ 3$, which is enough to recover the total ranking. Therefore, it is important to devise an algorithm whose sample complexity will not be harmed when SST fails to hold.

Contribution. In this chapter, we propose an active algorithm, termed *Probe-Rank*, that ranks n items based on pairwise comparisons. Probe-Rank is a maxing-based algorithm, i.e., it ranks items by performing $n-1$ steps of maxing. We show that as long as the WST condition is satisfied, with probability at least $1-\delta$, Probe-Rank returns the correct ranking after conducting at most

$$O\left(n \sum_{i=1}^{n}\left(\widetilde{\Delta}_i^{-2}\right)\left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(n/\delta\right)\right)\right) \tag{3.1.1}$$

comparisons, where $\widetilde{\Delta}_i = \min_{j:j \text{ and } i \text{ are adjacent}} \left|p_{i,j} - \frac{1}{2}\right|$. Probe-Rank is the first algorithm whose sample complexity only depends on comparison probabilities of adjacent items instead of all pairs of items (Mohajer et al., 2017; Ren et al., 2019; Szörényi et al., 2015; Wu et al., 2022). Theoretical analysis and numerical experiments under various settings are provided

[1] In fact, according to Farrell (1964), we need $\Theta\left((p_{i,j} - 1/2)^{-2}\right)$ comparisons to be confident enough about the order between any two items i and j , $i, j \in [n]$.

and show that Probe-Rank is more efficient than the state-of-the-art when comparing nonadjacent items is more difficult than comparing adjacent items. We also present a preliminary analysis on the sample complexity lower bound in the worst-case scenario when SST does not hold. Further, we present a variant of Probe-Rank, named Probe-Rank-SE, in Section 3.10. Numerical experiments show that the variant is more sample-efficient under various settings.

3.2 Preliminaries

Notation We write $p \sim Uni(a, b)$ to denote that p is sampled uniformly at random from the interval (a, b), and use $\mathrm{Ber}(p)$ to denote a Bernoulli random variable which equals 1 with probability p. We use (i, j) to denote the unordered item pair, i.e., $(i, j) = (j, i)$.

Problem setup We assume that there exists a total ordering '$\succ$' over $[n]$ such that $\sigma_1 \succ \sigma_2 \succ \cdots \succ \sigma_n$ for some permutation σ of $[n]$. The permutation σ is referred to as the true ranking. Two items are called adjacent if they are adjacent in σ, i.e., one ranks right next to the other. To ensure that the true ranking σ is consistent with comparisons, we also assume that i has a higher rank than j if and only if $p_{i,j} > \frac{1}{2}$. In other words, if an item i is more preferred than j in σ, then i has a better chance to win the comparison with j. This assumption is known as *Weak Stochastic Transitivity (WST)*. A more strict assumption, *Strong Stochastic Transitivity (SST)*, is also frequently adopted. In addition to WST, SST assumes that whenever $i \succ j \succ k$, $p_{i,k} \geqslant \max(p_{i,j}, p_{j,k})$. In this chapter, we assume only WST and our goal is to recover the true ranking σ with a given confidence level δ by taking pairwise comparisons and to minimize the sample complexity. Problem instances are uniquely determined by the permutation σ representing the true ranking and the comparison probabilities $\mathbf{P}$.

Definition 1 (δ-correct algorithm)**.** *An algorithm is said to be δ-correct if for any input instance, with probability at least $1 - \delta$, it returns a correct result in finite time.*

It is clear that the closer $p_{i,j}$ is to $\frac{1}{2}$, the more difficult it becomes to obtain the ordering between i and j. Therefore, the probability gap $\Delta_{i,j}$, defined as $\Delta_{i,j} = \left|p_{i,j} - \frac{1}{2}\right|$, provides a characterization of the ranking task difficulty and will be used as a parameter for measuring sample complexities of algorithms. For instance, (Ren et al., 2019, lemma 12) shows that for any δ-correct algorithm $\mathcal{A}$, $\limsup_{\Delta \to 0} \frac{T_{\mathcal{A}}[\Delta]}{\Delta^{-2}(\log\log\Delta^{-1} + \log\delta^{-1})} > 0$, where $T_{\mathcal{A}}[\Delta]$ is the expected number of samples taken by $\mathcal{A}$ on two items with probability gap Δ. Further, for each item i, we define $\Delta_i = \min_{j:j\neq i} \Delta_{i,j}$, the minimum probability gap between item i and any other item j, and

$$\widetilde{\Delta}_i = \min_{j:j \text{ and } i \text{ are adjacent in } \sigma} \Delta_{i,j}, \tag{3.2.1}$$

the minimum probability gap between i and its adjacent items in the true ranking. Note that $\Delta_i \leqslant \widetilde{\Delta}_i$ by definition and the equality holds when SST is satisfied.

3.3 Related Work

The problem of ranking under coherent probabilistic comparisons dates back to 1994 Feige et al. (1994b). Feige et al. (1994b) studied the comparison model assuming that $i \succ j \Leftrightarrow p_{i,j} = \frac{1}{2} + \Delta$ for some known Δ. It was shown that any δ-correct algorithm finds the true ranking with at least $\Theta(n\Delta^{-2}\log(n/\delta))$ comparisons in the worst case. Later in Mohajer et al. (2017), a δ-correct algorithm TOP was proposed to rank the top-k elements by assuming only the existence of a total ranking (WST). The state-of-the-art IIR algorithm was proposed in Ren et al. (2019), as discussed in Section 3.1. A comparison of related algorithms are presented in Table 3.1.

Ranking or maxing has also been widely studied under more strict assumptions, e.g., SST, RST[2] and STI[3] and usually in the probably approximately correct (PAC) setting Falahatgar

[2]Under relaxed stochastic transitivity (RST), it is assumed that for all $i \succ j \succ k$, $\Delta_{i,k} \geqslant \gamma \max\{\Delta_{i,j}, \Delta_{j,k}\}$ for some $0 < \gamma < 1$.

[3]Under stochastic triangle inequality (STI), it is assumed that for all $i \succ j \succ k$, $\Delta_{i,k} \leqslant \Delta_{i,j} + \Delta_{j,k}$.

Table 3.1: δ-correct algorithms for exact ranking with sample complexity guarantee. Definitions of $\Delta_{i,j}, \Delta_i, \widetilde{\Delta}_i$ can be found in Section 3.2.

Algorithm	Assumptions on $\mathbf{P}$	Sample complexity
Single Elimination Tournament Mohajer et al. (2017)	WST	$O\left(\frac{n(\log n)^2 \log(1/\delta)}{\min_{1\leqslant i<j\leqslant n} \Delta_{i,j}^2}\right)$
PLPAC-AMPR Szörényi et al. (2015)	The Plackett-Luce model	$O\left(n \log n \max_{i\in[n]}\{\frac{1}{\Delta_i^2}\log(\frac{n}{\delta\Delta_i})\}\right)$
Iterative-Insertion-Ranking Ren et al. (2019)	WST	$O\left(\sum_{i=1}^{n}\frac{1}{\Delta_i^2}\left(\log\log\frac{1}{\Delta_i}+\log\frac{n}{\delta}\right)\right)$
Probe-Rank (ours)	WST	$O\left(n\sum_{i=1}^{n}\frac{1}{(\widetilde{\Delta}_i)^2}\left(\log\log\frac{1}{\Delta_i}+\log\frac{n}{\delta}\right)\right)$

et al. (2017a,b, 2018); Ren et al. (2018); Saha and Gopalan (2019, 2020); Szörényi et al. (2015); Yue and Joachims (2011a). In particular, Ren et al. (2018); Saha and Gopalan (2019, 2020); Szörényi et al. (2015) considered parametric comparison models such as the multinomial logit (MNL) model. Note that parametric models are often more restrictive and can imply SST/STI conditions. In the PAC setting, the goal is to find an ϵ-ranking $r_1 \succ r_2 \succ \cdots \succ r_n$ such that $p_{r_i,r_j} > \frac{1}{2} - \epsilon$ for all $i < j$. Although ϵ-rankings become closer to the true ranking as ϵ goes to 0, it is pointed out by Ren et al. (2019) that PAC ranking algorithms cannot be easily extended to the case when $\epsilon = 0$. Among all, Falahatgar et al. (2018) is the most relevant work to this chapter. In Falahatgar et al. (2018), PAC ranking and maxing were studied for both SST and WST settings. For WST, an instance-independent lower bound $\Theta(n^2)$ was proved, and a brute-force algorithm which compares each pair to an accuracy of ϵ and thus conducts $O((n^2/\epsilon^2)\log(n/\delta))$ comparisons was proposed. Note that in this chapter, we are aiming at recovering the exact ranking instead of an ϵ-ranking. An exact ranking is preferred over an epsilon-ranking in competitive applications like voting and sport games, where people are not satisfied with an approximate winner. Furthermore, as suggested by Ren et al. (2019), analyzing the exact ranking helps us to gain a better understanding about the instance-wise upper and lower bounds. A trivial extension of the brute-force algorithm can lead to sample complexity $\widetilde{O}\left(\frac{n^2}{\min_{i,j}\Delta_{i,j}^2}\right)$, which is substantially worse than our proposed algorithm.

Although we believe WST can be considered a natural and reasonably weak assumption, there are situations that WST does not hold as a ranking over items may not exist or, if it does, all comparison probabilities are not necessarily consistent with that ranking. So another line of research is to allow comparison probabilities $p_{i,j}$ take any values in $(0,1)$ as long as $p_{i,j}+p_{j,i}=1$. In such scenarios, rankings can be defined and derived based on various criteria including Borda score Heckel et al. (2019); Katariya et al. (2018); Shah and Wainwright (2017) and Copeland score Busa-Fekete et al. (2013); Zoghi et al. (2015a). The ranking problem has also been studied from a heterogeneous perspective Jin et al. (2020); Wu et al. (2022), where queries are made by multiple agents with different comparison probabilities. In Haddenhorst et al. (2021), the problem of testing whether the WST condition holds was studied. More broadly, the problems of ranking, maxing or selection can be formulated in the context of dueling bandits. A comprehensive survey can be found in Bengs et al. (2021).

3.4 Proposed Algorithm

In this section, we propose a δ-correct algorithm for exact ranking of all problem instances that satisfy the WST condition. As mentioned previously, our algorithm is designed to outperform existing methods in situations where nonadjacent items can be more difficult to compare than adjacent items.

To avoid spending unnecessary samples on item pairs with small probability gaps, we propose a subroutine named *Successive-Comparison* (SC) (see Subroutine 10). SC uses a parameter τ for controlling to what extent the comparison should last. Specifically, SC compares a given item pair for a fixed number $b_\tau = \lceil (2/\epsilon_\tau^2)\log(1/\delta_\tau)\rceil$ times with an accuracy level $\epsilon_\tau = 2^{-\tau}$ and confidence level $\delta_\tau = 6\delta/(\tau^2\pi^2)$. If the empirical probability that i (respectively, j) wins is over $1/2$ by more than $\epsilon_\tau/2$, then SC returns i (respectively, j) as the more preferred item. Otherwise, SC will return 'unsure' to inform us that more samples are needed.

For two items i and j, SC (i, j, δ, τ) will be called successively with τ increasing by 1 at a time. We show in Section 3.9 that after τ gets large enough such that $\epsilon_\tau \leqslant \Delta_{i,j}$, the correct ordering between i and j will be returned with high probability.

Subroutine 10 Successive-Comparison(i, j, δ, τ) (SC)

```
1: **Input:** items $i, j$, confidence level $\delta$, probing parameter $\tau$
2: $w_i = 0, \epsilon_\tau = 2^{-\tau}, \delta_\tau = \frac{\delta}{c\tau^2}, c = \frac{\pi^2}{6}$, $b_\tau = \lceil \frac{2}{\epsilon_\tau^2} \log \frac{1}{\delta_\tau} \rceil$;
3: **For** $t = 1$ to $b_\tau$ **do**
4:     compare $i$ and $j$ once; if $i$ wins, $w_i = w_i + 1$;
5: $\widehat{p}_i = w_i / b_\tau$;
6: Return $[i, j]$ **if** $\widehat{p}_i - \frac{1}{2} > \frac{1}{2}\epsilon_\tau$; Return $[j, i]$ **if** $\widehat{p}_i - \frac{1}{2} < -\frac{1}{2}\epsilon_\tau$; and Return 'unsure' **else**;
```

Partial Order Preserving Graph During the ranking process, we maintain a directed graph T to store the partial orders we have obtained from SC instances so far. The graph T is initialized with n nodes $V_1, \ldots, V_n$ and no edge exists between any two nodes. Nodes $V_1, V_2, \ldots, V_n$ represent items $1, 2, \ldots, n$, respectively. In our algorithm, T is involved with three types of operations, *edge update*, *node removal* and *maximal set selection*. Every time an instance of SC returns a pairwise order, e.g., $i \succ j$, we add a directed edge from V_i to V_j, written as $T = T \cup (i \succ j)$. Moreover, we also complete all edges in the transitive closure of the existing edges. In other words, if the edge between V_i and V_j induces a directed path from V_{k_1} to V_{k_2}, then a directed edge from V_{k_1} to V_{k_2} is also added to T. By completing the transitive closure, we can avoid comparing pairs whose ordering can be inferred from current knowledge and keep T acyclic. In the ranking process, we only run comparisons on item pairs that are not connected by edges and hence no contradictions in orderings will be returned by SC. By removing node V_i, we remove V_i and all edges of V_i from T. The maximal elements of T are the nodes which do not have any incoming edges. Since edges represent comparison results returned by SC, maximal elements correspond to items that have not lost to any other items. Note that since T is acyclic, maximal elements always exist.

Next, we establish our ranking algorithm *Probe-Rank* (see Algorithm 11). Probe-Rank finds the true ranking by performing maxing for $n-1$ rounds. In every round t, subroutine *Probe-Max* returns an item in S_t as the most preferred item (the maximum), where S_t denotes the set of remaining unranked items right before round t. The strategy of Probe-Max is to repeatedly apply SC on all item pairs. For every item pair (i,j), we initialize a global variable $\tau_{i,j}$ as the probing parameter for SC instances that run over i,j. The graph T storing obtained partial orders is also viewed as a global variable. Parameters $\tau_{i,j}$ and graph T will be accessed and altered in Probe-Max.

Algorithm 11 Probe-Rank

```
1: Input: items $[n]$, confidence level $\delta$
2: $S_1 = [n], Ans = [0]^n$, initialize $T$, $\tau_{i,j} = 1$ for all pairs of items $i \neq j$;
3: For $t = 1$ to $n-1$ do
4:     $i_{max}$ = Probe-Max($S_t, 2\delta/n^2$);
5:     remove $i_{max}$ from $T$; $Ans[t-1] = i_{max}$; $S_{t+1} = S_t \backslash \{i_{max}\}$;
6: $Ans[n-1] = S_n[0]$; Return $Ans$;
```

In Probe-Max(S,δ) (see Subroutine 12), SC instances are performed only on items that are possible to be the actual maximum. Let U be the set of maximal elements in T. By definition, every item in U has not lost to any other item in S yet. Assuming all previous comparison results (obtained form SC) are correct, to find the actual maximum, it suffices to focus on items in U. We use S^2 to denote the set of all unordered item pairs in S, i.e., $S^2 = \{(a,b) : a,b \in S, a \neq b\}$. All 'legitimate' pairs that can potentially provide us with information about the maximum item in S are thus

$$P = \{(i,j) : (i \in U \text{ or } j \in U), (i,j) \in S^2, (i,j) \notin T\}, \tag{3.4.1}$$

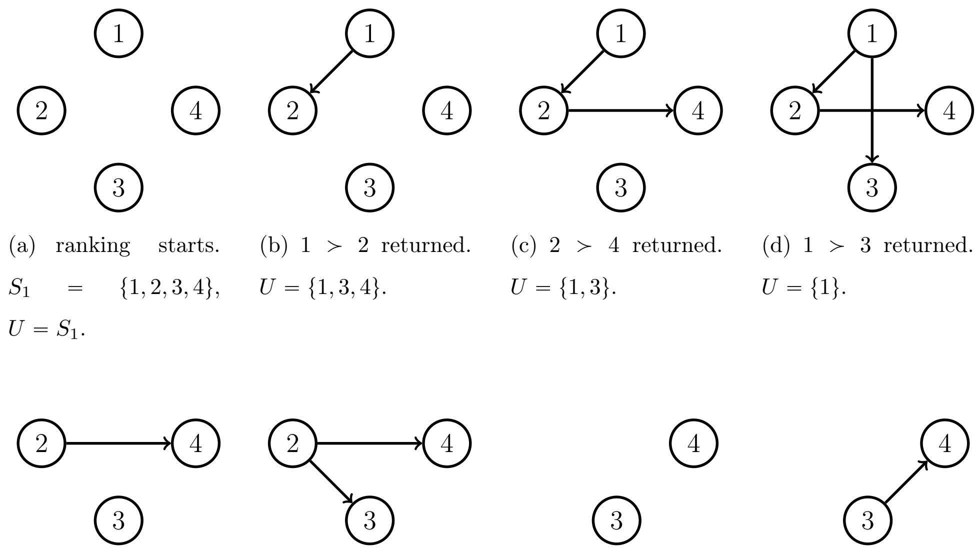
where $(i,j) \notin T$ means that nodes V_i and V_j are not connected in T. While U contains more than one items, Probe-Max keeps applying SC on item pairs in P. If an item in U loses a comparison, then we remove it from U. In every iteration of the while loop, the pairs (i^*, j^*) in P with the smallest τ value are chosen and S$C\left(i^*, j^*, \delta, \tau_{i^*, j^*}\right)$ are performed. Note that the τ value increases by one after each call of SC. Starting with item pairs with small τ values guarantees that we do not miss any useful information that can be obtained by paying only a small amount of comparisons.

Subroutine 12 Probe-Max(S, δ)

```
1: **Input:** set of unranked items $S$, S$C$ confidence level $\delta$
2: Let $U$ be the set of maximal elements according to $T$;
3: **While** $|U| > 1$ **do**
4:     Let $P = \{(i,j) : (i \in U \text{ or } j \in U), (i,j) \in S^2, (i,j) \notin T\}$;
5:     **For** $(a,b)$ in $argmin_{(x,y)\in P}\tau_{x,y}$ **do**
6:         $Ans$ = S$C\left(a, b, \delta, \tau_{a,b}\right)$; $\tau_{a,b} = \tau_{a,b} + 1$;
7:         **If** $Ans$ is not 'unsure' **then**
8:             $w, l = Ans$; $T = T \cup (w \succ l)$; **If** $|U| > 1$ and $l \in U$ **then** $U = U \backslash \{l\}$;
9: Return $U[0]$;
```

We provide a simple example demonstrating the ranking process.

Example 3.4.1. Consider items $\{1, 2, 3, 4\}$ with true ranking $1 \succ 2 \succ 3 \succ 4$. Figure 3.1 shows the status of T, U, S_t throughout the ranking process. In particular, we assume the pairwise comparison results are all correct and returned in order $1 \succ 2$, $2 \succ 4$, $1 \succ 3$, $2 \succ 3$, $3 \succ 4$.



(a) ranking starts. $S_1 = \{1, 2, 3, 4\}$, $U = S_1$.

(b) $1 \succ 2$ returned. $U = \{1, 3, 4\}$.

(c) $2 \succ 4$ returned. $U = \{1, 3\}$.

(d) $1 \succ 3$ returned. $U = \{1\}$.

(e) 1 is the maximum, remove it. $S_2 = \{2, 3, 4\}$, $U = \{2, 3\}$.

(f) $2 \succ 3$ returned. $U = \{2\}$.

(g) 2 is the maximum, remove it. $S_3 = \{3, 4\}$, $U = \{3, 4\}$.

(h) $3 \succ 4$ returned. $U = \{3\}$.

Figure 3.1: An illustration of the steps by Probe-Ranking, assuming true ranking as $1 \succ 2 \succ 3 \succ 4$.

3.5 Upper Bound on the Sample Complexity of Probe-Rank

In this section, we provide a sample complexity upper bound for the proposed algorithm Probe-Rank.

Theorem 2. *Let $\delta > 0$ be an arbitrary constant. For all problem instances satisfying the Weak Stochastic Transitivity (WST) property, with probability at least $1 - \delta$, Probe-Rank returns the true ranking of n items and conducts at most*

$$O\left(n \sum_{i=1}^{n} \left(\widetilde{\Delta}_i^{-2}\right) \left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(\frac{n}{\delta}\right)\right)\right) \tag{3.5.1}$$

comparisons, where $\widetilde{\Delta}_i$ is defined as in (3.2.1).

The proof of Theorem 2 is deferred to Section 3.9.

By the preceding theorem, the sample complexity of Probe-Rank is upper bounded by the sum of terms $(\widetilde{\Delta}_i)^{-2}(\log\log(\widetilde{\Delta}_i)^{-1} + \log(n/\delta))$ with an additional multiplicative factor of n. Recall from Section 3.2 that the term $(\widetilde{\Delta}_i)^{-2}(\log\log(\widetilde{\Delta}_i)^{-1} + \log(n/\delta))$ can be viewed as a lower bound on the number of comparisons that is needed for obtaining the order between i and its adjacent items with confidence level δ/n. Theorem 2 thus suggests that in Probe-Rank, every item is compared until it can be distinguished from its neighbors and no further. This matches with our intuition that only comparisons between adjacent items are necessary, and a single nonadjacent pair being extremely hard to distinguish should not harm the overall sample complexity. In contrast, sample complexities of existing algorithms are determined by the smallest probability gap between items, which can lead to a substantially large amount of unnecessary comparisons.

However, Probe-Rank achieves the dependence on $\widetilde{\Delta}_i$ instead of Δ_i at the cost of an additional multiplicative factor of n. Intuitively, because we have zero prior information about which items are adjacent and which are not, Probe-Rank pays $\Theta(n)$ attempts for each item i in order to 'identify' its neighbors and get the ordering feedback.

We compare Probe-Rank with the state-of-the-art IIR algorithm. Let $\mathcal{C}\,(\text{Probe})$ and $\mathcal{C}\,(\text{IIR})$ denote the sample complexities of two algorithms. From Table 3.1 and Theorem 2,

$$\mathcal{C}\,(\text{Probe}) = \sum_{i=1}^{n} \widetilde{\Theta}\left(n(\widetilde{\Delta}_i)^{-2}\right), \quad \mathcal{C}\,(\text{IIR}) = \sum_{i=1}^{n} \widetilde{\Theta}\left((\Delta_i)^{-2}\right), \tag{3.5.2}$$

noting that from the proofs, the sample complexity upper bounds are both tight in the worst case.

Under WST with no other conditions assumed, $\Delta_i \leqslant \widetilde{\Delta}_i$. In particular, when $\widetilde{\Delta}_i/\Delta_i = \Theta(\sqrt{n})$ for all i, then $\mathcal{C}\,(\text{Probe})$ and $\mathcal{C}\,(\text{IIR})$ are of the same asymptotic order with respect to n; if $\widetilde{\Delta}_i/\Delta_i = \omega(\sqrt{n})$, then Probe-Rank is asymptotically more sample-efficient than IIR. These phenomena are also reflected in our numerical experiments in Section 3.6 (see Figure 3.3).

Remark. It is worth noting that IIR is optimal if the more strict assumption SST as well as some other conditions are made, as shown in Ren et al. (2019). When SST holds, $\widetilde{\Delta}_i = \Delta_i$. Probe-Rank thus suffers from an additional factor of n. This case is also included in our numerical experiment (see Figure 3.2(a)).

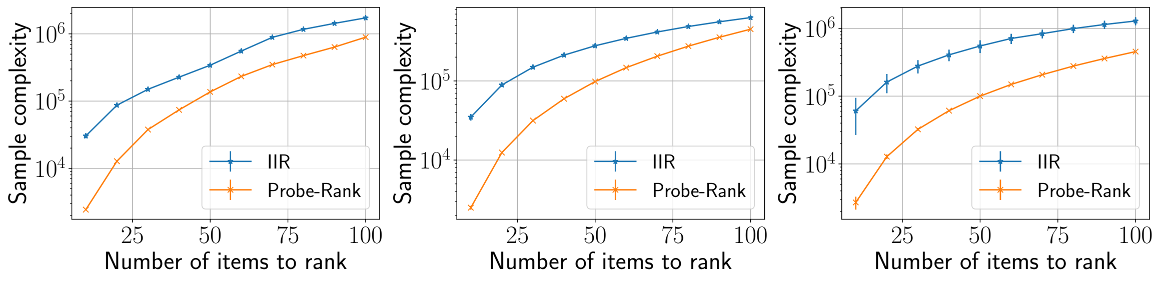
3.6 Experiments

In this section, we present numerical experiments demonstrating the practical performance of Probe-Rank. We compare Probe-Rank with the IIR algorithm, which was shown to outperform all the other baseline algorithms both theoretically and numerically (Ren et al., 2019). Our implementation can be found on Github [4].

We study different settings where SST is satisfied, not guaranteed, or violated, but WST always holds, which is consistent with our theory. Specifically, we want to rank n items with the true ranking $\sigma_1 \succ \sigma_2 \succ \cdots \succ \sigma_n$, where n varies over $[10, 100]$. The probabilistic comparison model p_{ij} is generated in different ways to satisfy different assumptions. Note that Δ and Δ_d are tuning parameters in all the following settings.

- `SST`: SST is satisfied. Comparison probabilities p_{ij} are generated from the MNL model, where $p_{\sigma_i,\sigma_j} = (\exp(s_{\sigma_i} - s_{\sigma_j}) + 1)^{-1}$, and $s_{\sigma_1}, \ldots, s_{\sigma_n}$ is a decreasing sequence where $s_{\sigma_i} = 100\Delta_d \cdot \frac{(n+1-i)}{n}$.
- `WST`: SST does not necessarily hold. Let $p_{i,j} \sim Uni(\frac{1}{2} + \Delta_d, 1)$ for all items $i \succ j$.
- `NON-SST`: SST does not hold. For adjacent items, we have $p_{\sigma_i,\sigma_{i+1}} \sim Uni\left(\frac{1}{2} + \Delta_d, 1\right)$. Otherwise, we have $p_{\sigma_i,\sigma_j} \sim Uni\left(\frac{1}{2} + \frac{\Delta_d}{10}, \frac{1}{2} + \Delta_d\right)$ for $j > i + 1$.
- `ADJ-ASYM`: SST does not hold. This setting is used to verify the asymptotic analysis in Section 3.5. For adjacent items, we set $p_{\sigma_i,\sigma_{i+1}} = \frac{1}{2} + \Delta_d$. Otherwise, we set

[4]https://github.com/tao-j/aht/releases/tag/v0.1



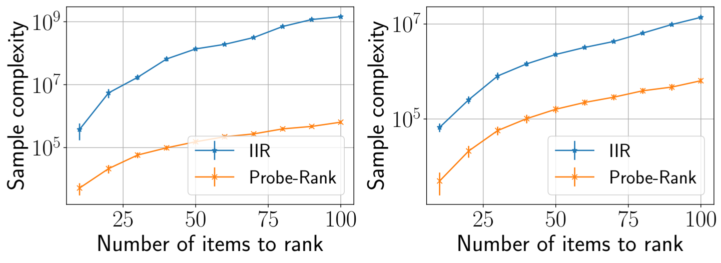
(a) SST: $\Delta_d = 0.3$ (b) WST: $\Delta_d = 0.3$ (c) NON-SST: $\Delta_d = 0.3$

Figure 3.2: Comparison of sample complexities of Probe-Rank and IIR under various settings. In each subfigure, Δ_d is fixed while the number of items varies.

$p_{\sigma_i,\sigma_j} = \frac{1}{2} + \frac{\Delta_d}{n^\alpha}$ for $j > i + 1$. We consider cases where α equals 0.5 or 1.

- `ADJ-CNST`: SST does not hold. For adjacent items, we set $p_{\sigma_i,\sigma_{i+1}} = \frac{1}{2} + \Delta$. Otherwise $p_{\sigma_i,\sigma_j} = \frac{1}{2} + \Delta_d$ for $j > i + 1$. Here $\Delta > \Delta_d$.

All experiments are averaged over 100 independent trials. For each trial, the ground truth ranking σ is generated uniformly at random and the comparison probabilities are assigned accordingly. The confidence level δ is fixed to be 0.1. Throughout the experiment, every trial for every algorithm successfully recovered the correct ranking.



(a) ADJ-ASYM: $\Delta_d = 0.3$, $\alpha = 1$ (b) ADJ-ASYM: $\Delta_d = 0.3, \alpha = 0.5$

Figure 3.3: Relationship between n and gap Δ_d

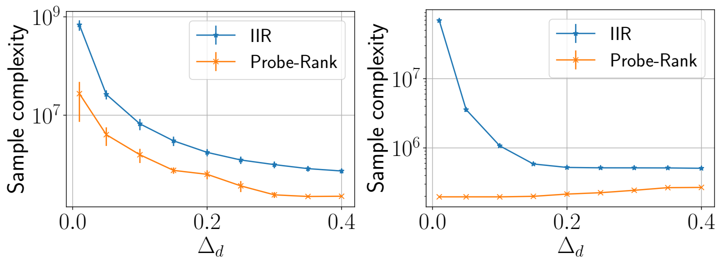
We use internal clusters of intel "Skylake" generation CPUs. Each job contains a single model type for item numbers ranging from 10 to 100 with a step size of 10. Models are generated from a job unique random seed shared among the two algorithms. Most jobs with sample complexity smaller than 10^7 terminate in 3 minutes. For $\Delta_d = 0.1$ under the

ADJ-ASYM model, 3 hours are needed due to high sample complexity. Due to the space limit, more detailed experimental setups and thorough ablation studies can be found in Section 3.11.

Performance Comparison Figure 3.2 with y-axis in log-scale shows comparison of IIR and Probe-Ranking under the SST, WST and NON-SST settings. The parameter Δ_d is set to be 0.3. It can be seen that under the SST and WST settings (Figures 3.2(a), 3.2(b)), Probe-Rank consumes less samples than IIR for small n. As n gets larger, however, IIR becomes more sample-efficient due to that Probe-Rank has an additional factor of n in its sample complexity compared with IIR for instances satisfy SST. However, under the NON-SST setting where SST does not hold, Probe-Rank has a clear advantage over IIR, as shown in Figure 3.2(c).

Dependence on n and the Probability Gaps Following Theorem 2, we verify that the sample complexity of Probe-Rank is lower than IIR when the number of items n gets larger. We use the

ADJ-ASYM setting to simulate situations where nonadjacent items can be much more difficult to compare. In particular, we choose $\alpha = 1$ (see Figure 3.3(a)) and $\alpha = 1/2$ (see Figure 3.3(b)). It can be seen from Figure 3.3(a) that as the number of items n gets larger, the gap between the two curves also gets larger. This matches our analysis that when $\widetilde{\Delta}_i/\Delta_i = \omega(\sqrt{n})$, then the sample complexity of IIR is of higher order than that of Probe-Rank. When $\widetilde{\Delta}_i/\Delta_i = \Theta(\sqrt{n})$, Figure 3.3(b) shows



(a) NON-SST: $n = 80$ (b) ADJ-CNST: $n = 80$, $\Delta = 0.4$

Figure 3.4: Ablation study on the dependence of the sample complexity on the probability gap Δ_d.

that the gap between the two sample complexities varies little as n increases. Our analysis also suggests that sample complexities of two algorithms are of the same order.

Furthermore, we show through the `NON-SST` and `ADJ-CNST` settings that when the probability gaps of nonadjacent item pairs decrease, the advantage of our algorithm will be more and more prominent.

In Figure 3.4, we fix $n = 80$ and let Δ_d vary. Clearly, Probe-Rank has an advantage over IIR in both settings. In particular, Figure 3.4(b) shows the comparison of two algorithms in the `ADJ-CNST` setting with the probability gaps between adjacent items Δ fixed as 0.4. As the probability gap between nonadjacent items Δ_d varies from 0.01 to 0.4, it can be seen that the sample complexity of Probe-Rank does not vary much. However, the sample complexity of IIR has a positive correlation with $\frac{1}{\Delta_d^2}$. This numerical result matches our analysis that Probe-Ranking is not affected by the comparison probability of nonadjacent items, which does not hold for IIR.

3.7 Discussion on the Lower Bound

In this section, we provide some insights about the lower bound for pairwise ranking by proposing a conjecture based on a particularly hard instance $\mathcal{I}_{WST}$ that satisfies the WST condition.

Problem 1 ($\mathcal{I}_{WST}$)**.** The problem instance $\mathcal{I}_{WST}$ is constructed as follows. Consider n items with an underlying ordering '$\succ$'. For all $i \succ j$,

$$p_{i,j} = \begin{cases} \frac{1}{2} + \Delta, & \text{if } i \text{ and } j \text{ are adjacent,} \\ \frac{1}{2} + cn^{-10}\Delta^2/\log(1/\delta), & \text{otherwise,} \end{cases}$$

where c and Δ are constants and n^{-10} can be replaced by any other quantity that is smaller than n^{-2}.

By a reduction, any δ-correct algorithm that finds the maximum item for $\mathcal{I}_{WST}$ can be constructed to find the maximum item for $\mathcal{I}_{SNG}$, described below in Problem 2. Therefore, a lower bound on the sample complexity for maxing in Problem 2 will imply a lower bound of the same order for the maxing (and thus, ranking) problem for $\mathcal{I}_{WST}$. This lower bound is also a worst-case lower bound for ranking under WST. In the following, we provide an analysis for Problem 2. The reduction technique will be deferred to Section 3.12.

Problem 2 ($\mathcal{I}_{SNG}$)**.** Consider n items with an underlying ordering '$\succ$'. One can make queries of the form 'if $i \succ j$'. The feedback $Y_{i,j}$ is a binary random variable which takes value 1 if the answer is YES and takes value 0 otherwise. The random variables $Y_{i,j}$ are defined to follow distributions:

$$Y_{i,j} \sim \begin{cases} \mathrm{Ber}(\frac{1}{2} - 2\Delta), & \text{if } i \prec j \text{ and } i, j \text{ are adjacent,} \\ \mathrm{Ber}(\frac{1}{2}), & \text{otherwise.} \end{cases}$$

Consider random vectors defined by $\boldsymbol{p}_i = (Y_{i,1}, Y_{i,2}, \ldots, Y_{i,n})$ in Problem 2. The maximum element i^* corresponds to the random vector $\boldsymbol{p}_{i^*}$, where each entry is a 1/2-Bernoulli random variable. For every other non-maximum element i, $\boldsymbol{p}_i$ will contain exactly one $(1/2 - 2\Delta)$-Bernoulli random variable. Under such problem setting, finding the maximum item is equivalent to finding which vector has all its entries as 1/2-Bernoulli random variables.

We conjecture that any δ-correct algorithm that can find the maximum item for $\mathcal{I}_{SNG}$ has a sample complexity at least

$$\Omega\big(n^2 \Delta^{-2} \log(1/\delta)\big). \tag{3.7.1}$$

We start from viewing it as a hypothesis testing problem. Consider that an agent is asked to determine if $\boldsymbol{p}_1$ satisfies hypothesis H_0, defined as

$$H_0 : \boldsymbol{p}_1 = (p_{1,1}, \ldots, p_{1,n}), \text{ where } p_{1,k} \sim \mathrm{Ber}(1/2), \forall k \in [n],$$

or H_j, in which the j-th entry is biased:

$$H_j : \boldsymbol{p}_1 = (p_{1,1}, \ldots, p_{1,n}), \text{ where } p_{1,j} \sim \mathrm{Ber}(1/2 - 2\Delta), p_{1,k} \sim \mathrm{Ber}(1/2), \forall k \neq j.$$

Suppose the hypothesis testing algorithm $\mathcal{A}$ is δ-correct and stops within T rounds of interactions. We denote $\mathcal{A}(T)$ as the output at the T-th round, which is either 0 (accept H_0) or 1 (reject H_0). For any given $j \neq 1$, by the Bretagnolle–Huber inequality, we have

$$2\delta \geqslant \mathbb{P}_0(\mathcal{A}(T) \neq 0) + \mathbb{P}_j(\mathcal{A}(T) = 0) \geqslant \frac{1}{2} e^{-\mathrm{KL}(\mathbb{P}_0^{\mathcal{A}} || \mathbb{P}_j^{\mathcal{A}})}, \tag{3.7.2}$$

where $\mathbb{P}_0$ is the probability measure under H_0, and $\mathbb{P}_0^{\mathcal{A}}$ is the probability measure of the canonical bandit model under H_0. In fact, we have the divergence decomposition lemma (Lattimore and Szepesvári, 2020, Lemma 15.1):

$$\mathrm{KL}(\mathbb{P}_0^{\mathcal{A}} || \mathbb{P}_j^{\mathcal{A}}) = \sum_{k=1}^{n} \mathbb{E}_0[N_k(T)] \mathrm{KL}(\mathbb{P}_{0,k} || \mathbb{P}_{j,k}) = \mathbb{E}_0[N_j(T)] \mathrm{KL}(\mathrm{Ber}(1/2) || \mathrm{Ber}(1/2 - 2\Delta)), \tag{3.7.3}$$

where $\mathbb{E}_0$ denotes the expectation under H_0; $\mathbb{E}_0[N_k(T)]$ denotes under H_0, the expected number of queries for the entry $p_{1,k}$ within T rounds.; $\mathbb{P}_{0,k}$, $\mathbb{P}_{j,k}$ are the Bernoulli distributions specified by $p_{1,k}$ under H_0, H_j, respectively. The second equality is due to the fact that the only difference between H_0 and H_j is that the j-th entry has different Bernoulli distributions.

Combining the two inequality above gives:

$$\mathbb{E}_0[N_j(T)] \mathrm{KL}(\mathrm{Ber}(1/2) || \mathrm{Ber}(1/2 - 2\Delta)) \geqslant \log(1/4\delta). \tag{3.7.4}$$

Since $\mathrm{KL}(\mathrm{Ber}(1/2)||\mathrm{Ber}(1/2 - x)) < (4x)^2$ for all $x < 2/9$, we get $\mathbb{E}_0[N_j(T)] = \Omega(\Delta^{-2} \log(1/\delta))$. Thus, the total expected number of queries under H_0 will be $\Omega(n\Delta^{-2} \log(1/\delta))$.

In Problem 2, there are in total n vectors. We reasonably conjecture that to identify which vector satisfies H_0 requires at least $\Omega(n)$ attempts, with each attempt costs $\Omega(n\Delta^{-2} \log(1/\delta))$, i.e, any δ-correct algorithm requires $\Omega\big(n^2 \log(1/\delta)/\Delta^2\big)$ queries.

3.8 Conclusion and Future Work

In this chapter, we studied the problem of exact ranking under the most general assumption WST. We proposed a δ-correct algorithm Probe-Rank, and derived an instance-wise upper bound on its sample complexity. The upper bound shows that the performance of Probe-Rank only depends on the comparison probabilities of adjacent items and thus improves existing results when SST does not hold. Numerical results also suggest that our ranking algorithm outperforms the state-of-the-art. A discussion over the lower bound for pairwise ranking is also provided. We propose a conjecture that in the worst case, any algorithm has sample complexity n times the number of comparisons needed for comparing all adjacent items. However, it remains an open problem whether our conjecture holds and will be left to future work.

3.9 Proof of the Sample Complexity Upper Bound on Probe-Rank

In this section, we prove our theoretical upper bound presented in Theorem 2, Section 3.5.

We first show in the following lemma that the subroutine Successive-Comparison returns desired outcomes with high probability. Given an item pair (i, j) with probability gap $\Delta_{i,j} > 0$ and a positive integer τ, we say S$C\,(i, j, \delta, \tau)$ is successful if one of the following two events holds,

$$\mathcal{E}_1 = \{\Delta_{i,j} \geqslant \epsilon_\tau \text{ and S}C \text{ correctly returns } [i, j]\}, \tag{3.9.1}$$

$$\mathcal{E}_2 = \{\Delta_{i,j} < \epsilon_\tau \text{ and S}C \text{ returns `unsure' or } [i, j]\}. \tag{3.9.2}$$

Lemma 3. *For an item pair (i, j) with probability gap $\Delta_{i,j} > 0$ and a positive integer τ, S$C\,(i, j, \delta, \tau)$ is successful with probability at least $1 - \frac{\delta}{c\tau^2}$, where $c = \frac{\pi^2}{6}$.*

Proof of Lemma 3. Hoeffding's inequality gives that

$$\Pr\left(\widehat{p}_i - p_{i,j} \leqslant -\frac{1}{2}\epsilon_\tau\right) \leqslant \exp\left(-2b_\tau\left(\frac{1}{2}\epsilon_\tau\right)^2\right) \leqslant \frac{\delta}{c\tau^2}. \tag{3.9.3}$$

Therefore, the probability that SC outputs $[j, i]$ is at most

$$\Pr\left(\widehat{p}_i - \frac{1}{2} < -\frac{1}{2}\epsilon_\tau\right) \leqslant \Pr\left(\widehat{p}_i - p_{i,j} \leqslant -\frac{1}{2}\epsilon_\tau\right) \leqslant \frac{\delta}{c\tau^2}, \tag{3.9.4}$$

and the probability that SC returns $[i, j]$ or 'unsure' is at least $1 - \frac{\delta}{c\tau^2}$.

Further, if $\Delta_{i,j} \geqslant \epsilon_\tau$, the probability that SC returns $[i, j]$ is at least

$$\Pr\left(\widehat{p}_i - \frac{1}{2} > \frac{1}{2}\epsilon_\tau\right) = \Pr\left(\widehat{p}_i > \frac{1}{2} + \frac{1}{2}\epsilon_\tau\right) \geqslant \Pr\left(\widehat{p}_i > p_{i,j} - \frac{1}{2}\epsilon_\tau\right) \geqslant 1 - \frac{\delta}{c\tau^2}. \quad (3.9.5)$$

This completes the proof. □

By Lemma 3, with high probability, SC does not return the incorrect ordering. Further, if τ is large enough, then SC is guaranteed to return the correct ordering. We use Lemma 3 to show the theoretical performance of Probe-Rank.

Theorem 2. *Let $\delta > 0$ be an arbitrary constant. For all problem instances satisfying the Weak Stochastic Transitivity (WST) property, with probability at least $1 - \delta$, Probe-Rank returns the true ranking of n items and conducts at most*

$$O\left(n\sum_{i=1}^{n}\left(\widetilde{\Delta}_i^{-2}\right)\left(\log\log\left(\widetilde{\Delta}_i^{-1}\right) + \log\left(\frac{n}{\delta}\right)\right)\right) \quad (3.5.1)$$

comparisons, where $\widetilde{\Delta}_i$ is defined as in (3.2.1).

Proof of Theorem 2. Define events

$$\mathcal{E}_{i,j}(\tau) = \{\text{SC}\left(i, j, 2\delta/n^2, \tau\right) \text{ is successful}\}. \quad (3.9.6)$$

Define the bad event

$$\mathcal{E}^{bad} = \cup_{(i,j)\in[n]^2} \cup_{\tau=1}^{\infty} \left(\mathcal{E}_{i,j}(\tau)\right)^c. \quad (3.9.7)$$

By the union bound and Lemma 3

$$\Pr\left(\mathcal{E}^{bad}\right) \leqslant \sum_{(i,j)\in[n]^2}\sum_{\tau=1}^{\infty}\frac{2\delta}{cn^2\tau^2} \leqslant \sum_{\tau=1}^{\infty}\frac{\delta}{c\tau^2} \leqslant \delta. \quad (3.9.8)$$

In the following, we assume that $\mathcal{E}^{bad}$ does not happen.

Correctness. We show that when $\mathcal{E}^{bad}$ does not happen, in every round t, Probe-Max$(S_t, 2\delta/n^2)$ (line 4 of Algorithm 11) correctly returns the most preferred item in the set of remaining items S_t. Since the probability of $\mathcal{E}^{bad}$ is upper bounded by δ, the correctness of Probe-Rank thus follows.

Let x be the most preferred item in S_t. When $\mathcal{E}^{bad}$ does not happen, all comparison results returned by SC are correct and T is always consistent with the true ranking. Thus, no item in S_t is known to rank higher than x, i.e., at the beginning of Subroutine 12, $x \in U$. Moreover, x will not be eliminated from U since x will not lose to any other item in S_t during calls of SC.

We show that any other item in U will be eliminated from U after a finite number of iterations of the while loop in Probe-Max. Let $y \neq x$ be an item in U. Since x is the maximum, $y \prec x$ in the true ranking. Whenever $\epsilon_{\tau_{y,x}} \leqslant \Delta_{x,y}$, a successful call of S$C\left(x, y, 2\delta/n^2, \tau_{x,y}\right)$ will return the result $x \succ y$ and remove y from U if $\mathcal{E}^{bad}$ does not happen. Since $\epsilon_{\tau_{y,x}}$ converges to 0, there must exist $\tau^*_{x,y}$ such that $\epsilon_{\tau^*_{x,y}} \leqslant \Delta_{x,y}$. After each execution of SC, the corresponding τ value increases by one, therefore after at most $\binom{n}{2}\tau^*_{x,y}$ iterations of the while loop, S$C\left(x, y, 2\delta/n^2, \tau^*_{x,y}\right)$ must have been called. The same argument holds for any $y \in U, y \neq x$.

Sample complexity. We first note the asymptotic behavior that for any $N > 0$,

$$\sum_{\tau=1}^{N} b_\tau \leqslant \sum_{\tau=1}^{N} \frac{2}{4^{-\tau}} \log \frac{c\tau^2 n^2}{\delta} \leqslant \sum_{\tau=1}^{N} \frac{2}{4^{-\tau}} \log \frac{cN^2 n^2}{\delta} = O\left(4^N \log \frac{cN^2\delta^2}{\delta}\right) = O\left(b_N\right). \quad (3.9.9)$$

Without loss of generality, we assume the true ranking is $1 \succ 2 \succ \cdots \succ n$. When $\mathcal{E}^{bad}$ does not happen, all comparison results returned by SC coincide with the true ranking. Therefore, for every $i \in [n-1]$, item i belongs to $S_1, S_2, \ldots, S_i$ and gets eliminated during the execution of Probe-Max$(S_i, 2\delta/n^2)$.

Recall that SC is only called over item pairs in which at least one of them is a maximal element. For every SC called on items a, b, if a is maximal, we say item a initializes the comparison and we charge the number of comparisons taken by SC to item a (if both a and b are maximal, we charge the number of samples to both). Let $c(a)$ denote the number of comparisons charged to a. The total sample complexity of Probe-Rank is thus at most

$\sum_{a\in[n]} c(a)$.

Fix $i \in [n]$. We use τ_i° to denote the value of $\tau_{i,i-1}$ when the order between i and $i-1$ is revealed. Define $\tau_1^\circ = 0$ for completeness. We note that the order between i and $i-1$ can not be inferred from any other comparison results therefore can only be returned by S$C\left(i, i-1, 2\delta/n^2, \tau_i^\circ\right)$. When $\mathcal{E}^{bad}$ does not happen, $\tau_i^\circ \leqslant \lceil \log \frac{1}{\Delta_{i,i-1}} \rceil$ since a successful call of S$C\left(i, i-1, 2\delta/n^2, \lceil \log \frac{1}{\Delta_{i,i-1}} \rceil\right)$ will return the order.

For each $j \neq i$, we use $\tau_{i,j}^*$ to denote the value of $\tau_{i,j}$ when the last time SC is initialized by i and called over i, j before the beginning of Probe-Max$(S_i, 2\delta/n^2)$. In other words, for any $\tau > \tau_{i,j}^*$, if S$C\left(i, j, 2\delta/n^2, \tau\right)$ is called in Probe-Max$(S_t, 2\delta/n^2)$ for some $t < i$, then it must not be initialized by i. Moreover, we use $\tau_{i,j}^t$ to denote the value of $\tau_{i,j}$ right after Probe-Max$(S_t, 2\delta/n^2)$ terminates. Since i is ranked and removed from T after Probe-Max$(S_i, 2\delta/n^2)$ is called, $\tau_{i,j}^i$ is also the value of $\tau_{i,j}$ when Probe-Rank terminates. It is clear that

$$c(i) \leqslant \sum_{j\neq i} \sum_{\tau=1}^{\tau_{i,j}^*} b_\tau + \sum_{j \neq i} \sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i} b_\tau. \tag{3.9.10}$$

We consider the first term on the right-hand side of (3.9.10). Before Probe-Max$(S_{i-1}, 2\delta/n^2)$ terminates, item $i-1$ is in T. Therefore, whenever i is a maximal element, the order between i and $i-1$ must have not been revealed. So when i initializes the comparison S$C\left(i, j, 2\delta/n^2, \tau_{i,j}^*\right)$, the item pair $(i, i-1)$ is also in the set of 'legitimate' pairs P. Therefore, $\tau_{i,j}^*$ is no larger than the value of $\tau_{i,i-1}$ at that point, and further no larger than τ_i°. The same argument holds for any j. It follows that

$$\sum_{j\neq i} \sum_{\tau=1}^{\tau_{i,j}^*} b_\tau \leqslant \sum_{j\neq i} \sum_{\tau=1}^{\tau_{i,j}^*} b_\tau \leqslant \sum_{\tau=1}^{\tau_i^\circ} n b_\tau. \tag{3.9.11}$$

Next, we bound the second term on the right-hand side of (3.9.10). Note that if there is no SC called during Probe-Max$(S_i, 2\delta/n^2)$, then $\sum_{j\neq i} \sum_{\tau=\tau_{i,j}^{i-1}+1}^{\tau_{i,j}^i} b_\tau = 0$. So it suffices to consider the case when at least one instance of SC is called during Probe-Max$(S_i, 2\delta/n^2)$. Consider the last group of SC called in Probe-Max$(S_i, 2\delta/n^2)$, here group means that there might be multiple item pairs whose τ values are the minimum in P. Denote their τ values by τ^i. There must be some $\mathrm{S}C\left(a_i, b_i, 2\delta/n^2, \tau^i\right)$ returning $b_i \succ a_i$ such that a_i is a maximal item, otherwise no maximal item is removed from U and Probe-Max will not terminate. When $\mathcal{E}^{bad}$ does not happen, a_i is not the maximum in S_i so $a_i > i$. Thus, item $a_i - 1$ is also in S_i and before the call of $\mathrm{S}C\left(a_i, b_i, 2\delta/n^2, \tau^i\right)$, the ordering between $a_i - 1$ and a_i is not revealed, i.e., $\tau^i \leqslant \tau^{\circ}_{a_i}$. Moreover, $\tau^i_{i,j} \leqslant \tau^i$ by the fact that we always compare item pairs with the smallest τ values. It follows that

$$\sum_{j \neq i} \sum_{\tau=\tau^{i-1}_{i,j}+1}^{\tau^i_{i,j}} b_\tau \leqslant n \sum_{\tau=1}^{\tau^i} b_\tau = O\left(n b_{\tau^i}\right). \tag{3.9.12}$$

The same argument holds for all $i \in [n-1]$.

Consider the sets

$$\mathcal{D}_1 = \{b_{\tau^i} : i = 1, 2, \ldots, n-1\}, \quad \mathcal{D}_2 = \cup_{i=2}^n \mathcal{D}_2^i = \cup_{i=2}^n \{b_\tau : \tau = 1, 2, \ldots, \tau^{\circ}_i\}. \tag{3.9.13}$$

We claim that if $i_1 \neq i_2$, then the pairs (a_{i_1}, τ^{i_1}) and (a_{i_2}, τ^{i_2}) do not equal. With the facts that $a_i > i$ and $\tau^i \leqslant \tau^{\circ}_{a_i}$, there is an injective mapping from $\mathcal{D}_1$ to $\mathcal{D}_2$ given by b_{τ^i} is mapped to the element b_{τ^i} in $\mathcal{D}_2^{a_i}$. It follows that

$$\sum_{i=1}^{n-1} O\left(n b_{\tau^i}\right) = O\left(\sum_{x \in \mathcal{D}_1} nx\right) \leqslant O\left(\sum_{x \in \mathcal{D}_2} nx\right) = O\left(\sum_{i=2}^{n} \sum_{\tau=1}^{\tau^{\circ}_i} n b_\tau\right). \tag{3.9.14}$$

The reason for pairs (a_{i_1}, τ^{i_1}) and (a_{i_2}, τ^{i_2}) equal if and only if $i_1 = i_2$ is as follows. Let $i_2 > i_1$ and suppose $a_{i_1} = a_{i_2} = a$. When $\mathrm{S}C\left(a, b_{i_1}, 2\delta/n^2, \tau^{i_1}\right)$ is called, $\mathrm{S}C\left(a, b, 2\delta/n^2, \tau^{i_1}\right)$ for all b such that $(a, b) \notin T$ and $\tau_{a,b} = \tau^{i_1}$ are also called. It follows that $\tau_{a,b} > \tau^i$ for all such b after this point. When $\mathrm{S}C\left(a, b_{i_2}, 2\delta/n^2, \tau^{i_2}\right)$ is called, the order between a and b_{i_2} is not know and thus also not known when $\mathrm{S}C\left(a, b_{i_1}, 2\delta/n^2, \tau^{i_1}\right)$ was called. So τ^{i_2} must be larger than τ^{i_1}.

Combining (3.9.10), (3.9.11) and (3.9.14) gives,

$$\sum_{i=1}^{n} c(i) \leqslant \sum_{i=2}^{n}\sum_{j\neq i}\sum_{\tau=1}^{\tau^*_{i,j}} b_\tau + \sum_{i=1}^{n-1}\sum_{j\neq i}\sum_{\tau=\tau^{i-1}_{i,j}+1}^{\tau^{i}_{i,j}} b_\tau \tag{3.9.15}$$

$$\leqslant O\left(\sum_{i=2}^{n}\sum_{\tau}^{\tau_i^\circ} n b_\tau\right) = O\left(n\sum_{i=2}^{n} b_{\tau_i^\circ}\right). \tag{3.9.16}$$

The desired sample complexity follows from $\tau_i^\circ \leqslant \lceil \log \frac{1}{\Delta_{i,i-1}} \rceil$ and

$$b_{\lceil \log \frac{1}{\Delta} \rceil} = O\left(\frac{1}{\Delta^2}\left(\log\log\frac{1}{\Delta} + \log\frac{n}{\delta}\right)\right), \tag{3.9.17}$$

which completes the proof. □

3.10 A Sample-Efficient Variant of Probe-Rank

In this section, we present a variant of Probe-Rank, named *Probe-Rank-SE*. When demonstrating more detailed experiments in Section 3.11, Probe-Rank-SE is also included and is shown to have better practical performance. However, we will not prove its correctness due to the high similarity it shares with Probe-Rank.

Compared with Probe-Rank, the variant Probe-Rank-SE finds the ranking also by performing $n-1$ steps of maxing and differs only in the subroutine for collecting comparison samples. Specifically, Probe-Rank-SE takes queries from all unknown item pairs simultaneously. Comparison results for pairs that terminate earlier are still collected and stored in the graph T, which represents our current knowledge about the ranking. We use T to decide whether to pause, drop or resume comparisons of remaining item pairs.

We adopt the Successive Elimination (SE) algorithm from Even-Dar et al. (2002a), shown in Algorithm 13, as a procedure to perform comparisons.

Subroutine 13 Successive Elimination (modified for comparing two items)

```
1: Input: items $i, j$, confidence level $\delta$
2: $t = 1$;
3: while true do
4:     Compare $i$ and $j$ for $2^t$ times; Let $\widehat{p}_i^t$ be the winning rate of $i$;
5:     Let $\alpha_t = \sqrt{\frac{\log(ct^2/\delta)}{2^t}}$, $c = \frac{\pi^2}{3}$;
6:     Return $i \succ j$ if $\widehat{p}_i^t - \frac{1}{2} > \alpha_t$; Return $j \succ i$ if $\widehat{p}_i^t - \frac{1}{2} < -\alpha_t$; $t = t + 1$ else;
7: end while
```

It was shown that with probability at least $1 - \delta$, Subroutine 13 correctly returns the more preferred item between i and j using at most $O\left(\frac{1}{\Delta_{i,j}^2}\left(\log \frac{1}{\delta} + \log\log \frac{1}{\Delta_{i,j}}\right)\right)$ comparisons (Even-Dar et al., 2002a, Remark 1).

In Probe-Rank-SE, we do not call SE directly, rather, SE is used as a black-boxed unit that repeatedly collects query samples from the input pair i, j. Moreover, after every sample, it generates feedback which is either Null, $i \succ j$ or $j \succ i$, where Null corresponds to that the number of samples has not accumulated to 2^t or $\left|\widehat{p}_i^t - \frac{1}{2}\right| < \alpha_t$; feedback $i \succ j$ and $j \succ i$ correspond to that inside the black box, SE actually terminates and returns the order between i and j. Note that the SE procedure can be replaced by any algorithm that can rank two items, including all best-arm-identification algorithms.

Denote the instance of Successive Elimination that runs over items i, j with confidence level δ as $\mathrm{S}E_{i,j}(\delta)$. When the value of δ is given without ambiguity, we will drop the dependence and write $\mathrm{S}E_{i,j}$ as a shorthand. We define two operations on $\mathrm{S}E_{i,j}$, named a*dvance* and f*eed*. The a*dvance* operation returns one of the three possible internal outcomes, Null, $i \succ j$ or $j \succ i$. The f*eed* operation is used for simulating the sampling process. We write f*eed* $(\mathrm{S}E_{i,j}, Y_{i,j})$ to represent that $\mathrm{S}E_{i,j}$ is fed with a comparison sample $Y_{i,j}$. As a black-boxed unit, before a*dvance* returns one of $i \succ j$ and $j \succ i$, a*dvance* and f*eed* operations are invoked in an alternating fashion. The idea of viewing a sampling subroutine as a black-box controlled by artificial operations was also used in Ailon et al. (2014), but for a different problem setting.

Probe-Rank-SE is presented in Algorithm 14. We initialize $\binom{n}{2}$ independent instances of $\mathrm{S}E_{i,j}\left(2\delta/n^2\right)$, each for obtaining the order between an item pair $(i,j), 1 \leqslant i < j \leqslant n$. The probability of being unable to recover the true ranking is thus upper bounded by probability that at least one of the SE instances fails, which is at most δ. Same as Probe-Rank, we use T to denote the transitive closure composed of results returned by the $\mathrm{S}E$ instances.

Algorithm 14 Probe-Rank-SE

```
1: **Input:** items $[n]$, confidence level $\delta$
2: $S_1 = [n], Ans = [0]^n$; initialize $T$;
3: initialize $\mathrm{S}E_{i,j}\left({}^{2\delta}/_{n^2}\right)$ for all $1 \leqslant i < j \leqslant n$;
4: **for** t from 1 to $n-1$ **do**
5:     $i_{max}$ =Probe-Max-SE($S_t$);
6:     remove $i_{max}$ from $T$; $Ans[t-1] = i_{max}$; $S_{t+1} = S_t \backslash \{i_{max}\}$;
7: **end for**
8: $Ans[n-1] = S_n[0]$; Return $Ans$;
```

The procedure Probe-Max-SE serves as a switch for the $\mathrm{S}E$ instances. Let S_t^2 denote the set of unordered item pairs $\{(i,j) : i,j \in S_t, i \neq j\}$. In each round t, all $\mathrm{S}E$ instances for 'legitimate' pairs in S_t^2 are turned on and take queries in a round-robin fashion. 'Legitimate' pairs are similarly defined as in Probe-Rank. A pair (i,j) is 'legitimate' if the order between i,j is unknown, i.e., not in T, and at least one of i and j is a maximal element in S_t.

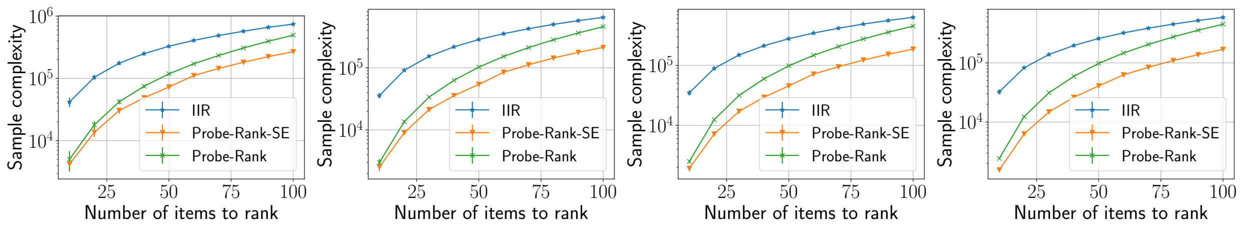
Algorithm 15 Probe-Max-SE(S_t)

```
 1: Let $U$ be sets of maximal elements according to $T$
 2: while $|U| \geqslant 1$ do
 3:     $C = [\ ]$
 4:     for $(i, j)$ in $S_t^2$ do
 5:         if $(i \in U$ or $j \in U)$ and $(i, j) \notin T$ then
 6:             compare $i$ with $j$ once and get result $Y_{i,j}$; $feed\left(\mathrm{S}E_{i,j}\left(\delta/n^2\right), Y_{i,j}\right)$
 7:             if $advance\left(\mathrm{S}E_{i,j}\left(2\delta/n^2\right)\right) == i \succ j$ then
 8:                 $C.append([i, j])$;
 9:             else if $advance\left(\mathrm{S}E_{i,j}\left(2\delta/n^2\right)\right) == j \succ i$ then
10:                 $C.append([j, i])$;
11:             end if
12:         end if
13:     end for
14:     for $w, l$ in $C$ do
15:         if $(w, l) \notin T$ then
16:             $T = T \cup (w \succ l)$;
17:             if $|U| > 1$ and $l \in U$ then
18:                 $U = U \backslash \{l\}$;
19:             end if
20:         end if
21:     end for
22: end while
23: Return $U[0]$;
```

3.11 Additional Experiments

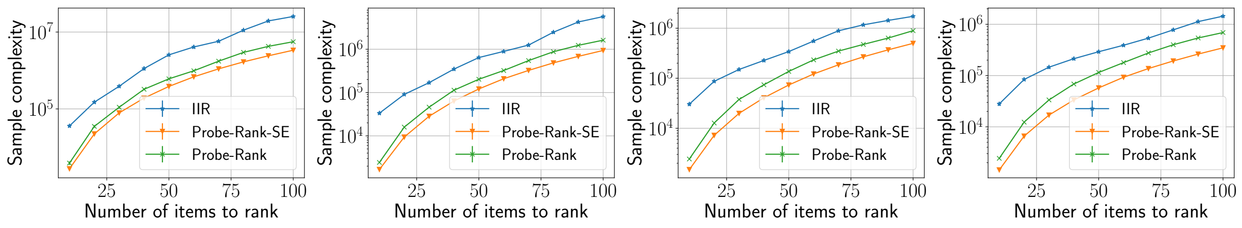
In this section, we present more detailed numerical experiments comparing the sample complexities of Probe-Rank, Probe-Rank-SE and the state-of-the-art algorithm IIR by Ren et al. (2019). In particular, we focus on the `WST`, `SST`, `NON-SST` and `ADJ-ASYM` settings and perform these three algorithms with various parameters. Same as the results presented in Section 3.6, all experiments are averaged over 100 independent trials. For each trial, the ground truth ranking σ is generated uniformly at random and the comparison probabilities are assigned according to the chosen setting. The confidence level δ is fixed to be 0.1. Throughout the experiment, every trial for every algorithm successfully recovered the correct ranking. Moreover, for IIR, if the rank has not been recovered after the sample complexity reaches 10^9, we manually stop the ranking process and record the sample complexity as 10^9 to avoid extremely large running times. Note that the extreme cases happen in Figures 3.8(a), 3.8(b) 3.8(c) and 3.12(d).

Figures 3.5, 3.6, 3.7 and 3.8 compare the three algorithms under different settings where the difficulty parameter Δ_d is fixed and the number of items n varies from 10 to 100. Figures 3.9, 3.10, 3.11 and 3.12 compare the three algorithms under different settings where the number of items n is fixed and the difficulty parameter Δ_d varies from 0.1 to 0.4. It can be seen that Probe-Rank and its variant always consume less samples than IIR to recover the true ranking. Note that in the WST setting, comparison probabilities are all identically distributed and thus on average, adjacent items are as hard as nonadjacent items to compare. When Δ_d is fixed, as n gets larger and larger, IIR will eventually outperform Probe-Rank. This is consistent with our theoretical results presented in Section 3.5. Moreover, as indicated by the experimental results, Probe-Rank-SE can further reduce the sample complexity compared with Probe-Rank.



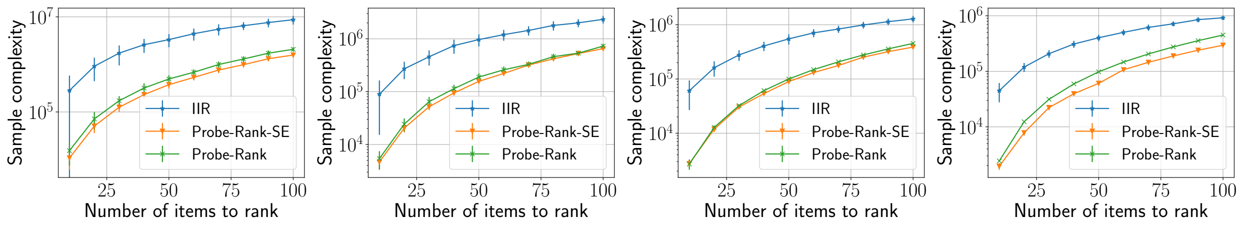
(a) WST: $\Delta_d = 0.1$ (b) WST: $\Delta_d = 0.2$ (c) WST: $\Delta_d = 0.3$ (d) WST: $\Delta_d = 0.4$

Figure 3.5: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the WST setting. In each subfigure, Δ_d is fixed while the number of items varies.



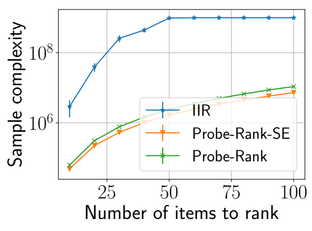
(a) SST: $\Delta_d = 0.1$ (b) SST: $\Delta_d = 0.2$ (c) SST: $\Delta_d = 0.3$ (d) SST: $\Delta_d = 0.4$

Figure 3.6: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the SST setting. In each subfigure, Δ_d is fixed while the number of items varies.

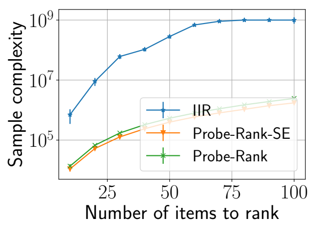


(a) NON-SST: $\Delta_d = 0.1$ (b) NON-SST: $\Delta_d = 0.2$ (c) NON-SST: $\Delta_d = 0.3$ (d) NON-SST: $\Delta_d = 0.4$

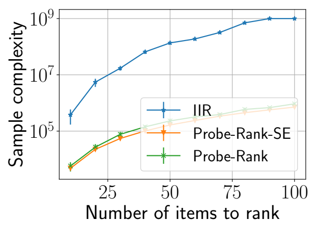
Figure 3.7: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the NON-SST setting. In each subfigure, Δ_d is fixed while the number of items varies.



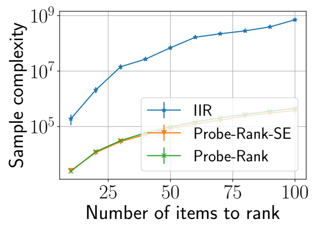
(a) $\Delta_d = 0.1$, $\alpha = 1$



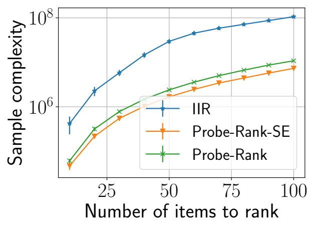
(b) $\Delta_d = 0.2$, $\alpha = 1$



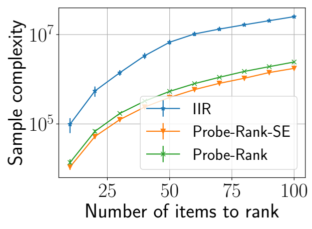
(c) $\Delta_d = 0.3$, $\alpha = 1$



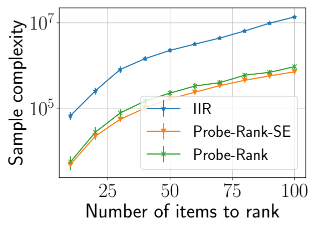
(d) $\Delta_d = 0.4$, $\alpha = 1$



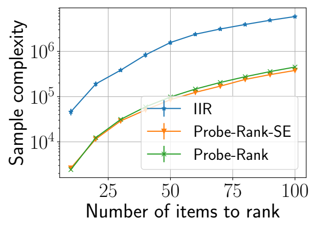
(e) $\Delta_d = 0.1$, $\alpha = 0.5$



(f) $\Delta_d = 0.2$, $\alpha = 0.5$

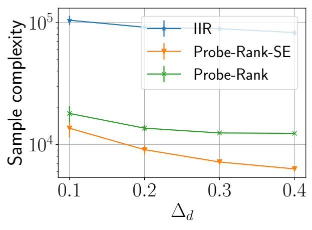


(g) $\Delta_d = 0.3$, $\alpha = 0.5$

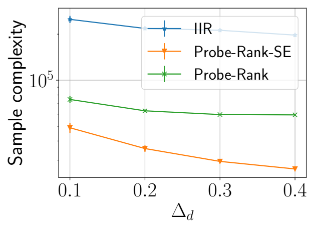


(h) $\Delta_d = 0.4$, $\alpha = 0.5$

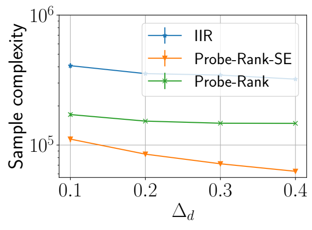
Figure 3.8: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the ADJ-ASYM setting. In each subfigure, Δ_d and α are fixed while the number of items varies.



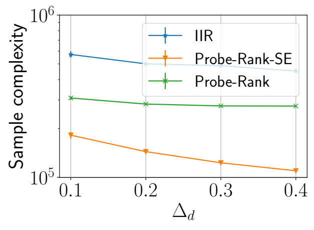
(a) WST, $n = 20$



(b) WST, $n = 40$

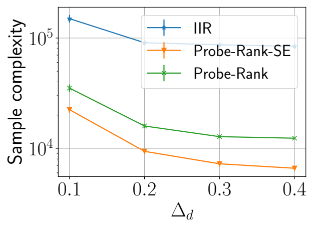


(c) WST, $n = 60$

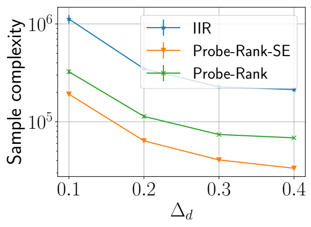


(d) WST, $n = 80$

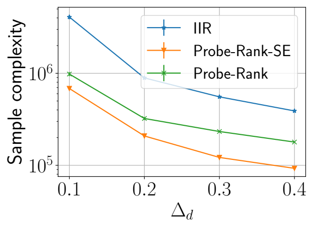
Figure 3.9: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the WST setting. In each subfigure, n is fixed while Δ_d varies.



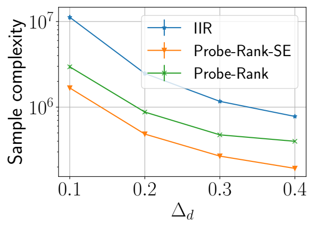
(a) SST, $n = 20$



(b) SST, $n = 40$

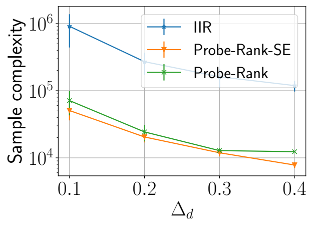


(c) SST, $n = 60$

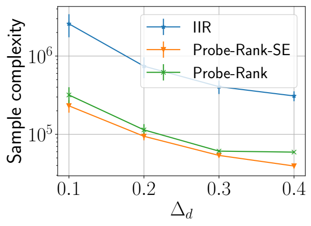


(d) SST, $n = 80$

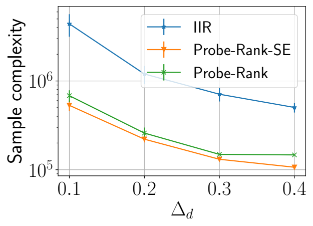
Figure 3.10: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the SST setting. In each subfigure, n is fixed while Δ_d varies.



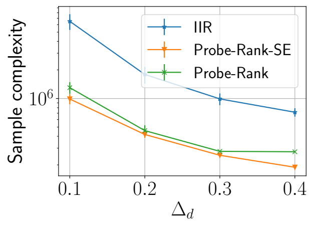
(a) NON-SST, $n = 20$



(b) NON-SST, $n = 40$

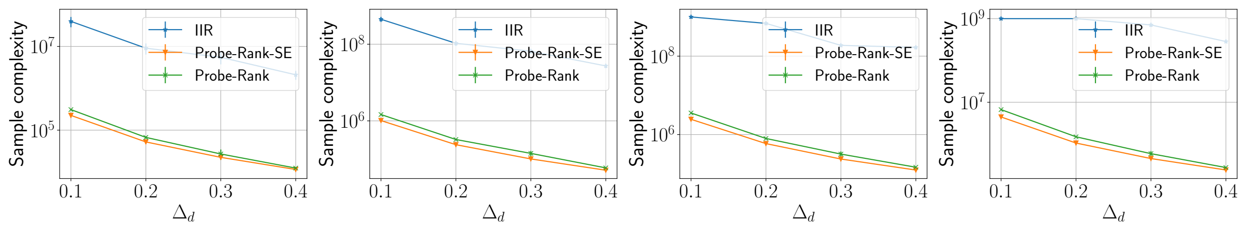


(c) NON-SST, $n = 60$

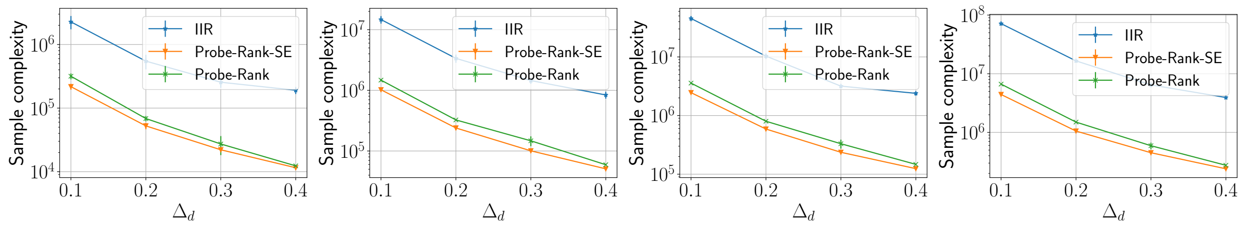


(d) NON-SST, $n = 80$

Figure 3.11: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the NON-SST setting. In each subfigure, n is fixed while Δ_d varies.



(a) $\alpha = 1$, $n = 20$ (b) $\alpha = 1$, $n = 40$ (c) $\alpha = 1$, $n = 60$ (d) $\alpha = 1$, $n = 80$



(e) $\alpha = 1/2$, $n = 20$ (f) $\alpha = 1/2$, $n = 40$ (g) $\alpha = 1/2$, $n = 60$ (h) $\alpha = 1/2$, $n = 80$

Figure 3.12: Comparison of Probe-Rank, Probe-Rank-SE and IIR under the ADJ-ASYM setting. In each subfigure, n and α are fixed while Δ_d varies.

3.12 Lower Bound Analysis

In this section, we present the reduction from the maxing problem for $\mathcal{I}_{SNG}$ (Problem 2) to the maxing problem for $\mathcal{I}_{WST}$ (Problem 1). The two problems are restated as follows.

We first consider another problem instance $\mathcal{I}_{SYM}$ and show that the maxing problem for $\mathcal{I}_{SYM}$ can be reduced to the maxing problem for $\mathcal{I}_{WST}$. The problem instance $\mathcal{I}_{SYM}$ is modified from $\mathcal{I}_{WST}$ by setting the values of $p_{i,j}$ to be exactly $\frac{1}{2}$.

Problem 3 ($\mathcal{I}_{SYM}$)**.** Consider n items with an underlying ordering '$\succ$'. The comparison probabilities are defined as:

$$p_{i,j} := \begin{cases} \frac{1}{2} + \Delta, \text{ if } i \succ j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2} - \Delta, \text{ if } i \prec j \text{ and } i, j \text{ are adjacent,} \\ \frac{1}{2}, \text{ otherwise.} \end{cases}$$

Note that $\mathcal{I}_{SYM}$ does not satisfy the WST condition as the comparison probabilities can be $\frac{1}{2}$. However, any δ-correct algorithm that finds the maximum item for $\mathcal{I}_{WST}$ efficiently can also find the maximum item for $\mathcal{I}_{SYM}$ efficiently, shown as follows.

Reduction from $\mathcal{I}_{SYM}$ to $\mathcal{I}_{WST}$ Let $\mathcal{A}$ be any δ-algorithm that finds the maximum item for any instance that satisfies the WST condition. Algorithm $\mathcal{A}$ is also able to find the maximum item for $\mathcal{I}_{WST}$ with any $c > 0$. Consider any interaction trajectory $\mathcal{T}$ defined by the sequence of comparisons (including the choices for item pairs and their outcomes) with length smaller than $Cn^2 \log(1/\delta)/\Delta^2$ for some constant C. Under the two instances $\mathcal{I}_{SYM}$ and $\mathcal{I}_{WST}$, the probabilities of occurrences of $\mathcal{T}$, denoted $\mathbb{P}_{SYM}$ and $\mathbb{P}_{WST}$, satisfy

$$\begin{aligned}
\frac{\mathbb{P}_{SYM}(\mathcal{T})}{\mathbb{P}_{WST}(\mathcal{T})} &\geqslant \left(\frac{1/2 - cn^{-10}\Delta^2/\log(1/\delta)}{1/2}\right)^{Cn^2\log(1/\delta)/\Delta^2} \\
&> \left(\frac{1}{1 + 4cn^{-10}\Delta^2/\log(1/\delta)}\right)^{Cn^2\log(1/\delta)/\Delta^2} \\
&\geqslant e^{-4cn^{-10}\Delta^2/\log(1/\delta)\cdot Cn^2\log(1/\delta)/\Delta^2} \\
&= e^{-4cCn^{-8}} \\
&\geqslant 1 - \delta, \qquad (3.12.1)
\end{aligned}$$

where the first inequality holds because the likelihood ratio for one query is upper bounded by the base and the number of queries on nonadjacent pairs is bounded by the exponent; the second inequality assumes $cn^{-10}\Delta^2/\log(1/\delta) < 1/4$, which holds for n sufficiently large; the third inequality is due to $(1 + x) \leqslant e^x$. The last inequality can hold by choosing a small enough c.

If $\mathcal{A}$ solves the maxing problem for $\mathcal{I}_{WST}$ with probability at least $1 - \delta$ and conducts at most $Cn^2\log(1/\delta)/\Delta^2$ comparisons, then by inequality (3.12.1),

$$\begin{aligned}\mathbb{P}_{SYM}(\mathcal{A}\text{ finds the correct maximum}) &= \sum_{\mathcal{T}\in\mathcal{E}_f}\mathbb{P}_{SYM}(\mathcal{T}) \geqslant (1-\delta)\sum_{\mathcal{T}\in\mathcal{E}_f}\mathbb{P}_{WST}(\mathcal{T})\\ &= (1-\delta)\,\mathbb{P}_{WST}(\mathcal{A}\text{ finds the correct maximum})\\ &\geqslant (1-\delta)^2\\ &> 1-2\delta, \end{aligned} \tag{3.12.2}$$

where $\mathcal{E}_f$ denotes the collection of trajectories where $\mathcal{A}$ returns the correct maximum. In other words, $\mathcal{A}$ is also a 2δ-correct algorithm that solves the maxing problem for $\mathcal{I}_{SYM}$ and conducts at most $Cn^2\log(1/\delta)/\Delta^2$ comparisons. Further, if we force $\mathcal{A}$ to terminate after $Cn^2\log(1/\delta)/\Delta^2$ comparisons have been made, then $\mathcal{A}$ is still correct with probability at least $1-\delta$ and with expected number of samples upper bounded by $Cn^2\log(1/\delta)/\Delta^2$.

The second step is to reduce the maxing problem for $\mathcal{I}_{SNG}$ to the maxing problem for $\mathcal{I}_{SYM}$.

Reduction from $\mathcal{I}_{SNG}$ to $\mathcal{I}_{SYM}$ Let $\mathcal{A}$ be any δ-correct algorithm that can find the maximum item for $\mathcal{I}_{SYM}$. Without loss of generality, we can assume that when comparing an item pair i,j, $i<j$, $\mathcal{A}$ takes in answer 1 representing i is more preferred and answer 0 representing j is more preferred. We construct a δ-correct algorithm $\mathcal{A}'$ that can find the maximum item for $\mathcal{I}_{SNG}$ from $\mathcal{A}$:

> Given $\mathcal{A}$, whenever $\mathcal{A}$ compares item pair (i,j),
> with probability 1/2, $\mathcal{A}'$ queries 'if $i \succ j$', gets the sample Y and feeds Y to $\mathcal{A}$;
> with probability 1/2, $\mathcal{A}'$ queries 'if $j \succ i$', gets the sample Y and feeds $1-Y$ to $\mathcal{A}$. Whenever $\mathcal{A}$ terminates and return an item, $\mathcal{A}'$ also terminates and return the same item.

It is clear that, if $\mathcal{A}$ queries an adjacent pair i,j with $i \succ j$, the feedback Y is an average over $Y_{i,j}$ (Ber($\frac{1}{2}$)) and $1-Y_{j,i}$ (1-Ber($\frac{1}{2}-2\Delta$)), which is Ber($\frac{1}{2}+\Delta$); if i,j are adjacent and

$i \prec j$, the feedback Y is an average of $\mathrm{Ber}(\frac{1}{2} - 2\Delta)$ and $1-\mathrm{Ber}(\frac{1}{2})$, which is $\mathrm{Ber}(\frac{1}{2} - \Delta)$; if i, j are nonadjacent, the feedback Y is an average of two $\mathrm{Ber}(\frac{1}{2})$ random variables, which is still $\mathrm{Ber}(\frac{1}{2})$. Therefore, $\mathcal{A}$ gets the same feedback when it is performed over $\mathcal{I}_{SYM}$. If $\mathcal{A}$ is a δ-correct maxing algorithm for $\mathcal{I}_{SYM}$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ comparisons on average, then $\mathcal{A}'$ is a δ-correct maxing algorithm for $\mathcal{I}_{SNG}$ with the same sample complexity.

To summarize, if there exists a δ-correct algorithm $\mathcal{A}$ that solves the maxing problem for $\mathcal{I}_{WST}$ and conducts at most $Cn^2 \log(1/\delta)/\Delta^2$ on average, then with the reduction, we can conclude there exists a 2δ-correct algorithm $\mathcal{A}'$ that solves Problem 2 with the same sample complexity. Since the above argument holds for any $C > 0$ and in Section 3.7, we argued that Example 2 requires $\Omega(n^2 \log(1/\delta)/\Delta^2)$ queries, we thus conjecture that any δ-correct algorithm $\mathcal{A}$ that solves the maxing (and thus ranking) problem for $\mathcal{I}_{WST}$ conducts $\Omega(n^2 \log(1/\delta)/\Delta^2)$ comparisons.

CHAPTER 4

Borda Regret Minimization for Generalized Linear Dueling Bandits

4.1 Introduction

Multi-armed bandits (MAB) (Lattimore and Szepesvári, 2020) is an interactive game where in each round, an agent chooses an arm to pull and receives a noisy reward as feedback. In contrast to numerical feedback considered in classic MAB settings, preferential feedback is more natural in various online learning tasks including information retrieval Yue and Joachims (2009), recommendation systems Sui and Burdick (2014), ranking Minka et al. (2018), crowdsourcing Chen et al. (2013), etc. Moreover, numerical feedback is also more difficult to gauge and prone to errors in many real-world applications. For example, when provided with items to shop or movies to watch, it is more natural for a customer to pick a preferred one than scoring the options. This motivates *Dueling Bandits* (Yue and Joachims, 2009), where the agent repeatedly pulls two arms at a time and is provided with feedback being the binary outcome of "duels" between the two arms.

In dueling bandits problems, the outcome of duels is commonly modeled as Bernoulli random variables due to their binary nature. In each round, suppose the agent chooses to compare arm i and j, then the binary feedback is assumed to be sampled independently from a Bernoulli distribution. For a dueling bandits instance with K arms, the probabilistic model of the instance can be fully characterized by a $K \times K$ preference probability matrix with each entry being: $p_{i,j} = \mathbb{P}(\text{arm } i \text{ is chosen over arm } j)$.

In a broader range of applications such as ranking, “arms” are often referred to as “items”. We will use these two terms interchangeably in the rest of this chapter. One central goal of dueling bandits is to devise a strategy to identify the “optimal” item as quickly as possible, measured by either sample complexity or cumulative regret. However, the notion of optimality for dueling bandits is way harder to define than for multi-armed bandits. The latter can simply define the arm with the highest numerical feedback as the optimal arm, while for dueling bandits there is no obvious definition solely dependent on $\{p_{i,j} | i, j \in [K]\}$.

The first few works on dueling bandits imposed strong assumptions on $p_{i,j}$. For example, Yue et al. (2012) assumed that there exists a true ranking that is coherent among all items, and the preference probabilities must satisfy both strong stochastic transitivity (SST) and stochastic triangle inequality (STI). While relaxations like weak stochastic transitivity (Falahatgar et al., 2018) or relaxed stochastic transitivity (Yue and Joachims, 2011b) exist, they typically still assume the true ranking exists and the preference probabilities are consistent, i.e., $p_{i,j} > \frac{1}{2}$ if and only if i is ranked higher than j. In reality, the existence of such coherent ranking aligned with item preferences is rarely the case. For example, $p_{i,j}$ may be interpreted as the probability of one basketball team i beating another team j, and there can be a circle among the match advantage relations.

In this chapter, we do not assume such coherent ranking exists and solely rely on the *Borda score* based on preference probabilities. The Borda score $B(i)$ of an item i is the probability that it is preferred when compared with another random item, namely $B(i) := \frac{1}{K-1} \sum_{j \neq i} p_{i,j}$. The item with the highest Borda score is called the *Borda winner*. The Borda winner is intuitively appealing and always well-defined for any set of preferential probabilities. The Borda score also does not require the problem instance to obey any consistency or transitivity, and it is considered one of the most general criteria.

To identify the Borda winner, estimations of the Borda scores are needed. Since estimating the Borda score for one item requires comparing it with every other items, the sample complexity is prohibitively high when there are numerous items. On the other hand, in many real-world applications, the agent has access to side information that can assist the evaluation of $p_{i,j}$. For instance, an e-commerce item carries its category as well as many other attributes, and the user might have a preference for a certain category (Wang et al., 2018). For a movie, the genre and the plot as well as the directors and actors can also be taken into consideration when making choices (Liu et al., 2017).

Based on the above motivation, we consider *Generalized Linear Dueling Bandits.* In each round, the agent selects two items from a finite set of items and receives a comparison result of the preferred item. The comparisons depend on known intrinsic contexts/features associated with each pair of items. The contexts can be obtained from upstream tasks, such as topic modeling (Zhu et al., 2012) or embedding (Vasile et al., 2016). Our goal is to adaptively select items and minimize the regret with respect to the optimal item (i.e., Borda winner).

4.1.1 Organization of this Chapter

In this chapter, we will achieve the above-mentioned goal from both theoretical and empirical perspectives. We review the most relevant work in the literature to ours in Section 4.2. We present the backgrounds and preliminaries of the problem setting in Section 4.3 and discuss the challenges of extending from previous works. In Section 4.4, we show a hardness result regarding the Borda regret minimization for the (generalized) linear model. We prove a worst-case regret lower bound $\Omega(d^{2/3}T^{2/3})$ for our dueling bandit model, showing that even in the stochastic setting, minimizing the Borda regret is difficult. In Section 4.5, we propose an explore-then-commit type algorithm under the stochastic setting, which can achieve a nearly matching upper bound $\widetilde{O}(d^{2/3}T^{2/3})$. In Section 4.6, we propose an EXP3 type algorithm for linear dueling bandits under the adversarial setting, which can achieve a nearly matching upper bound $\widetilde{O}\big((d\log K)^{1/3}T^{2/3}\big)$. In Section 4.7, we conduct empirical studies to verify the correctness of our theoretical claims. Finally, we conclude the chapter with Section 4.8. We defer the detailed proof of the theorems to Section 4.9.

Additional Notations The weighted ℓ_2-norm associated with a positive-definite matrix $\mathbf{A}$ is defined as $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$. The minimum eigenvalue of a matrix $\mathbf{A}$ is written as $\lambda_{\min}(\mathbf{A})$. We use $\mathbf{A} \geq \mathbf{B}$ to denote that the matrix $\mathbf{A} - \mathbf{B}$ is positive semi-definite.

4.2 Related Work

Multi-armed and Contextual Bandits Multi-armed bandit is a problem of identifying the best choice in a sequential decision-making system. It has been studied in numerous ways with a wide range of applications (Even-Dar et al., 2002b; Lai et al., 1985; Kuleshov and Precup, 2014). Contextual linear bandit is a special type of contextual bandit problem where the agent is provided with side information, i.e., contexts, and rewards are assumed to have a linear structure. Various algorithms (Rusmevichientong and Tsitsiklis, 2010; Filippi et al., 2010; Abbasi-Yadkori et al., 2011; Li et al., 2017; Jun et al., 2017) have been proposed to utilize this contextual information.

Dueling Bandits and Its Performance Metrics Dueling bandits is a variant of MAB with preferential feedback (Yue et al., 2012; Zoghi et al., 2014a, 2015b). A comprehensive survey is provided by Bengs et al. (2021). As discussed previously, the probabilistic structure of a dueling bandits problem is governed by the preference probabilities, over which an optimal item needs to be defined. Optimality under the *Borda score* criteria has been adopted by several previous works (Jamieson et al., 2015; Falahatgar et al., 2017a; Heckel et al., 2018; Saha et al., 2021a). The most relevant work to ours is Saha et al. (2021a), where they studied the problem of regret minimization for adversarial dueling bandits and proved a T-round Borda regret upper bound $\widetilde{O}(K^{1/3}T^{2/3})$. They also provide an $\Omega(K^{1/3}T^{2/3})$ lower bound for stationary dueling bandits using Borda regret.

Apart from the Borda score, *Copeland score* is also a widely used criteria (Urvoy et al., 2013; Busa-Fekete et al., 2013, 2014; Zoghi et al., 2015b, 2014b; Wu and Liu, 2016; Komiyama et al., 2016; Brandt et al., 2022; Bengs et al., 2023). It is defined as $C(i) := \frac{1}{K-1}\sum_{j\neq i}\mathbb{1}\{p_{i,j} > 1/2\}$. A Copeland winner is the item that beats the most number of other items. It can be viewed as a "thresholded" version of Borda winner. In addition to Borda and Copeland winners, optimality notions such as a von Neumann winner were also studied by Ramamohan et al. (2016); Dudík et al. (2015); Balsubramani et al. (2016). Saha and Krishnamurthy (2021) considered a related regret definition based on the optimality gap in hindsight and proposed algorithms for the general function class.

Another line of work focuses on identifying the optimal item or the total ranking, assuming the preference probabilities are consistent. Common consistency conditions include Strong Stochastic Transitivity (Yue et al., 2012; Falahatgar et al., 2017a,b), Weak Stochastic Transitivity (Falahatgar et al., 2018; Ren et al., 2019; Wu et al., 2022; Lou et al., 2022), Relaxed Stochastic Transitivity (Yue and Joachims, 2011b) and Stochastic Triangle Inequality. Sometimes the aforementioned transitivity can also be implied by some structured models like the Bradley–Terry model. We emphasize that these consistency conditions are not assumed or implicitly implied in our setting.

Contextual Dueling Bandits Dudík et al. (2015) first incorporated contextual information into the dueling bandit framework. Later, Saha (2021) studied structured contextual dueling bandits where each item i has its own contextual vector $\mathbf{x}_i$ (sometimes called Linear Stochastic Transitivity). Each item then has an intrinsic score v_i equal to the linear product of an unknown parameter vector $\boldsymbol{\theta}^*$ and its contextual vector $\mathbf{x}_i$. The preference probability between two items i and j is assumed to be $\mu(v_i - v_j)$ where $\mu(\cdot)$ is the logistic function. These intrinsic scores of items naturally define a ranking over items. The regret is also computed as the gap between the scores of pulled items and the best item. Bengs et al. (2022) further extended the problem to general link functions and proved a stronger lower bound on the weak regret. Di et al. (2024) proposed a variance-aware algorithm for this setting with improved regret upper bound. Unlike the works mentioned above, we assume that the contextual vectors are associated with item pairs and define regret on the Borda score. In Section 4.3.3, we provide a more detailed discussion showing that the linear stochastic transitivity setting can be viewed as a special case of our model.

4.3 Backgrounds and Preliminaries

4.3.1 Problem Setting

We first consider the stochastic preferential feedback model with K items in the fixed time horizon setting. We denote the item set by $[K]$ and let T be the total number of rounds. In each round t, the agent can pick any pair of items (i_t, j_t) to compare and receive stochastic feedback about whether item i_t is preferred over item j_t, (denoted by $i_t \succ j_t$). We denote the probability of seeing the event $i \succ j$ as $p_{i,j} \in [0,1]$. Naturally, we assume $p_{i,j} + p_{j,i} = 1$, and $p_{i,i} = 1/2$.

In this chapter, we are concerned with the generalized linear model (GLM), where there is assumed to exist an *unknown* parameter $\boldsymbol{\theta}^* \in \mathbb{R}^d$, and each pair of items (i,j) has its own *known* contextual/feature vector $\boldsymbol{\phi}_{i,j} \in \mathbb{R}^d$ with $\|\boldsymbol{\phi}_{i,j}\| \leqslant 1$. There is also a fixed known link function (sometimes called comparison function) $\mu(\cdot)$ that is monotonically increasing and satisfies $\mu(x) + \mu(-x) = 1$, e.g. a linear function or the logistic function $\mu(x) = 1/(1+e^{-x})$. The preference probability is defined as $p_{i,j} = \mu(\boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*)$. In each round, denote $r_t = \mathbb{1}\{i_t \succ j_t\}$, then we have

$$\mathbb{E}[r_t | i_t, j_t] = p_{i_t,j_t} = \mu(\boldsymbol{\phi}_{i_t,j_t}^\top \boldsymbol{\theta}^*).$$

Then our model can also be written as

$$r_t = \mu(\boldsymbol{\phi}_{i_t,j_t}^\top \boldsymbol{\theta}^*) + \epsilon_t,$$

where the noises $\{\epsilon_t\}_{t \in [T]}$ are zero-mean, 1-sub-Gaussian and assumed independent from each other. Note that, given the constraint $p_{i,j} + p_{j,i} = 1$, it is implied that $\boldsymbol{\phi}_{i,j} = -\boldsymbol{\phi}_{j,i}$ for any $i \in [K], j \in [K]$.

The agent's goal is to maximize the cumulative Borda score. The (slightly modified [1]) Borda score of item i is defined as $B(i) = \frac{1}{K}\sum_{j=1}^{K} p_{i,j}$, and the Borda winner is defined as $i^* = \mathrm{a}rgmax_{i\in[K]}B(i)$. The problem of merely identifying the Borda winner was deemed trivial (Zoghi et al., 2014a; Busa-Fekete et al., 2018) because for a fixed item i, uniformly random sampling j and receiving feedback $r_{i,j} = \text{Bernoulli}(p_{i,j})$ yield a Bernoulli random variable with its expectation being the Borda score $B(i)$. This so-called *Borda reduction* trick makes identifying the Borda winner as easy as the best-arm identification for K-armed bandits. Moreover, if the regret is defined as $\text{Regret}(T) = \sum_{t=1}^{T}(B(i^*) - B(i_t))$, then any optimal algorithms for multi-arm bandits can achieve $\widetilde{O}(\sqrt{T})$ regret.

However, the above definition of regret does not respect the fact that a pair of items is selected in each round. When the agent chooses two items to compare, it is natural to define the regret so that both items contribute equally. A commonly used regret, e.g., in Saha et al. (2021a), has the following form:

$$\text{Regret}(T) = \sum_{t=1}^{T}\big(2B(i^*) - B(i_t) - B(j_t)\big), \tag{4.3.1}$$

where the regret is defined as the sum of the sub-optimality of both selected arms. Sub-optimality is measured by the gap between the Borda scores of the compared items and the Borda winner. This form of regret deems any classical multi-arm bandit algorithm with Borda reduction vacuous because taking j_t into consideration will invoke $\Theta(T)$ regret.

Adversarial Setting Saha et al. (2021b) considered an adversarial setting for the multi-armed case, where in each round t, the comparison follows a potentially different probability model, denoted by $\{p_{i,j}^t\}_{i,j\in[K]}$. In this chapter, we consider its contextual counterpart. Formally, we assume there is an underlying parameter $\boldsymbol{\theta}_t^*$, and in round t, the preference probability is defined as $p_{i,j}^t = \mu(\boldsymbol{\phi}_{i,j}^\top\boldsymbol{\theta}_t^*)$.

[1]Previous works define Borda score as $B_i' = \frac{1}{K-1}\sum_{j\neq i} p_{i,j}$, excluding the diagonal term $p_{i,i} = 1/2$. Our definition is equivalent since the difference between two items satisfies $B(i) - B_j = \frac{K-1}{K}(B_i' - B_j')$. Therefore, the regret will be in the same order for both definitions.

The Borda score of item $i \in [K]$ in round t is defined as $B_t(i) = \frac{1}{K}\sum_{j=1}^{K} p_{i,j}^t$, and the Borda winner in round T is defined as $i^* = argmax_{i\in[K]} \sum_{t=1}^{T} B_t(i)$. The T-round regret is thus defined as $\text{Regret}(T) = \sum_{t=1}^{T} \big(2B_t(i^*) - B_t(i_t) - B_t(j_t)\big)$.

4.3.2 Assumptions

In this section, we present the assumptions required for establishing theoretical guarantees. Due to the fact that the analysis technique is largely extracted from Li et al. (2017), we follow them to make assumptions to enable regret minimization for generalized linear dueling bandits.

We make a regularity assumption about the distribution of the contextual vectors:

Assumption 4.3.1. There exists a constant $\lambda_0 > 0$ such that

$$\lambda_{\min}\big(\frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} \boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top\big) \geqslant \lambda_0$$

.

This assumption is only utilized to initialize the design matrix $\mathbf{V}_\tau = \sum_{t=1}^{\tau} \boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top$ so that the minimum eigenvalue is large enough. We follow Li et al. (2017) to deem λ_0 as a constant.

We also need the following assumption regarding the link function $\mu(\cdot)$:

Assumption 4.3.2. Let $\dot{\mu}$ be the first-order derivative of μ. We have

$$\kappa := \inf_{\|\mathbf{x}\|\leqslant 1, \|\boldsymbol{\theta}-\boldsymbol{\theta}^*\|\leqslant 1} \dot{\mu}(\mathbf{x}^\top\boldsymbol{\theta}) > 0.$$

Assuming $\kappa > 0$ is necessary to ensure the maximum log-likelihood estimator can converge to the true parameter $\boldsymbol{\theta}^*$ (Li et al., 2017, Section 3). This type of assumption is commonly made in previous works for generalized linear models (Filippi et al., 2010; Li et al., 2017; Faury et al., 2020).

Another common assumption is regarding the continuity and smoothness of the link function.

Assumption 4.3.3. μ is twice differentiable. Its first and second-order derivatives are upper-bounded by constants L_μ and M_μ respectively.

This is a very mild assumption. For example, it is easy to verify that the logistic link function satisfies Theorem 4.3.3 with $L_\mu = M_\mu = 1/4$.

4.3.3 Existing Results for Structured Contexts

A structural assumption made by some previous works (Saha, 2021) is that $\boldsymbol{\phi}_{i,j} = \mathbf{x}_i - \mathbf{x}_j$, where $\mathbf{x}_i$ can be seen as some feature vectors tied to the item. In this work, we do not consider minimizing the Borda regret under the structural assumption.

The immediate reason is that, when $p_{i,j} = \mu(\mathbf{x}_i^\top \boldsymbol{\theta}^* - \mathbf{x}_j^\top \boldsymbol{\theta}^*)$, with $\mu(\cdot)$ being the logistic function, the probability model $p_{i,j}$ effectively becomes (a linear version of) the well-known Bradley-Terry model. Namely, each item is tied to a value $v_i = \mathbf{x}_i^\top \boldsymbol{\theta}^*$, and the comparison probability follows $p_{i,j} = \frac{e^{v_i}}{e^{v_i}+e^{v_j}}$. More importantly, this kind of model satisfies both the strong stochastic transitivity (SST) and the stochastic triangle inequality (STI), which are unlikely to satisfy in reality.

Furthermore, when stochastic transitivity holds, there is a true ranking among the items, determined by $\mathbf{x}_i^\top \boldsymbol{\theta}^*$. A true ranking renders concepts like the Borda winner or Copeland winner redundant because the rank-one item will always be the winner in every sense. When $\boldsymbol{\phi}_{i,j} = \mathbf{x}_i - \mathbf{x}_j$, Saha (2021) proposed algorithms that can achieve nearly optimal regret $\widetilde{O}(d\sqrt{T})$, with regret being defined as

$$\text{Regret}(T) = \sum_{t=1}^{T} 2\langle \mathbf{x}_{i^*}, \boldsymbol{\theta}^* \rangle - \langle \mathbf{x}_{i_t}, \boldsymbol{\theta}^* \rangle - \langle \mathbf{x}_{j_t}, \boldsymbol{\theta}^* \rangle, \tag{4.3.2}$$

where $i^* = argmax_i \langle \mathbf{x}_i, \boldsymbol{\theta}^* \rangle$, which also happens to be the Borda winner[2]. Meanwhile, by

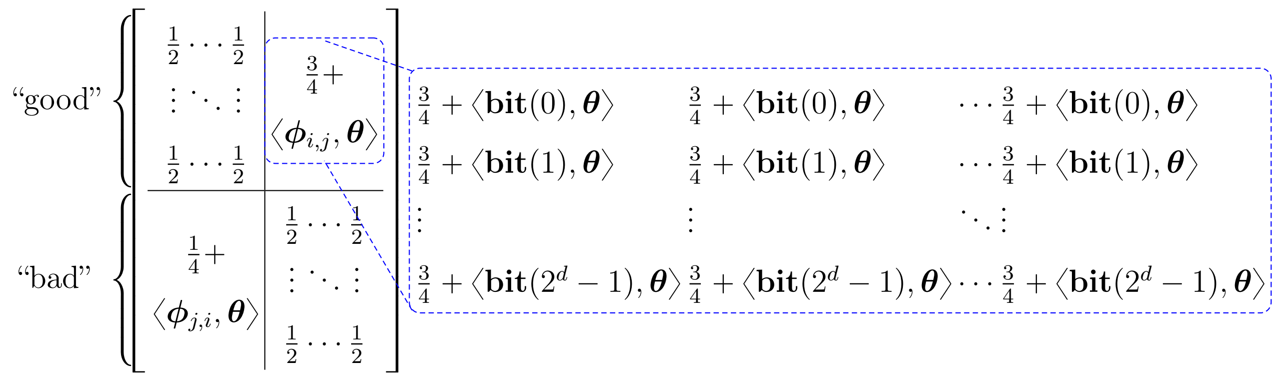
[2]Saha (2021) and the following works consider the case of time-varying features and consequently the

Theorem 4.3.3,

$$B(i^*) - B(j) = \frac{1}{K}\sum_{k=1}^{K}\big[\mu(\langle \mathbf{x}_{i^*} - \mathbf{x}_k, \boldsymbol{\theta}^*\rangle) - \mu(\langle \mathbf{x}_j - \mathbf{x}_k, \boldsymbol{\theta}^*\rangle)\big]$$
$$\leqslant L_\mu \cdot \langle \mathbf{x}_{i^*} - \mathbf{x}_j, \boldsymbol{\theta}^*\rangle,$$

where L_μ is the upper bound on the derivative of $\mu(\cdot)$. For logistic function $L_\mu = 1/4$. The Borda regret (4.3.1) is thus at most a constant multiple of (4.3.2). This shows Borda regret minimization can be sufficiently solved by Saha (2021) when structured contexts are present. We consider the most general case where the only restriction is the implicit assumption that $\boldsymbol{\phi}_{i,j} = -\boldsymbol{\phi}_{j,i}$.

4.4 The Hardness Result



"good"
"bad"
$\frac{1}{2} \cdots \frac{1}{2}$
$\frac{3}{4} + \langle \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta}\rangle$
$\frac{1}{4} + \langle \boldsymbol{\phi}_{j,i}, \boldsymbol{\theta}\rangle$
$\frac{3}{4} + \langle \mathbf{bit}(0), \boldsymbol{\theta}\rangle$
$\frac{3}{4} + \langle \mathbf{bit}(1), \boldsymbol{\theta}\rangle$
$\frac{3}{4} + \langle \mathbf{bit}(2^d - 1), \boldsymbol{\theta}\rangle$

Figure 4.1: Illustration of the hard-to-learn preference probability matrix $\{p^{\boldsymbol{\theta}}_{i,j}\}_{i\in[K],j\in[K]}$. There are $K = 2^{d+1}$ items in total. The first 2^d items are "good" items with higher Borda scores, and the last 2^d items are "bad" items. The upper right block $\{p_{i,j}\}_{i<2^d, j\geqslant 2^d}$ is defined as shown in the blue bubble. The lower left block satisfies $p_{i,j} = 1 - p_{j,i}$. For any $\boldsymbol{\theta}$, there exist one and only best item i such that $\mathbf{bit}(i) = \mathbf{sign}(\boldsymbol{\theta})$.

best arm changes over time. Here we restrict to fixed arms for better illustration.

This section presents Theorem 4.4.1, a worst-case regret lower bound for the stochastic linear dueling bandits. The proof of Theorem 4.4.1 relies on a class of hard instances, as shown in Figure 4.1. We show that any algorithm will incur a certain amount of regret when applied to this hard instance class. The constructed hard instances follow a stochastic linear model, which is a sub-class of the generalized linear model. Saha et al. (2021b) first proposed a similar construction for finite many arms with no contexts. Their design contains $K-1$ identical arms against one best arm. This design will not work in our setting as it leads to lower bounds sub-optimal in dimension d. The new construction of our lower bound is based on the hardness of identifying the best arm in the d-dimensional linear bandit model and the proof of the lower bound takes a rather different route.

For any $d > 0$, we construct the class of hard instances as follows. An instance is specified by a vector $\boldsymbol{\theta} \in \{-\Delta, +\Delta\}^d$. The instance contains 2^{d+1} items (indexed from 0 to $2^{d+1}-1$). The preference probability for an instance is defined by $p_{i,j}^{\boldsymbol{\theta}}$ as:

$$p_{i,j}^{\boldsymbol{\theta}} = \begin{cases} \frac{1}{2}, \text{ if } i < 2^d, j < 2^d \\ \frac{1}{2}, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ \frac{3}{4}, \text{ if } i < 2^d, j \geqslant 2^d \\ \frac{1}{4}, \text{ if } i \geqslant 2^d, j < 2^d \end{cases} + \langle \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta} \rangle,$$

and the d-dimensional feature vectors $\boldsymbol{\phi}_{i,j}$ are given by

$$\boldsymbol{\phi}_{i,j} = \begin{cases} \mathbf{0}, \text{ if } i < 2^d, j < 2^d \text{ or if } i \geqslant 2^d, j \geqslant 2^d \\ \mathbf{bit}(i), \text{ if } i < 2^d, j \geqslant 2^d \\ -\mathbf{bit}(j), \text{ if } i \geqslant 2^d, j < 2^d, \end{cases}$$

where $\mathbf{bit}(\cdot)$ is the (shifted) bit representation of non-negative integers, i.e., suppose x has the binary representation $x = b_0 \times 2^0 + b_1 \times 2^1 + \cdots + b_{d-1} \times 2^{d-1}$, then

$$\mathbf{bit}(x) = (2b_0 - 1, 2b_1 - 1, \ldots, 2b_{d-1} - 1) = 2\boldsymbol{b} - 1.$$

Note that $\mathbf{bit}(\cdot) \in \{-1, +1\}^d$, and that $\boldsymbol{\phi}_{i,j} = -\boldsymbol{\phi}_{j,i}$ is satisfied. The definition of $p^{\boldsymbol{\theta}}_{i,j}$ can be slightly tweaked to fit exactly the model described in Section 4.3 (see Remark 4.9.1 in Section 4.9.1).

Some calculation shows that the Borda scores of the 2^{d+1} items are:

$$B^{\boldsymbol{\theta}}(i) = \begin{cases} \frac{5}{8} + \frac{1}{2}\langle \mathbf{bit}(i), \boldsymbol{\theta} \rangle, \text{ if } i < 2^d, \\ \frac{3}{8}, \text{ if } i \geqslant 2^d. \end{cases}$$

Intuitively, the former half of items (those indexed from 0 to $2^d - 1$) are "good" items (one among them is optimal, others are nearly optimal), while the latter half of items are "bad" items. Under such hard instances, every time one of the two pulled items is a "bad" item, then a one-step regret $B^{\boldsymbol{\theta}}(i^*) - B^{\boldsymbol{\theta}}(i) \geqslant 1/4$ is incurred. To minimize regret, we should thus try to avoid pulling "bad" items. However, in order to identify the best item among all "good" items, comparisons between "good" and "bad" items are necessary. The reason is simply that comparisons between "good" items give no information about the Borda scores as the comparison probabilities are $p^{\boldsymbol{\theta}}_{i,j} = \frac{1}{2}$ for all $i, j < 2^d$. Hence, any algorithm that can decently distinguish among the "good" items has to pull "bad" ones for a fair amount of times, and large regret is thus incurred. A similar observation is also made by Saha et al. (2021a).

This specific construction emphasizes the intrinsic hardness of Borda regret minimization: to differentiate the best item from its close competitors, the algorithm must query the bad items to gain information.

Formally, this class of hard instances leads to the following regret lower bound for both stochastic and adversarial settings:

Theorem 4.4.1. For any algorithm $\mathcal{A}$, there exists a hard instance $\{p^{\boldsymbol{\theta}}_{i,j}\}$ with $T > 4d^2$, such that $\mathcal{A}$ will incur expected regret at least $\Omega(d^{2/3}T^{2/3})$.

The construction of this hard instance for linear dueling bandits is inspired by the worst-case lower bound for the stochastic linear bandit (Dani et al., 2008), which has the order

$\Omega(d\sqrt{T})$, while ours is $\Omega(d^{2/3}T^{2/3})$. The difference is that for the linear or multi-armed stochastic bandit, eliminating bad arms can make further exploration less expensive. But in our case, any amount of exploration will not reduce the cost of further exploration. This essentially means that exploration and exploitation must be separate, which is also supported by the fact that a simple explore-then-commit algorithm shown in Section 4.5 can be nearly optimal.

To prove the lower bound, we first apply a new reduction step to restrict the choice of i_t. Then we bound from below the regret by the expected number of sub-optimal arm pulls. The proof in Saha et al. (2021b) is directly based on hypothesis testing: either identifying the best arm with gap ϵ within T rounds (if $T > \frac{K}{1440\epsilon^3}$) or incurring ϵT regret (if $T \leqslant \frac{K}{1440\epsilon^3}$). In contrast, our proof technique bounds from below the regret by the expected number of sub-optimal arm pulls and does not divide the problem instances into two cases (i.e. whether $T \leqslant \frac{K}{1440\epsilon^3}$).

4.5 Stochastic Contextual Dueling Bandit

4.5.1 Algorithm Description

We propose an algorithm named Borda Explore-Then-Commit for Generalized Linear Models (BETC-GLM), presented in Algorithm 16. Our algorithm is inspired by the algorithm for generalized linear models proposed by Li et al. (2017).

At the high level, Algorithm 16 can be divided into two phases: the exploration phase (Line 2-11) and the exploitation phase (Line 12-14). The exploration phase ensures that the MLE estimator $\widehat{\boldsymbol{\theta}}$ is accurate enough so that the estimated Borda score is within $\widetilde{O}(\epsilon)$-range of the true Borda score (ignoring other quantities). Then the exploitation phase simply chooses the empirical Borda winner to incur small regret.

During the exploration phase, the algorithm first performs "pure exploration" (Line 2-5),

Algorithm 16 BETC-GLM

```
1: **Input:** time horizon $T$, number of items $K$, feature dimension $d$, feature vectors $\boldsymbol{\phi}_{i,j}$ for $i \in [K]$, $j \in [K]$, exploration rounds $\tau$, error tolerance $\epsilon$, failure probability $\delta$.
2: **for** $t = 1, 2, \dots, \tau$ **do**
3:     sample $i_t \sim \text{Uniform}([K])$, $j_t \sim \text{Uniform}([K])$
4:     query pair $(i_t, j_t)$ and receive feedback $r_t$
5: **end for**
6: Find the G-optimal design $\pi(i, j)$ based on $\boldsymbol{\phi}_{i,j}$ for $i \in [K]$, $j \in [K]$
7: Let $N(i, j) = \left\lceil \frac{d\pi(i,j)}{\epsilon^2} \right\rceil$ for any $(i, j) \in \text{supp}(\pi)$ , denote $N = \sum_{i=1}^{K} \sum_{j=1}^{K} N(i, j)$
8: **for** $i \in [K]$, $j \in [K]$, $s \in [N(i, j)]$ **do**
9:     set $t \leftarrow t + 1$, set $(i_t, j_t) = (i, j)$
10:    query pair $(i_t, j_t)$ and receive feedback $r_t$
11: **end for**
12: Calculate the empirical MLE estimator $\widehat{\boldsymbol{\theta}}_{\tau+N}$ based on all $\tau + N$ samples via (4.5.1)
13: Estimate the Borda score for each item:
    $$\widehat{B}(i) = \frac{1}{K} \sum_{j=1}^{K} \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}}_{\tau+N}), \qquad \widehat{i} = argmax_{i \in [K]} \widehat{B}(i)$$
14: Keep querying $(\widehat{i}, \widehat{i})$ for the rest of the time.
```

which can be seen as an initialization step for the algorithm. The purpose of this step is to ensure the design matrix $\mathbf{V}_{\tau+N} = \sum_{t=1}^{\tau+N} \boldsymbol{\phi}_{i_t,j_t} \boldsymbol{\phi}_{i_t,j_t}^\top$ is positive definite.

After that, the algorithm will perform the "designed exploration". Line 6 will find the G-optimal design, which minimizes the objective function $g(\pi) = \max_{i,j} \|\boldsymbol{\phi}_{i,j}\|_{\mathbf{V}(\pi)^{-1}}^2$, where $\mathbf{V}(\pi) := \sum_{i,j} \pi(i, j) \boldsymbol{\phi}_{i,j} \boldsymbol{\phi}_{i,j}^\top$. The G-optimal design $\pi^*(\cdot)$ satisfies $\|\boldsymbol{\phi}_{i,j}\|_{\mathbf{V}(\pi^*)^{-1}}^2 \leqslant d$, and can be efficiently approximated by the Frank-Wolfe algorithm (See Theorem 4.5.4 for a detailed discussion). Then the algorithm will follow $\pi(\cdot)$ found at Line 6 to determine how many samples (Line 7) are needed. At Line 8-11, there are in total $N = \sum_{i=1}^{K} \sum_{j=1}^{K} N(i, j)$ samples queried, and the algorithm shall index them by $t = \tau + 1, \tau + 2, \dots, \tau + N$.

At Line 12, the algorithm collects all the $\tau + N$ samples and performs the maximum likelihood estimation (MLE). For the generalized linear model, the MLE estimator $\widehat{\boldsymbol{\theta}}_{\tau+N}$ satisfies:

$$\sum_{t=1}^{\tau+N} \mu(\boldsymbol{\phi}_{i_t,j_t}^\top \widehat{\boldsymbol{\theta}}_{\tau+N})\boldsymbol{\phi}_{i_t,j_t} = \sum_{t=1}^{\tau+N} r_t \boldsymbol{\phi}_{i_t,j_t}, \tag{4.5.1}$$

or equivalently, it can be determined by solving a strongly concave optimization problem:

$$\widehat{\boldsymbol{\theta}}_{\tau+N} \in \mathrm{a}rgmax_{\boldsymbol{\theta}} \sum_{t=1}^{\tau+N} \Big(r_t \boldsymbol{\phi}_{i_t,j_t}^\top \boldsymbol{\theta} - m(\boldsymbol{\phi}_{i_t,j_t}^\top \boldsymbol{\theta}) \Big),$$

where $\dot{m}(\cdot) = \mu(\cdot)$. For the logistic link function, $m(x) = \log(1 + e^x)$. As a special case of our generalized linear model, the linear model has a closed-form solution for (4.5.1). For example, if $\mu(x) = \frac{1}{2} + x$, i.e. $p_{i,j} = \frac{1}{2} + \boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*$, then (4.5.1) becomes:

$$\widehat{\boldsymbol{\theta}}_{\tau+N} = \mathbf{V}_{\tau+N}^{-1} \sum_{t=1}^{\tau+N} (r_t - 1/2)\boldsymbol{\phi}_{i_t,j_t},$$

where $\mathbf{V}_{\tau+N} = \sum_{t=1}^{\tau+N} \boldsymbol{\phi}_{i_t,j_t} \boldsymbol{\phi}_{i_t,j_t}^\top$.

After the MLE estimator is obtained, Line 13 will calculate the estimated Borda score $\widehat{B}(i)$ for each item based on $\widehat{\boldsymbol{\theta}}_{\tau+N}$, and pick the empirically best one.

4.5.2 A Matching Regret Upper Bound

Algorithm 16 can be configured to tightly match the worst-case lower bound. The configuration and performance are described as follows:

Theorem 4.5.1. Suppose Assumption 4.3.1-4.3.3 hold and $T = \Omega(d^2)$. For any $\delta > 0$, if we set $\tau = C_4 \lambda_0^{-2}(d + \log(1/\delta))$ (C_4 is a universal constant) and $\epsilon = d^{1/6} T^{-1/3}$, then with probability at least $1 - 2\delta$, Algorithm 16 will incur regret bounded by:

$$O\Big(\kappa^{-1} d^{2/3} T^{2/3} \sqrt{\log\big(T/d\delta\big)}\Big).$$

By setting $\delta = T^{-1}$, the expected regret is bounded as $\widetilde{O}(\kappa^{-1} d^{2/3} T^{2/3})$.

For linear bandit models, such as the hard-to-learn instances in Section 4.4, κ is a universal constant. Therefore, Theorem 4.5.1 tightly matches the lower bound in Theorem 4.4.1, up to logarithmic factors. The detailed proof can be found in Section 4.9.2.1.

Remark 4.5.2 (Regret for Fewer Arms)**.** In typical scenarios, the number of items K is not exponentially large in the dimension d. In this case, we can choose a different parameter set of τ and ϵ such that Algorithm 16 can achieve a smaller regret bound $\widetilde{O}\big(\kappa^{-1}(d\log K)^{1/3}T^{2/3}\big)$ with smaller dependence on the dimension d. See Theorem 4.5.5 in Section 4.5.3.

Remark 4.5.3 (Regret for Infinitely Many Arms)**.** In most practical scenarios of dueling bandits, it is adequate to consider a finite number K of items (e.g., ranking items). Nonetheless, BETC-GLM can be easily adapted to accommodate infinitely many arms in terms of regret. We can construct a covering over all $\boldsymbol{\phi}_{i,j}$ and perform optimal design and exploration on the covering set. The resulting regret will be the same as our upper bound, i.e., $\widetilde{O}(d^{2/3}T^{2/3})$ up to some error caused by the epsilon net argument.

Remark 4.5.4 (Approximate G-optimal Design)**.** Algorithm 16 assumes an exact G-optimal design π is obtained. In the experiments, we use the Frank-Wolfe algorithm to solve the constraint optimization problem (See Algorithm 20, Section 4.11.4). To find a policy π such that $g(\pi) \leqslant (1+\varepsilon)g(\pi^*)$, roughly $O(d/\varepsilon)$ optimization steps are needed. Such a near-optimal design will introduce a factor of $(1+\varepsilon)^{1/3}$ into the upper bounds.

4.5.3 Regret Bound for Fewer Arms

In typical scenarios, the number of items K is not exponentially large in the dimension d. If this is the case, then we can choose a different parameter set of τ and ϵ such that Algorithm 16 can achieve a regret bound depending on $\log K$, and reduce the dependence on d. The performance can be characterized by the following theorem:

Theorem 4.5.5. For any $\delta > 0$, suppose the number of total rounds T satisfies,

$$T \geqslant \frac{C_3}{\kappa^6 \lambda_0^{3/2}} \max\Big\{d^{5/2}, \frac{\log(K^2/\delta)}{\sqrt{d}}\Big\}, \tag{4.5.2}$$

where C_3 is some large enough universal constant. Then if we set $\tau = (d\log(K/\delta))^{1/3}T^{2/3}$ and $\epsilon = d^{1/3}T^{-1/3}\log(3K^2/\delta)^{-1/6}$, Algorithm 16 will incur regret bounded by:

$$O\big(\kappa^{-1}(d\log(K/\delta))^{1/3}T^{2/3}\big).$$

By setting $\delta = T^{-1}$, the expected regret is bounded as $\widetilde{O}\big(\kappa^{-1}(d\log K)^{1/3}T^{2/3}\big)$.

The detailed proof can be found in Section 4.9.2.2.

4.6 Adversarial Contextual Dueling Bandit

This section addresses Borda regret minimization under the adversarial setting. As we introduced in Section 4.3.1, the unknown parameter $\boldsymbol{\theta}_t$ can vary for each round t, while the contextual vectors $\boldsymbol{\phi}_{i,j}$ are fixed.

Our proposed algorithm, BEXP3, is designed for the contextual linear model. Formally, in round t and given pair (i,j), we have $p_{i,j}^t = \frac{1}{2} + \langle \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta}_t^* \rangle$.

4.6.1 Algorithm Description

Algorithm 17 is adapted from the DEXP3 algorithm in Saha et al. (2021b), which deals with the adversarial multi-armed dueling bandit. Algorithm 17 maintains a distribution $q_t(\cdot)$ over $[K]$, initialized as uniform distribution (Line 2). In every round t, two items are chosen following q_t independently. Then Line 6 calculates the one-sample unbiased estimate $\widehat{\boldsymbol{\theta}}_t$ of the true underlying parameter $\boldsymbol{\theta}_t^*$. Line 7 further calculates the unbiased estimate of the (shifted) Borda score. Note that the true Borda score in round t satisfies $B_t(i) = \frac{1}{2} + \langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta}_t^* \rangle$. $\widehat{B}_t$ instead only estimates the second term of the Borda score. This is a choice to simplify the proof. The cumulative estimated score $\sum_{l=1}^t \widehat{B}_l(i)$ can

Algorithm 17 BEXP3

```
1: Input: time horizon $T$, number of items $K$, feature dimension $d$, feature vectors $\boldsymbol{\phi}_{i,j}$ for
   $i \in [K]$, $j \in [K]$, learning rate $\eta$, exploration parameter $\gamma$.
2: Initialize: $q_1(i) = \frac{1}{K}$.
3: for $t = 1, \ldots, T$ do
4:     Sample items $i_t \sim q_t$, $j_t \sim q_t$.
5:     Query pair $(i_t, j_t)$ and receive feedback $r_t$
6:     Calculate $Q_t = \sum_{i\in[K]} \sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$, $\widehat{\boldsymbol{\theta}}_t = Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}r_t$.
7:     Calculate the (shifted) Borda score estimates $\widehat{B}_t(i) = \langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, \widehat{\boldsymbol{\theta}}_t \rangle$.
8:     Update for all $i \in [K]$, set
       $$\widetilde{q}_{t+1}(i) = \frac{\exp(\eta \sum_{l=1}^t \widehat{B}_l(i))}{\sum_{j\in[K]} \exp(\eta \sum_{l=1}^t \widehat{B}_l(j))}; \qquad q_{t+1}(i) = (1-\gamma)\widetilde{q}_{t+1}(i) + \frac{\gamma}{K}.$$
9: end for
```

be seen as the estimated cumulative reward of item i in round t. In Line 8, q_{t+1} is defined by the classic exponential weight update, along with a uniform exploration policy controlled by γ.

4.6.2 Upper Bounds

Algorithm 17 can also be configured to tightly match the worst-case lower bound:

Theorem 4.6.1. Suppose Theorem 4.3.1 holds. If we set $\eta = (\log K)^{2/3}d^{-1/3}T^{-2/3}$ and $\gamma = \sqrt{\eta d/\lambda_0} = (\log K)^{1/3}d^{1/3}T^{-1/3}\lambda_0^{-1/2}$, then the expected regret is upper-bounded by

$$O\big((d\log K)^{1/3}T^{2/3}\big).$$

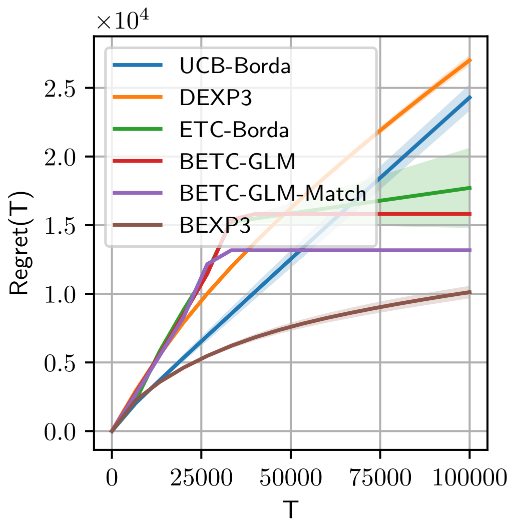
Note that the lower bound construction in Theorem 4.4.1 is for the linear model and has $K = O(2^d)$, thus exactly matching the upper bound. Meanwhile, it is viable to slightly modify Algorithm 17 to improve the regret to $\widetilde{O}(d^{2/3}T^{2/3})$ for very large K. The high-level idea is to use an ϵ-cover argument. In the d-dimensional space, it suffices to choose $O((1/\epsilon)^d)$ representative vectors to cover all the K average contextual vectors $\frac{1}{K}\sum_{j=1}^{K}\boldsymbol{\phi}_{i,j}$. The detailed reasoning and algorithm design can be found in Section 4.12.

4.7 Experiments

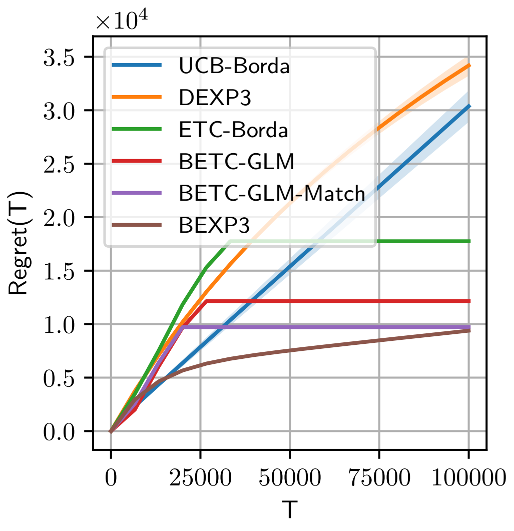
This section compares the proposed algorithm BETC-GLM with existing ones that are capable of minimizing Borda regret. We use random responses (generated from fixed preferential matrices) to interact with all tested algorithms. Each algorithm is run for 50 times over a time horizon of $T = 10^6$. We report both the mean and the standard deviation of the cumulative Borda regret and supply some analysis. The following list summarizes all methods we study in this section, a more complete description of the methods and parameters is available in Section 4.10: BETC-GLM(-Match): Algorithm 16 proposed in this chapter with different parameters. UCB-Borda: The UCB algorithm (Auer et al., 2002) using *Borda reduction*. DEXP3: Dueling-Exp3 developed by Saha et al. (2021a). ETC-Borda: A simple explore-then-commit algorithm that does not take any contextual information into account. BEXP3: The proposed method for adversarial Borda bandits displayed in Algorithm 17.

Generated Hard Case We first test the algorithms on the hard instances constructed in Section 4.4. We generate $\boldsymbol{\theta}^*$ randomly from $\{-\Delta, +\Delta\}^d$ with $\Delta = \frac{1}{4d}$ so that the comparison probabilities $p_{i,j}^{\boldsymbol{\theta}^*} \in [0,1]$ for all $i, j \in [K]$. We pick the dimension $d = 6$ and the number of arms is therefore $K = 2^{d+1} = 128$. Note the dual usage of d in our construction and the model setup in Section 4.3.1. We refer readers to Theorem 4.9.1 in Section 4.9.1 for more details.

As depicted in Figure 4.2(a), the proposed algorithms (BETC-GLM, BEXP3) outperform the baseline algorithms in terms of cumulative regret when reaching the end of time



(a) Generated Hard Case



(b) EventTime

Figure 4.2: The regret of the proposed algorithms (BETC-GLM, BEXP3) and the baseline algorithms (UCB-Borda, DEXP3, ETC-Borda).

horizon T. For UCB-Borda, since it is not tailored for the dueling regret definition, it suffers from a linear regret as its second arm is always sampled uniformly at random, leading to a constant regret per round. DEXP3 and ETC-Borda are two algorithms designed for K-armed dueling bandits. Both are unable to utilize contextual information and thus demand more exploration. As expected, their regrets are higher than BETC-GLM or BEXP3. We do not fine-tune the hyper-parameters of both algorithms. In Section 4.13, we show fine-tuning the error tolerance ϵ of BETC-GLM can further reduce the regret, and the two algorithms perform equally well.

Real-world Dataset To showcase the performance of the algorithms in a real-world setting, we use EventTime dataset (Zhang et al., 2016). In this dataset, $K = 100$ historical events are compared in a pairwise fashion by crowd-sourced workers. We first calculate the empirical preference probabilities $\widetilde{p}_{i,j}$ from the collected responses. A visualized preferential matrix consisting of $\widetilde{p}_{i,j}$ is shown in Figure 4.3 in Section 4.11.1, which demonstrates that

STI and SST conditions hardly hold in reality. During simulation, $\widetilde{p}_{i,j}$ is the parameter of the Bernoulli distribution that is used to generate the responses whenever a pair (i,j) is queried. The contextual vectors $\boldsymbol{\phi}_{i,j}$ are generated randomly from $\{-1,+1\}^5$. For simplicity, we assign the item pairs that have the same probability value with the same contextual vector, i.e., if $\widetilde{p}_{i,j} = \widetilde{p}_{k,l}$ then $\boldsymbol{\phi}_{i,j} = \boldsymbol{\phi}_{k,l}$. The MLE estimator $\widehat{\boldsymbol{\theta}}$ in (4.5.1) is obtained to construct the recovered preference probability $\widehat{p}_{i,j} := \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}})$ where $\mu(x) = 1/(1+e^{-x})$ is the logistic function. We ensure that the recovered preference probability $\widehat{p}_{i,j}$ is close to $\widetilde{p}_{i,j}$, so that $\boldsymbol{\phi}_{i,j}$ are informative enough. As shown in Figure 4.2(b), our algorithm outperforms the baseline methods as expected. In particular, the gap between our algorithm and the baselines is even larger than that under the generated hard case. In both settings, our algorithms demonstrated a stable performance with negligible variance.

4.8 Conclusion

In this chapter, we introduced Borda regret into the generalized linear dueling bandits setting, along with an explore-then-commit type algorithm BETC-GLM and an EXP3 type algorithm BEXP3. The algorithms can achieve a nearly optimal regret upper bound, which we corroborate with a matching lower bound. The theoretical performance of the algorithms is verified empirically. It demonstrates superior performance compared to other baseline methods.

4.9 Omitted Proofs

4.9.1 Omitted Proof in Section 4.4

The proof relies on a class of hard-to-learn instances. We first present the construction again for completeness.

For any $d > 0$, we construct a hard instance with 2^{d+1} items (indexed from 0 to $2^{d+1}-1$).

We construct the hard instance $p_{i,j}^{\boldsymbol{\theta}}$ for any $\boldsymbol{\theta} \in \{-\Delta, +\Delta\}^d$ as:

$$p_{i,j}^{\boldsymbol{\theta}} = \begin{cases} \frac{1}{2}, \text{ if } i < 2^d, j < 2^d \\ \frac{1}{2}, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ \frac{3}{4}, \text{ if } i < 2^d, j \geqslant 2^d \\ \frac{1}{4}, \text{ if } i \geqslant 2^d, j < 2^d \end{cases} + \langle \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta} \rangle, \tag{4.9.1}$$

where the feature vectors $\boldsymbol{\phi}_{i,j}$ and the parameter $\boldsymbol{\theta}$ are of dimension d, and have the following forms:

$$\boldsymbol{\phi}_{i,j} = \begin{cases} \mathbf{0}, \text{ if } i < 2^d, j < 2^d \\ \mathbf{0}, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ \mathbf{bit}(i), \text{ if } i < 2^d, j \geqslant 2^d \\ -\mathbf{bit}(j), \text{ if } i \geqslant 2^d, j < 2^d, \end{cases}$$

where $\mathbf{bit}(\cdot)$ is the (shifted) bit representation of non-negative integers, i.e., suppose $x = b_0 \times 2^0 + b_1 \times 2^1 + \cdots + b_{d-1} \times 2^{d-1}$, then $\mathbf{bit}(x) = 2\boldsymbol{b} - 1$. Note that $\mathbf{bit}(\cdot) \in \{-1, +1\}^d$, and $\boldsymbol{\phi}_{i,j} = -\boldsymbol{\phi}_{j,i}$.

Remark 4.9.1 ($d+1$-dimensional instance)**.** The hard instance described above does not strictly satisfy the assumption that $p_{i,j}^{\boldsymbol{\theta}} = \langle \boldsymbol{\theta}, \boldsymbol{\phi}_{i,j} \rangle$, but can be easily fixed by appending an additional dimension to address the bias term defined in (4.9.1). More specifically, we can set $F(x) = \frac{1}{2} + x$ and $p_{i,j}^{\boldsymbol{\theta}} = F(\langle \widetilde{\boldsymbol{\phi}}_{i,j}, \widetilde{\boldsymbol{\theta}} \rangle)$, where $\widetilde{\boldsymbol{\theta}} \in \{-\Delta, +\Delta\}^d \times \{\frac{1}{4}\} \subset \mathbb{R}^{d+1}$ and $\widetilde{\boldsymbol{\phi}}_{i,j} = (\boldsymbol{\phi}_{i,j}, c_{i,j})$, with $c_{i,j} = \begin{cases} 0, \text{ if } i < 2^d, j < 2^d \\ 0, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ 1, \text{ if } i < 2^d, j \geqslant 2^d \\ -1, \text{ if } i \geqslant 2^d, j < 2^d. \end{cases}$ To ensure $\|\widetilde{\boldsymbol{\phi}}_{i,j}\|_2 \leqslant 1$, we can further set $\widetilde{\boldsymbol{\phi}}_{i,j} \leftarrow (d+1)^{-1/2} \widetilde{\boldsymbol{\phi}}_{i,j}$ and $\widetilde{\boldsymbol{\theta}} \leftarrow (d+1)^{1/2} \widetilde{\boldsymbol{\theta}}$.

We rewrite (4.9.1) as:

$$p_{i,j}^{\boldsymbol{\theta}} = \begin{cases} \frac{1}{2}, \text{ if } i < 2^d, j < 2^d \\ \frac{1}{2}, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ \frac{3}{4}, \text{ if } i < 2^d, j \geqslant 2^d \\ \frac{1}{4}, \text{ if } i \geqslant 2^d, j < 2^d \end{cases} + \begin{cases} 0, \text{ if } i < 2^d, j < 2^d \\ 0, \text{ if } i \geqslant 2^d, j \geqslant 2^d \\ \langle \mathbf{bit}(i), \boldsymbol{\theta} \rangle, \text{ if } i < 2^d, j \geqslant 2^d \\ -\langle \mathbf{bit}(j), \boldsymbol{\theta} \rangle, \text{ if } i \geqslant 2^d, j < 2^d, \end{cases} \tag{4.9.2}$$

and the Borda scores are:

$$B^{\boldsymbol{\theta}}(i) = \begin{cases} \frac{5}{8} + \frac{1}{2}\langle \mathbf{bit}(i), \boldsymbol{\theta} \rangle, \text{ if } i < 2^d, \\ \frac{3}{8}, \text{ if } i \geqslant 2^d. \end{cases}$$

Intuitively, the former half arms indexed from 0 to $2^d - 1$ are "good" arms (one among them is optimal), while the latter half arms are "bad" arms. It is clear that choosing a "bad" arm i will incur regret $B(i^*) - B(i) \geqslant 1/4$.

Now we are ready to present the proof.

Proof of Theorem 4.4.1. First, we present the following lemma:

Lemma 4.9.2. Under the hard instance, we constructed above, for any algorithm $\mathcal{A}$ that ever makes queries $i_t \geqslant 2^d$, there exists another algorithm $\mathcal{A}'$ that only makes queries $i_t < 2^d$ for every $t > 0$ and always achieves no larger expected regret than $\mathcal{A}$.

Proof of Lemma 4.9.2. The proof is done by reduction. For any algorithm $\mathcal{A}$, we wrap $\mathcal{A}$ with such a agent $\mathcal{A}'$:

1. If $\mathcal{A}$ queries (i_t, j_t) with $i_t < 2^d$, the agent $\mathcal{A}'$ will pass the same query (i_t, j_t) to the environment and send the feedback r_t to $\mathcal{A}$;
2. If $\mathcal{A}$ queries (i_t, j_t) with $i_t \geqslant 2^d, j_t < 2^d$, the agent $\mathcal{A}'$ will pass the query (j_t, i_t) to the environment and send the feedback $1 - r_t$ to $\mathcal{A}$;

3. If $\mathcal{A}$ queries (i_t, j_t) with $i_t \geqslant 2^d, j_t \geqslant 2^d$, the agent $\mathcal{A}'$ will uniform-randomly choose i'_t from 0 to $2^d - 1$, pass the query (i'_t, i'_t) to the environment and send the feedback r_t to $\mathcal{A}$.

For each of the cases defined above, the probabilistic model of bandit feedback for $\mathcal{A}$ is the same as if $\mathcal{A}$ is directly interacting with the original environment. For Case 1, the claim is trivial. For Case 2, the claim holds because of the symmetry of our model, that is $p^{\boldsymbol{\theta}}_{i,j} = 1 - p^{\boldsymbol{\theta}}_{j,i}$. For Case 3, both will return r_t following Bernoulli(1/2). Therefore, the expected regret of $\mathcal{A}$ in this environment wrapped by $\mathcal{A}'$ is equal to the regret of $\mathcal{A}$ in the original environment.

Meanwhile, we will show $\mathcal{A}'$ will incur no larger regret than $\mathcal{A}$. For the first two cases, $\mathcal{A}'$ will incur the same one-step regret as $\mathcal{A}$. For the third case, we know that $B^{\boldsymbol{\theta}}(i_t) = B^{\boldsymbol{\theta}}(j_t) = \frac{3}{8}$, while $\mathbb{E}[B^{\boldsymbol{\theta}}(i'_t)] = \frac{5}{8} + \frac{1}{2}\langle \mathbb{E}_{i'_t}[\mathbf{bit}(i'_t)], \boldsymbol{\theta}\rangle = \frac{5}{8} + \frac{1}{2}\langle \mathbf{0}, \boldsymbol{\theta}\rangle = \frac{5}{8}$, meaning that the one-step regret is smaller. □

Lemma 4.9.2 ensures it is safe to assume $i_t < 2^d$. For any $\boldsymbol{\theta}$ and $k \in [d]$, define

$$\mathbb{P}_{\boldsymbol{\theta},k} := \mathbb{P}_{\boldsymbol{\theta}}\left(\sum_{t=1}^{T} \mathbb{1}\{\mathbf{bit}^{[k]}(i_t) \neq \operatorname{sign}(\boldsymbol{\theta}^{[k]})\} \geqslant \frac{T}{2}\right),$$

where the superscript $^{[k]}$ over a vector denotes taking the k-th entry of the vector. Meanwhile, we define $\boldsymbol{\theta}^{\setminus k}$ to satisfy $(\boldsymbol{\theta}^{\setminus k})^{[k]} = -\boldsymbol{\theta}^{[k]}$ and be the same as $\boldsymbol{\theta}$ at all other entries. We have

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta}^{\setminus k},k} &:= \mathbb{P}_{\boldsymbol{\theta}^{\setminus k}}\left(\sum_{t=1}^{T} \mathbb{1}\left\{\mathbf{bit}^{[k]}(i_t) \neq \operatorname{sign}\big((\boldsymbol{\theta}^{\setminus k})^{[k]}\big)\right\} \geqslant \frac{T}{2}\right) \\
&= \mathbb{P}_{\boldsymbol{\theta}^{\setminus k}}\left(\sum_{t=1}^{T} \mathbb{1}\{\mathbf{bit}^{[k]}(i_t) = \operatorname{sign}(\boldsymbol{\theta}^{[k]})\} \geqslant \frac{T}{2}\right) \\
&= \mathbb{P}_{\boldsymbol{\theta}^{\setminus k}}\left(\sum_{t=1}^{T} \mathbb{1}\{\mathbf{bit}^{[k]}(i_t) \neq \operatorname{sign}(\boldsymbol{\theta}^{[k]})\} < \frac{T}{2}\right).
\end{aligned}$$

Denote $\mathbb{P}_{\boldsymbol{\theta},\mathcal{A}}(i_1, j_1, r_1, i_2, j_2, r_2, \dots)$ as the canonical probability distribution of algorithm $\mathcal{A}$ under the model $\mathbb{P}_{\boldsymbol{\theta}}$. By the Bretagnolle–Huber inequality and the decomposition of the relative entropy, we have

$$
\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta},k} + \mathbb{P}_{\boldsymbol{\theta}^{\backslash k},k} &\geqslant \frac{1}{2}\exp\big(- \mathrm{KL}(\mathbb{P}_{\boldsymbol{\theta},\mathcal{A}} \| \mathbb{P}_{\boldsymbol{\theta}^{\backslash k},\mathcal{A}})\big) \\
&\geqslant \frac{1}{2}\exp\bigg(- \mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T} \mathrm{KL}\Big(p_{i,j}^{\boldsymbol{\theta}} \Big\| p_{i,j}^{\boldsymbol{\theta}^{\backslash k}}\Big)\bigg]\bigg) \\
&\geqslant \frac{1}{2}\exp\bigg(- \mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T} 10\langle \boldsymbol{\phi}_{i_t,j_t}, \boldsymbol{\theta} - \boldsymbol{\theta}^{\backslash k}\rangle^2\bigg]\bigg) \\
&= \frac{1}{2}\exp\bigg(- \mathbb{E}_{\boldsymbol{\theta}}\bigg[40\Delta^2 \sum_{t=1}^{T} \mathbb{1}\{i_t < 2^d \wedge j_t \geqslant 2^d\}\bigg]\bigg),
\end{aligned}
$$

where the first inequality comes from the Bretagnolle–Huber inequality; the second inequality is the decomposition of the relative entropy; the third inequality holds because the Bernoulli KL divergence $KL(p\|p+x)$ is 10-strongly convex in x for any fixed $p \in [1/8, 7/8]$, and indeed $p_{i,j}^{\boldsymbol{\theta}} \in [1/8, 7/8]$ as long as $d\Delta \leqslant 1/8$; the last equation holds because $\boldsymbol{\phi}_{i_t,j_t}$ has non-zero entries only when (i_t, j_t) belongs to that specific regions.

From now on, we denote $N(T) := \sum_{t=1}^{T} \mathbb{1}\{i_t < 2^d \wedge j_t \geqslant 2^d\}$. Further averaging over all $\boldsymbol{\theta} \in \{-\Delta, +\Delta\}^d$, we have

$$
\begin{aligned}
\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d} \mathbb{P}_{\boldsymbol{\theta},k} &\geqslant \frac{1}{4}\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d} \exp\big(- 40\Delta^2 \mathbb{E}_{\boldsymbol{\theta}}[N(T)]\big) \\
&\geqslant \frac{1}{4}\exp\bigg(- 40\Delta^2 \frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d} \mathbb{E}_{\boldsymbol{\theta}}[N(T)]\bigg),
\end{aligned}
$$

where the first inequality is from averaging over all $\boldsymbol{\theta}$; the second inequality is from Jensen's inequality.

Utilizing the inequality above, we establish that

$$
\begin{aligned}
\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathrm{Regret}(T;\boldsymbol{\theta},\mathcal{A}) &\geqslant \frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T}B^{\boldsymbol{\theta}}(i^*)-B^{\boldsymbol{\theta}}(i_t)\bigg]\\
&= \frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T}\langle\boldsymbol{\theta},\mathbf{sign}(\boldsymbol{\theta})-\mathbf{bit}(i_t)\rangle\bigg]\\
&= \frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T}\sum_{k=1}^{d}2\Delta\,\mathbb{1}\{\mathbf{bit}^{[k]}(i_t)\neq\mathrm{sign}(\boldsymbol{\theta}^{[k]})\}\bigg]\\
&= \frac{2\Delta}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\sum_{k=1}^{d}\mathbb{E}_{\boldsymbol{\theta}}\bigg[\sum_{t=1}^{T}\mathbb{1}\{\mathbf{bit}^{[k]}(i_t)\neq\mathrm{sign}(\boldsymbol{\theta}^{[k]})\}\bigg]\\
&\geqslant \frac{2\Delta}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\sum_{k=1}^{d}\mathbb{P}_{\boldsymbol{\theta},k}\cdot\frac{T}{2}\\
&\geqslant \frac{\Delta dT}{4}\exp\bigg(-40\Delta^2\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}[N(T)]\bigg), \qquad (4.9.3)
\end{aligned}
$$

where the first inequality comes from the Borda regret; the second inequality comes from the inequality $\mathbb{E}[X]\geqslant a\mathbb{P}(X\geqslant a)$ for any non-negative random variable; the last inequality is from rearranging terms and invoking the results above.

Meanwhile, we have (remember $N(T):=\sum_{t=1}^{T}\mathbb{1}\{i_t<2^d\wedge j_t\geqslant 2^d\}$)

$$
\begin{aligned}
\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathrm{Regret}(T;\boldsymbol{\theta},\mathcal{A}) &\geqslant \frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}\bigg[\frac{1}{4}\sum_{t=1}^{T}\mathbb{1}\{i_t<2^d\wedge j_t\geqslant 2^d\}\bigg]\\
&= \frac{1}{4}\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}[N(T)], \qquad (4.9.4)
\end{aligned}
$$

where the first inequality comes from that any items $i\geqslant 2^d$ will incur at least $1/4$ regret.

Combining (4.9.3) and (4.9.4) together and denoting that $X=\frac{1}{2^d}\sum_{\boldsymbol{\theta}\in\{-\Delta,+\Delta\}^d}\mathbb{E}_{\boldsymbol{\theta}}[N(T)]$, we have that for any algorithm $\mathcal{A}$, there exists some $\boldsymbol{\theta}$, such that (set $\Delta = \frac{d^{-1/3}T^{-1/3}}{\sqrt{40}}$)

$$\begin{aligned}
\text{Regret}(T;\boldsymbol{\theta},\mathcal{A}) &\geqslant \max\left\{\frac{\Delta dT}{4}\exp(-40\Delta^2 X), \frac{X}{4}\right\} \\
&= \max\left\{\frac{d^{2/3}T^{2/3}}{4\sqrt{40}}\exp(-d^{-2/3}T^{-2/3}X), \frac{X}{4}\right\} \\
&\geqslant \frac{d^{2/3}T^{2/3}}{4\sqrt{40}}\max\left\{\exp(-d^{-2/3}T^{-2/3}X), d^{-2/3}T^{-2/3}X\right\} \\
&\geqslant \frac{d^{2/3}T^{2/3}}{8\sqrt{40}},
\end{aligned}$$

where the first inequality is the combination of (4.9.3) and (4.9.4); the second inequality is a rearrangement and loosely lower bounds the constant; the last is due to $\max\{e^{-y}, y\} > 1/2$ for any y. □

4.9.2 Omitted Proof in Section 4.5

We first introduce the lemma about the theoretical guarantee of G-optimal design: given an action set $\mathcal{X} \subseteq \mathbb{R}^d$ that is compact and $\text{span}(\mathcal{X}) = \mathbb{R}^d$. A fixed design $\pi(\cdot) : \mathcal{X} \to [0,1]$ satisfies $\sum_{\mathbf{x}\in\mathcal{X}} \pi(\mathbf{x}) = 1$. Define $\mathbf{V}(\pi) := \sum_{\mathbf{x}\in\mathcal{X}} \pi(\mathbf{x})\mathbf{x}\mathbf{x}^\top$ and $g(\pi) := \max_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x}\|^2_{\mathbf{V}(\pi)^{-1}}$.

Lemma 4.9.3 (The Kiefer–Wolfowitz Theorem, Section 21.1, Lattimore and Szepesvári (2020))**.** There exists an optimal design $\pi^*(\cdot)$ such that $|\text{supp}(\pi)| \leqslant d(d+1)/2$, and satisfies:

1. $g(\pi^*) = d$.

2. π^* is the minimizer of $g(\cdot)$.

The following lemma is also useful to show that under mild conditions, the minimum eigenvalue of the design matrix can be lower-bounded:

Lemma 4.9.4 (Proposition 1, Li et al. 2017)**.** Define $\mathbf{V}_\tau = \sum_{t=1}^{\tau} \boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top$, where each (i_t, j_t) is drawn i.i.d. from some distribution ν. Suppose $\lambda_{\min}\big(\mathbb{E}_{(i,j)\sim\nu}[\boldsymbol{\phi}_{i,j}^\top\boldsymbol{\phi}_{i,j}]\big) \geqslant \lambda_0$, and

$$\tau \geqslant \left(\frac{C_1\sqrt{d} + C_2\sqrt{\log(1/\delta)}}{\lambda_0}\right)^2 + \frac{2B}{\lambda_0},$$

where C_1 and C_2 are some universal constants. Then with probability at least $1-\delta$,

$$\lambda_{\min}(\mathbf{V}_\tau) \geqslant B.$$

4.9.2.1 Proof of Theorem 4.5.1

The proof relies on the following lemma to establish an upper bound on $|\langle \boldsymbol{\phi}_{i,j}, \widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}^* \rangle|$.

Lemma 4.9.5 (extracted from Lemma 3, Li et al. (2017))**.** Suppose $\lambda_{\min}(\mathbf{V}_{\tau+N}) \geqslant 1$. For any $\delta > 0$, with probability at least $1-\delta$, we have

$$\|\widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}^*\|_{\mathbf{V}_{\tau+N}} \leqslant \frac{1}{\kappa}\sqrt{\frac{d}{2}\log(1+2(\tau+N)/d) + \log(1/\delta)}.$$

Proof of Theorem 4.5.1. The proof can be divided into three steps: 1. invoke Lemma 4.9.4 to show that the initial τ rounds for exploration will guarantee $\lambda_{\min}(\mathbf{V}_\tau) \geqslant 1$; 2. invoke Lemma 4.9.3 to obtain an optimal design π and utilize Cauchy-Schwartz inequality to show that $|\langle \widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}, \boldsymbol{\phi}_{i,j} \rangle| \leqslant 3\epsilon/\kappa$; 3. balance the not yet determined ϵ to obtain the regret upper bound.

Since we set τ such that

$$\begin{aligned} \tau &= C_4 \lambda_0^{-2}(d + \log(1/\delta)) \\ &\geqslant \left(\frac{C_1\sqrt{d} + C_2\sqrt{\log(1/\delta)}}{\lambda_0}\right)^2 + \frac{2}{\lambda_0}, \end{aligned}$$

with a large enough universal constant C_4, by Theorem 4.9.4 to obtain that with probability at least $1-\delta$,

$$\lambda_{\min}(\mathbf{V}_\tau) \geqslant 1. \tag{4.9.5}$$

From now on, we assume (4.9.5) always holds.

Define $N := \sum_{i,j} N(i,j)$, $\mathbf{V}_{\tau+1:\tau+N} := \sum_{t=\tau+1}^{\tau+N} \boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top$, $\mathbf{V}_{\tau+N} := \mathbf{V}_\tau + \mathbf{V}_{\tau+1:\tau+N}$. Given the optimal design $\pi(i,j)$, the algorithm queries the pair $(i,j) \in \operatorname{supp}(\pi)$ for exactly $N(i,j) =$

$\lceil d\pi(i,j)/\epsilon^2 \rceil$ times. Therefore, the design matrix $\mathbf{V}_{\tau+N}$ satisfies

$$\begin{aligned}
\mathbf{V}_{\tau+N} &\succeq \mathbf{V}_{\tau+1:\tau+N} \\
&= \sum_{i,j} N(i,j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top \\
&\succeq \sum_{i,j} \frac{d\pi(i,j)}{\epsilon^2}\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top \\
&= \frac{d}{\epsilon^2}\mathbf{V}(\pi),
\end{aligned}$$

where $\mathbf{V}(\pi) := \sum_{i,j} \pi(i,j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$. The first inequality holds because $\mathbf{V}_\tau$ is positive semi-definite, and the second inequality holds due to the choice of $N(i,j)$.

When (4.9.5) holds, from Theorem 4.9.5, we have with probability at least $1-\delta$, that for each $\boldsymbol{\phi}_{i,j}$,

$$\begin{aligned}
|\langle \widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*, \boldsymbol{\phi}_{i,j}\rangle| &\leqslant \|\widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}^*\|_{\mathbf{V}_{\tau+N}} \cdot \|\boldsymbol{\phi}_{i,j}\|_{\mathbf{V}_{\tau+N}^{-1}} \\
&\leqslant \|\widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}^*\|_{\mathbf{V}_{\tau+N}} \cdot \frac{\epsilon\|\boldsymbol{\phi}_{i,j}\|_{\mathbf{V}(\pi)^{-1}}}{\sqrt{d}} \\
&\leqslant \|\widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}^*\|_{\mathbf{V}_{\tau+N}} \cdot \epsilon \\
&\leqslant \frac{\epsilon}{\kappa} \cdot \sqrt{\frac{d}{2}\log(1+2(\tau+N)/d) + \log(1/\delta)} \qquad (4.9.6)
\end{aligned}$$

where the first inequality is due to the Cauchy-Schwartz inequality; the second inequality holds because $\mathbf{V}_{\tau+N} \succeq \frac{d}{\epsilon^2}\mathbf{V}(\pi)$; the third inequality holds because π is an optimal design and by Lemma 4.9.3, $\|\boldsymbol{\phi}_{i,j}\|_{\mathbf{V}(\pi)^{-1}}^2 \leqslant d$; the last inequality comes from Theorem 4.9.5.

To summarize, we have that with probability at least $1-2\delta$, for every $i \in [K]$,

$$
\begin{aligned}
|\widehat{B}(i) - B(i)| &= \left| \frac{1}{K} \sum_{j=1}^{K} \left(\mu(\boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*) - \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}}) \right) \right| \\
&\leqslant \frac{1}{K} \sum_{j=1}^{K} \left| \mu(\boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*) - \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}}) \right| \\
&\leqslant \frac{L_\mu}{K} \sum_{j=1}^{K} \left| \boldsymbol{\phi}_{i,j}^\top (\boldsymbol{\theta}^* - \widehat{\boldsymbol{\theta}}) \right| \\
&\leqslant \frac{3L_\mu \epsilon}{\kappa} \cdot \sqrt{\frac{d}{2} \log(1 + 2(\tau + N)/d) + \log(1/\delta)}, \qquad (4.9.7)
\end{aligned}
$$

where the first equality is by the definition of the empirical/true Borda score; the first inequality is due to the triangle inequality; the second inequality is from the Lipschitz-ness of $\mu(\cdot)$ ($L_\mu = 1/4$ for the logistic function); the last inequality holds due to (4.9.6). This further implies the gap between the empirical Borda winner and the true Borda winner is bounded by:

$$
\begin{aligned}
B(i^*) - B(\widehat{i}) &= B(i^*) - \widehat{B}(i^*) + \widehat{B}(i^*) - B(\widehat{i}) \\
&\leqslant B(i^*) - \widehat{B}(i^*) + \widehat{B}(\widehat{i}) - B(\widehat{i}) \\
&\leqslant \frac{6L_\mu \epsilon}{\kappa} \cdot \sqrt{\frac{d}{2} \log(1 + 2(\tau + N)/d) + \log(1/\delta)},
\end{aligned}
$$

where the first inequality holds due to the definition of $\widehat{i}$, i.e., $\widehat{B}(\widehat{i}) \geqslant \widehat{B}(i)$ for any i; the last inequality holds due to (4.9.7).

Meanwhile, since $N := \sum_{(i,j)\in \mathrm{supp}(\pi)} N(i,j)$ and $|\mathrm{supp}(\pi)| \leqslant d(d+1)/2$ from Lemma 4.9.3, we have that

$$
N \leqslant d(d+1)/2 + \frac{d}{\epsilon^2},
$$

because $\lceil x \rceil < x + 1$.

Therefore, with probability at least $1-2\delta$, the regret is bounded by:

$$\begin{aligned}\mathrm{Regret}(T) &= \mathrm{Regret}_{1:\tau} + \mathrm{Regret}_{\tau+1:\tau+N} + \mathrm{Regret}_{\tau+N+1:T}\\ &\leqslant \tau + N + \frac{12L_\mu \epsilon T}{\kappa}\cdot\sqrt{\frac{d}{2}\log(1+2(\tau+N)/d)+\log(1/\delta)}\\ &\leqslant \tau + d(d+1)/2 + \frac{d}{\epsilon^2} + \frac{12L_\mu\epsilon T}{\kappa}\cdot O\left(d^{1/2}\sqrt{\log\left(\frac{T}{d\delta}\right)}\right)\\ &= O\left(\kappa^{-1}d^{2/3}T^{2/3}\sqrt{\log\left(\frac{T}{d\delta}\right)}\right),\end{aligned}$$

where the first equation is simply dividing the regret into 3 stages: 1 to τ, $\tau+1$ to $\tau+N$, and $\tau+N+1$ to T; the second inequality is simply bounding the one-step regret from 1 to $\tau+N$ by 1, while for $t>\tau+N$, we have shown that the one-step regret is guaranteed to be smaller than $12L_\mu\epsilon\sqrt{d\log(1+2(\tau+N)/d)+\log(1/\delta)}/\sqrt{2}\kappa$. The last line holds because we set $\tau = O(d+\log(1/\delta))$ and $\epsilon = d^{1/6}T^{-1/3}$. Note that to ensure $\tau+N<T$, it suffices to assume $T=\Omega(d^2)$.

By setting $\delta = T^{-1}$, we can show that the expected regret of Algorithm 16 is bounded by

$$\widetilde{O}\big(\kappa^{-1}(d^{2/3}T^{2/3})\big).$$

$\square$

4.9.2.2 Proof of Theorem 4.5.5

The following lemma characterizes the non-asymptotic behavior of the MLE estimator. It is extracted from Li et al. (2017).

Lemma 4.9.6 (Theorem 1, Li et al. 2017)**.** Define $\mathbf{V}_s = \sum_{t=1}^{s}\boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top$, and $\widehat{\boldsymbol{\theta}}_s$ as the MLE estimator (4.5.1) in round s. If $\mathbf{V}_s$ satisfies

$$\lambda_{\min}(\mathbf{V}_s) \geqslant \frac{512M_\mu^2(d^2+\log(3/\delta))}{\kappa^4}, \tag{4.9.8}$$

then for any fixed $\mathbf{x} \in \mathbb{R}^d$, with probability at least $1-\delta$,

$$|\langle \widehat{\boldsymbol{\theta}}_s - \boldsymbol{\theta}^*, \mathbf{x}\rangle| \leqslant \frac{3}{\kappa}\sqrt{\|\mathbf{x}\|^2_{\mathbf{V}_s^{-1}} \log(3/\delta)}.$$

Proof of Theorem 4.5.5. The proof can be essentially divided into three steps: 1. invoke Lemma 4.9.4 to show that the initial τ rounds for exploration will guarantee (4.9.8) be satisfied; 2. invoke Lemma 4.9.3 to obtain an optimal design π and utilize Lemma 4.9.6 to show that $|\langle \widehat{\boldsymbol{\theta}}_{\tau+N} - \boldsymbol{\theta}, \boldsymbol{\phi}_{i,j}\rangle| \leqslant 3\epsilon/\kappa$; 3. balance the not yet determined ϵ to obtain the regret upper bound.

First, we explain why we assume

$$T \geqslant \frac{C_3}{\kappa^6 \lambda_0^{3/2}} \max\left\{d^{5/2}, \frac{\log(K^2/\delta)}{\sqrt{d}}\right\}.$$

To ensure (4.9.8) in Theorem 4.9.6 can hold, we resort to Theorem 4.9.4, that is

$$\tau \geqslant \left(\frac{C_1\sqrt{d} + C_2\sqrt{\log(1/\delta)}}{\lambda_0}\right)^2 + \frac{2B}{\lambda_0},$$
$$B := \frac{512 M_\mu^2 (d^2 + \log(3/\delta))}{\kappa^4}.$$

Since we set $\tau = (d\log(K^2/\delta))^{1/3} T^{2/3}$, this means T should be large enough, so that

$$(d\log(K^2/\delta))^{1/3} T^{2/3} \geqslant \left(\frac{C_1\sqrt{d} + C_2\sqrt{\log(1/\delta)}}{\lambda_0}\right)^2 + \frac{1024 M_\mu^2 (d^2 + \log(3K^2/\delta))}{\kappa^4 \lambda_0}.$$

With a large enough universal constant C_3, it is easy to verify that the inequality above will hold as long as

$$T \geqslant \frac{C_3}{\kappa^6 \lambda_0^{3/2}} \max\left\{d^{5/2}, \frac{\log(K^2/\delta)}{\sqrt{d}}\right\}.$$

By Lemma 4.9.4, we have that with probability at least $1-\delta$,

$$\lambda_{\min}(\mathbf{V}_\tau) \geqslant \frac{512 M_\mu^2 (d^2 + \log(3K^2/\delta))}{\kappa^4}. \tag{4.9.9}$$

From now on, we assume (4.9.9) always holds.

Define $N := \sum_{i,j} N(i,j)$, $\mathbf{V}_{\tau+1:\tau+N} := \sum_{t=\tau+1}^{\tau+N} \boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top$, $\mathbf{V}_{\tau+N} := \mathbf{V}_\tau + \mathbf{V}_{\tau+1:\tau+N}$. Given the optimal design $\pi(i,j)$, the algorithm queries each pair $(i,j) \in \mathrm{supp}(\pi)$ for exactly $N(i,j) = \lceil d\pi(i,j)/\epsilon^2 \rceil$ times. Therefore, the design matrix $\mathbf{V}_{\tau+N}$ satisfies

$$\begin{aligned}
\mathbf{V}_{\tau+N} &\succeq \mathbf{V}_{\tau+1:\tau+N} \\
&= \sum_{i,j} N(i,j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top \\
&\succeq \sum_{i,j} \frac{d\pi(i,j)}{\epsilon^2}\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top \\
&= \frac{d}{\epsilon^2}\mathbf{V}(\pi),
\end{aligned}$$

where $\mathbf{V}(\pi) := \sum_{i,j} \pi(i,j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$. The first inequality holds because $\mathbf{V}_\tau$ is positive semi-definite, and the second inequality holds due to the choice of $N(i,j)$.

To invoke Lemma 4.9.6, notice that $\lambda_{\min}(\mathbf{V}) \geqslant \lambda_{\min}(\mathbf{V}_\tau)$. Along with (4.9.9), by Lemma 4.9.6, we have for any fixed $\boldsymbol{\phi}_{i,j}$, with probability at least $1 - \delta/K^2$, that

$$\begin{aligned}
|\langle \widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*, \boldsymbol{\phi}_{i,j} \rangle| &\leqslant \frac{3}{\kappa}\sqrt{\|\boldsymbol{\phi}_{i,j}\|^2_{\mathbf{V}_{\tau+N}^{-1}} \log(3K^2/\delta)} \\
&\leqslant \frac{3}{\kappa}\sqrt{\frac{\epsilon^2}{d} \cdot \|\boldsymbol{\phi}_{i,j}\|^2_{\mathbf{V}(\pi)^{-1}} \log(3K^2/\delta)} \\
&= \frac{3\epsilon}{\kappa}\sqrt{\frac{\|\boldsymbol{\phi}_{i,j}\|^2_{\mathbf{V}(\pi)^{-1}}}{d}} \cdot \sqrt{\log(3K^2/\delta)} \\
&\leqslant \frac{3\epsilon}{\kappa} \cdot \sqrt{\log(3K^2/\delta)}, \qquad (4.9.10)
\end{aligned}$$

where the first inequality comes from Lemma 4.9.6; the second inequality holds because $\mathbf{V}_{\tau+N} \succeq \frac{d}{\epsilon^2}\mathbf{V}(\pi)$; the last inequality holds because π is an optimal design and by Lemma 4.9.3, $\|\boldsymbol{\phi}_{i,j}\|^2_{\mathbf{V}(\pi)^{-1}} \leqslant d$.

Taking union bound for each $(i,j) \in [K] \times [K]$, we have that with probability at least

$1-\delta$, for every $i \in [K]$,

$$
\begin{aligned}
|\widehat{B}(i) - B(i)| &= \left| \frac{1}{K} \sum_{j=1}^{K} \big(\mu(\boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*) - \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}}) \big) \right| \\
&\leqslant \frac{1}{K} \sum_{j=1}^{K} \left| \mu(\boldsymbol{\phi}_{i,j}^\top \boldsymbol{\theta}^*) - \mu(\boldsymbol{\phi}_{i,j}^\top \widehat{\boldsymbol{\theta}}) \right| \\
&\leqslant \frac{L_\mu}{K} \sum_{j=1}^{K} \left| \boldsymbol{\phi}_{i,j}^\top (\boldsymbol{\theta}^* - \widehat{\boldsymbol{\theta}}) \right| \\
&\leqslant \frac{3L_\mu \epsilon}{\kappa} \cdot \sqrt{\log(3K^2/\delta)}, \qquad (4.9.11)
\end{aligned}
$$

where the first equality is by the definition of the empirical/true Borda score; the first inequality is due to the triangle inequality; the second inequality is from the Lipschitz-ness of $\mu(\cdot)$ ($L_\mu = 1/4$ for the logistic function); the last inequality holds due to (4.9.10). This further implies the gap between the empirical Borda winner and the true Borda winner is bounded by:

$$
\begin{aligned}
B(i^*) - B(\widehat{i}) &= B(i^*) - \widehat{B}(i^*) + \widehat{B}(i^*) - B(\widehat{i}) \\
&\leqslant B(i^*) - \widehat{B}(i^*) + \widehat{B}(\widehat{i}) - B(\widehat{i}) \\
&\leqslant \frac{6L_\mu \epsilon}{\kappa} \cdot \sqrt{\log(3K^2/\delta)},
\end{aligned}
$$

where the first inequality holds due to the definition of $\widehat{i}$, i.e., $\widehat{B}(\widehat{i}) \geqslant \widehat{B}(i)$ for any i; the last inequality holds due to (4.9.11).

Meanwhile, since $N := \sum_{(i,j)\in \mathrm{supp}(\pi)} N(i,j)$ and $|\mathrm{supp}(\pi)| \leqslant d(d+1)/2$ from Lemma 4.9.3, we have that

$$
N \leqslant d(d+1)/2 + \frac{d}{\epsilon^2},
$$

because $\lceil x \rceil < x + 1$.

Therefore, with probability at least $1-2\delta$, the regret is bounded by:

$$\begin{aligned}
\text{Regret}(T) &= \text{Regret}_{1:\tau} + \text{Regret}_{\tau+1:\tau+N} + \text{Regret}_{\tau+N+1:T} \\
&\leqslant \tau + N + \frac{12L_\mu\epsilon}{\kappa}T\cdot\sqrt{\log(3K^2/\delta)} \\
&\leqslant \tau + d(d+1)/2 + \frac{d}{\epsilon^2} + \frac{12L_\mu\epsilon}{\kappa}T\cdot\sqrt{\log(3K^2/\delta)} \\
&= O\big(\kappa^{-1}(d\log(K/\delta))^{1/3}T^{2/3}\big),
\end{aligned}$$

where the first equation is simply dividing the regret into 3 stages: 1 to τ, $\tau+1$ to $\tau+N$, and $\tau+N+1$ to T. the second inequality is simply bounding the one-step regret from 1 to $\tau+N$ by 1, while for $t>\tau+N$, we have shown that the one-step regret is guaranteed to be smaller than $12L_\mu\epsilon\sqrt{\log(3K^2/\delta)}/\kappa$. The last line holds because we set $\tau = (d\log(3K^2/\delta))^{1/3}T^{2/3}$ and $\epsilon = d^{1/3}T^{-1/3}\log(3K^2/\delta)^{-1/6}$.

By setting $\delta = T^{-1}$, we can show that the expected regret of Algorithm 16 is bounded by

$$O\big(\kappa^{-1}(d\log(KT))^{1/3}T^{2/3})\big).$$

Note that if there are exponentially many contextual vectors ($K\approx 2^d$), the upper bound becomes $\widetilde{O}(d^{2/3}T^{2/3})$. $\square$

4.9.3 Omitted Proof in Section 4.6

We make the following notation. Let $\mathcal{H}_{t-1} := (q_1, P_1, (i_1, j_1), r_1, \ldots, q_t, P_t)$ denotes the history up to time t. Here P_t means the comparison probability $p^t_{i,j}$ in round t. The following lemmas are used in the proof. We first bound the estimate $\widehat{B}_t(i)$.

Lemma 4.9.7. For all $t\in[T]$, $i\in[K]$, it holds that $\widehat{B}_t(i)\leqslant\lambda_0^{-1}/\gamma^2$.

Proof of Lemma 4.9.7. Using our choice of $q_t\geqslant\gamma/K$, we have the following result for the matrix Q_t:

$$Q_t = \sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top \succeq \gamma^2 \frac{1}{K^2}\sum_{i\in[K]}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top. \tag{4.9.12}$$

Furthermore, we can use the definition of the estimate $\widehat{B}_t(i)$ to show that

$$\begin{aligned}\widehat{B}_t(i) &= \left\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, \widehat{\boldsymbol{\theta}}_t \right\rangle = \left\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\right\rangle r_t(i_t,j_t)\\ &\leqslant \frac{1}{K}\sum_{j\in[K]} \|\boldsymbol{\phi}_{i,j}\|^2_{Q_t^{-1}},\end{aligned}$$

where we use the fact that $|r_t| \leqslant 1$. Let $\boldsymbol{\Sigma} = \frac{1}{K^2}\sum_{i=1}^K\sum_{j=1}^K \boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$. With (4.9.12) we have $Q_t \succeq \gamma^2\boldsymbol{\Sigma}$. Therefore, we can further bound $\widehat{B}_t(i)$ with

$$\begin{aligned}\widehat{B}_t(i) &\leqslant \frac{1}{K\gamma^2}\sum_{j\in[K]} \|\boldsymbol{\phi}_{i,j}\|^2_{\boldsymbol{\Sigma}^{-1}}\\ &\leqslant \frac{1}{\gamma^2}\max_{i,j} \|\boldsymbol{\phi}_{i.j}\|^2_{\boldsymbol{\Sigma}^{-1}}\\ &\leqslant \frac{\lambda_0^{-1}}{\gamma^2},\end{aligned}$$

where the first inequality holds due to (4.9.12) and that $\|\mathbf{x}\|^2_{\mathbf{A}^{-1}} \leqslant \|\mathbf{x}\|^2_{\mathbf{B}^{-1}}$ if $\mathbf{A} \succeq \mathbf{B}$; the third inequality holds because we assume $\lambda_0 \leqslant \lambda_{\min}\big(\frac{1}{K^2}\sum_{i=1}^K\sum_{j=1}^K \boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top\big)$ and $\|\boldsymbol{\phi}_{i,j}\| \leqslant 1$. □

The following lemma proves that our (shifted) estimate is unbiased.

Lemma 4.9.8. For all $t \in [T]$, $i \in [K]$, the following equality holds:

$$\mathbb{E}[\widehat{B}_t(i)] = B_t(i) - \frac{1}{2}.$$

Proof of Lemma 4.9.8. Using our definition of $\widehat{B}_t(i)$, we have

$$\widehat{B}_t(i) = \left\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, \widehat{\boldsymbol{\theta}}_t \right\rangle = \left\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\right\rangle r_t(i_t,j_t).$$

Therefore, by the law of total expectation (tower rule), we have

$$\begin{aligned}
\mathbb{E}[\widehat{B}_t(i)] &= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\mathbb{E}_{(i_t,j_t,r_t)}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\Big\rangle r_t(i_t,j_t)|\mathcal{H}_{t-1}\Big]\Big] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\mathbb{E}_{(i_t,j_t)}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\Big\rangle \mathbb{E}_{r_t}[r_t(i_t,j_t)|(i_t,j_t)]\Big|\mathcal{H}_{t-1}\Big]\Big] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\mathbb{E}_{(i_t,j_t)}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\Big\rangle p_t(i_t,j_t)\Big|\mathcal{H}_{t-1}\Big]\Big]
\end{aligned}$$

Then we use the definition of p_t and the expectation. We can further get the equality

$$\begin{aligned}
\mathbb{E}[\widehat{B}_t(i)] &= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\mathbb{E}_{(i_t,j_t)}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top\boldsymbol{\theta}^*\Big\rangle\Big|\mathcal{H}_{t-1}\Big]\Big] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, Q_t^{-1}\Big(\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top\Big)\boldsymbol{\theta}^*\Big\rangle\Big|\mathcal{H}_{t-1}\Big] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\Big[\Big\langle \frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}, \boldsymbol{\theta}^*\Big\rangle\Big|\mathcal{H}_{t-1}\Big] \\
&= B_t(i) - \frac{1}{2}.
\end{aligned}$$

Therefore, we have completed the proof of Lemma 4.9.8. □

The following lemma is similar to Lemma 5 in Saha et al. (2021b).

Lemma 4.9.9. $\mathbb{E}_{\mathcal{H}_t}[q_t^\top \widehat{B}_t] = \mathbb{E}_{\mathcal{H}_{t-1}}\big[\mathbb{E}_{x\sim q_t}[B_t(x)|\mathcal{H}_{t-1}]\big] - \frac{1}{2}$, $\forall t \in [T]$.

Proof of Lemma 4.9.9. Taking conditional expectation, we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{H}_t}[q_t^\top \widehat{B}_t] &= \mathbb{E}_{\mathcal{H}_t}\left[\sum_{i=1}^K q_t(i)\widehat{B}_t(i)\right] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\left[\sum_{i=1}^K q_t(i)\mathbb{E}_{(i_t,j_t,r_t)}\left[\widehat{B}(i)\Big|\mathcal{H}_{t-1}\right]\right] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\left[\sum_{i=1}^K q_t(i)\left(B_t(i) - \frac{1}{2}\right)\right] \\
&= \mathbb{E}_{\mathcal{H}_{t-1}}\left[\mathbb{E}_{x\sim q_t}\left[B_t(x)\Big|\mathcal{H}_{t-1}\right]\right] - \frac{1}{2},
\end{aligned}$$

where we use the law of total expectation again as well as Theorem 4.9.8. □

The last lemma bounds a summation $\sum_{i\in[K]} q_t(i)\widehat{B}_t(i)^2$, which will be important in our proof.

Lemma 4.9.10. At any time t, $\mathbb{E}[\sum_{i\in[K]} q_t(i)\widehat{B}_t(i)^2] \leqslant d/\gamma$.

Proof of Lemma 4.9.10. Let $\widehat{P}_t(i,j) = \langle \boldsymbol{\phi}_{i,j}, \widehat{\boldsymbol{\theta}}_t\rangle$. Using the definition of $\widehat{B}_t$ and $\widehat{P}_t(i,j)$, we have the following inequality:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i\in[K]} q_t(i)\widehat{B}_t(i)^2\right] &= \mathbb{E}\left[\sum_{i\in[K]} q_t(i)\left(\frac{1}{K}\sum_{j\in[K]}\widehat{P}_t(i,j)\right)^2\right] \\
&\leqslant \mathbb{E}\left[\sum_{i\in[K]} q_t(i)\frac{1}{K}\sum_{j\in[K]}\widehat{P}_t^2(i,j)\right] \\
&= \mathbb{E}\left[\sum_{i\in[K]} q_t(i)\frac{1}{\gamma}\sum_{j\in[K]}\frac{\gamma}{K}\widehat{P}_t^2(i,j)\right] \\
&\leqslant \frac{1}{\gamma}\mathbb{E}\left[\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\widehat{P}_t^2(i,j)\right].
\end{aligned}$$

The first inequality holds due to the Cauchy-Schwartz inequality; the second inequality holds because the definition of q_t satisfies $q_t \geqslant \gamma/K$.

Expanding the definition of $\widehat{P}_t^2(i,j)$, we have

$$\begin{aligned}
\widehat{P}_t^2(i,j) &= r_t^2(i_t,j_t)\left(\boldsymbol{\phi}_{i,j}^\top Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t}\right)^2 \\
&\leqslant \boldsymbol{\phi}_{i_t,j_t}^\top Q_t^{-1}\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t},
\end{aligned}$$

where we use $0 \leqslant r_t^2(i_t,j_t) \leqslant 1$. Therefore, the following inequality holds,

$$\begin{aligned}
\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\widehat{P}_t^2(i,j) &\leqslant \sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i_t,j_t}^\top Q_t^{-1}\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t} \\
&= \boldsymbol{\phi}_{i_t,j_t}^\top Q_t^{-1}\left(\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top\right)Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t} \\
&= \boldsymbol{\phi}_{i_t,j_t}^\top Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t} \\
&= \operatorname{trace}(\boldsymbol{\phi}_{i_t,j_t}\boldsymbol{\phi}_{i_t,j_t}^\top Q_t^{-1}).
\end{aligned}$$

Using the property of trace, we have

$$\mathbb{E}\left[\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\widehat{P}_t^2(i,j)\right] \leqslant \operatorname{trace}\left(\sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top Q_t^{-1}\right) = d.$$

Therefore, we finish the proof of Lemma 4.9.10. □

Proof of Theorem 4.6.1. Our regret is defined as follows,

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_T}[R_T] &= \mathbb{E}_{\mathcal{H}_T}\left[\sum_{t=1}^T [2B_t(i^*) - B_t(i_t) - B_t(j_t)]\right] \\ &= \max_{i\in[K]} \mathbb{E}_{\mathcal{H}_T}\left[\sum_{t=1}^T [2B_t(i) - B_t(i_t) - B_t(j_t)]\right].\end{aligned}$$

The second equality holds because B_t and i^* are independent of the randomness of the algorithm. Furthermore, we can write the expectation of the regret as

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_T}[R_T] &= 2\max_{i\in[K]}\sum_{t=1}^T B_t(i) - \sum_{t=1}^T \mathbb{E}_{\mathcal{H}_T}\left[B_t(i_t) + B_t(j_t)\right] \\ &= 2\max_{i\in[K]}\sum_{t=1}^T B_t(i) - 2\sum_{t=1}^T \mathbb{E}_{\mathcal{H}_{t-1}}\left[\mathbb{E}_{x\sim q_t}\left[B_t(x)|\mathcal{H}_{t-1}\right]\right] \\ &= 2\max_{i\in[K]}\sum_{t=1}^T\left(B_t(i) - \frac{1}{2}\right) - 2\mathbb{E}_{\mathcal{H}_t}\left[q_t^\top \widehat{B}_t\right], \qquad (4.9.13)\end{aligned}$$

where the last equality is due to Lemma 4.9.9.

Then we follow the standard proof of EXP3 algorithm (Lattimore and Szepesvári, 2020). Let $S_{t,k} = \sum_{s=1}^t \left(B_s(k) - \frac{1}{2}\right)$, $\widehat{S}_{t,k} = \sum_{s=1}^t \widehat{B}_s(k)$, $\omega_t = \sum_{k\in[K]}\exp(-\eta\widehat{S}_{t,k})$ and $\omega_0 = K$. We have $\forall a \in [K]$,

$$\exp(-\eta\widehat{S}_{T,a}) \leqslant \sum_{k\in[K]}\exp(-\eta\widehat{S}_{T,k}) = \omega_T = \omega_0\cdot\prod_{t=1}^T\frac{\omega_t}{\omega_{t-1}}. \qquad (4.9.14)$$

For each term in the product, we have

$$\begin{aligned}\frac{\omega_t}{\omega_{t-1}} &= \sum_{k\in[K]}\frac{\exp(-\eta\widehat{S}_{t-1,k})}{\omega_{t-1}}\cdot\exp(-\eta\widehat{B}_t(k)) \\ &= \sum_{k\in[K]}\widetilde{q}_t(k)\exp(-\eta\widehat{B}_t(k)), \qquad (4.9.15)\end{aligned}$$

where the second equality holds because of the definition of $\widetilde{q}_t$. For any $\eta \leqslant \lambda_0\gamma^2$, Lemma 4.9.7 presents $|\eta\widehat{B}_t(k)| \leqslant 1$. Thus, using the basic inequality $\exp(x) \leqslant 1 + x + x^2/2$ when $x \leqslant 1$, and $\exp(x) \geqslant 1 + x$, we have

$$\begin{aligned}
\frac{\omega_t}{\omega_{t-1}} &\leqslant \sum_{k\in[K]} \widetilde{q}_t(k)\left(1 - \eta\widehat{B}_t(k) + \eta^2\widehat{B}_t^2(k)\right) \\
&= 1 - \eta\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t(k) + \eta^2\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t^2(k) \\
&\leqslant \exp\left(-\eta\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t(k) + \eta^2\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t^2(k)\right). \qquad (4.9.16)
\end{aligned}$$

Combining (4.9.14), (4.9.15) and (4.9.16), we have

$$\exp(-\eta\widehat{S}_{T,a}) \leqslant K\exp\left(\sum_{t=1}^{T}\left[-\eta\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t(k) + \eta^2\sum_{k\in[K]} \widetilde{q}_t(k)\widehat{B}_t^2(k)\right]\right),$$

and therefore

$$\sum_{t=1}^{T}\widehat{B}_t(a) - \sum_{t=1}^{T}\widetilde{q}_t^\top\widehat{B}_t \leqslant \frac{\log K}{\eta} + \eta\sum_{t=1}^{T}\sum_{k\in[K]}\widetilde{q}_t(k)\widehat{B}_t^2(k).$$

Since $\widetilde{q}_t = \frac{q_t - \gamma/K}{1-\gamma}$, we have

$$(1-\gamma)\sum_{t=1}^{T}\widehat{B}_t(a) - \sum_{t=1}^{T}q_t^\top\widehat{B}_t \leqslant \frac{\log K}{\eta} + \eta\sum_{t=1}^{T}\sum_{k\in[K]}\widetilde{q}_t(k)\widehat{B}_t^2(k).$$

Choosing $a = i^*$, changing the summation index to i and taking expectation on both sides, we have

$$(1-\gamma)\mathbb{E}_{\mathcal{H}_T}\sum_{t=1}^{T}\widehat{B}_t(i^*) - \sum_{t=1}^{T}\mathbb{E}_{\mathcal{H}_T}\left[q_t^\top\widehat{B}_t\right] \leqslant \frac{\log K}{\eta} + \mathbb{E}_{\mathcal{H}_T}\left[\eta\sum_{t=1}^{T}\sum_{i\in[K]}q_t(i)\widehat{B}_t^2(i)\right].$$

Substituting the above inequality into (4.9.13) and using Lemma 4.9.8, 4.9.9, we can bound the regret as

$$\begin{aligned}
\mathbb{E}[R_T] &\leqslant 2\gamma T + \frac{2\log K}{\eta} + 2\eta\sum_{t=1}^{T}\mathbb{E}_{\mathcal{H}_T}\left[\sum_{i\in[K]}q_t(i)s_t(i)^2\right] \\
&\leqslant 2\gamma T + 2\frac{\log K}{\eta} + \frac{2\eta dT}{\gamma} \\
&\leqslant 2(\log K)^{1/3}d^{1/3}T^{2/3}\sqrt{1/\lambda_0} + 2(\log K)^{1/3}d^{1/3}T^{2/3} + 2(\log K)^{1/3}d^{1/3}T^{2/3}\sqrt{\lambda_0},
\end{aligned}$$

where the second inequality holds due to Lemma 4.9.10. In the last inequality, we put in our choice of parameters $\eta = (\log K)^{2/3} d^{-1/3} T^{-2/3}$ and $\gamma = \sqrt{\eta d / \lambda_0} = (\log K)^{1/3} d^{1/3} T^{-1/3} \lambda_0^{-1/2}$. This finishes our proof of Theorem 4.6.1. □

4.10 Detailed Explanation of Algorithms in Experiments

The following list summarizes all the methods we implemented:

BETC-GLM(-Match): Algorithm 16 proposed in this chapter. For general link function, to find $\widehat{\boldsymbol{\theta}}$ by MLE in (4.5.1), 100 rounds of gradient descent are performed. The failure probability is set to $\delta = 1/T$. Parameters τ and ϵ are set to values listed in Theorem 4.5.5. For BETC-GLM-Match, we use the τ and ϵ outlined in Theorem 4.5.1.

UCB-Borda: The UCB algorithm (Auer et al., 2002) using *Borda reduction* technique mentioned by Busa-Fekete et al. (2018). The complete listing is displayed in Algorithm 18.

DEXP3: Dueling-Exp3 is an adversarial Borda bandit algorithm developed by Saha et al. (2021a), which also applies to our stationary bandit case. Relevant tuning parameters are set according to their upper-bound proof.

ETC-Borda: We devise a simple explore-then-commit algorithm, named ETC-Borda. Like DEXP3, ETC-Borda does not take any contextual information into account. The complete procedure of ETC-Borda is displayed in Algorithm 19, Section 4.11.3. The failure probability δ is optimized as $1/T$.

BEXP3: The proposed method for adversarial Borda bandits displayed in Algorithm 17. η and γ are chosen to be the value stated in Theorem 4.6.1.

4.11 Additional Information for Experiments

4.11.1 Data Visualization

The events in the EventTime dataset are ordered by the time they occurred. In Figure 4.3, the magnitude of each $\widetilde{p}_{i,j}$ is color coded. It is apparent that there is no total/consistent ordering (i.e., $\widetilde{p}_{i,j} > \frac{1}{2} \Leftrightarrow i \succ j$) can be inferred from this matrix due to inconsistencies in the ordering and many potential paradoxes. Hence STI and SST can hardly hold in this case.

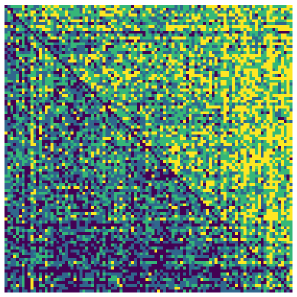


Figure 4.3: Estimated preferential matrix consists of $\widetilde{p}_{i,j}$ from the EventTime dataset.

4.11.2 The UCB-Borda Algorithm

The UCB-Borda procedure, displayed in Algorithm 18 is a UCB algorithm with Borda reduction only capable of minimization of regret in the following form:

$$\text{Regret}(T) = \sum_{t=1}^{T} \big(B(i^*) - B(i_t)\big).$$

Let $\mathbf{n}_i$ be the number of times arm $i \in [K]$ has been queried. Let $\mathbf{w}_i$ be the number of times arm i wins the duel. $\widehat{B}(i)$ is the estimated Borda score. α is set to 0.3 in all experiments.

Algorithm 18 UCB-Borda

```
1: Input: time horizon $T$, number of items $K$, exploration parameter $\alpha$.
2: Initialize:  $\mathbf{n} = \mathbf{w} = \{0\}^K$, $\widehat{B}(i) = \frac{1}{2}, i \in [K]$
3: for $t = 1, 2, \ldots, T$ do
4:     $i_t = \mathrm{ar}gmax_{k \in [K]}\big(\widehat{B}_k + \sqrt{\frac{\alpha \log(t)}{\mathbf{n}_k}}\big)$
5:     sample $j_t \sim \text{Uniform}([K])$
6:     query pair $(i_t, j_t)$ and receive feedback $r_t \sim \text{Bernoulli}(p_{i_t, j_t})$
7:     $\mathbf{n}_{i_t} = \mathbf{n}_{i_t} + 1$, $\mathbf{w}_{i_t} = \mathbf{w}_{i_t} + r_t$, $\widehat{B}(i_t) = \frac{\mathbf{w}_{i_t}}{\mathbf{n}_{i_t}}$
8: end for
```

4.11.3 The ETC-Borda Algorithm

The ETC-Borda procedure, displayed in Algorithm 19 is an explore-then-commit type algorithm capable of minimizing the Borda dueling regret. It can be shown that the regret of Algorithm 19 is $\widetilde{O}(K^{1/3}T^{2/3})$.

Algorithm 19 ETC-Borda

```
1: **Input:** time horizon $T$, number of items $K$, target failure probability $\delta$
2: **Initialize:** $\mathbf{n} = \mathbf{w} = \{0\}^K$, $\widehat{B}(i) = \frac{1}{2}, i \in [K]$
3: Set $N = \lceil K^{-2/3}T^{2/3}\log(K/\delta)^{1/3}\rceil$
4: **for** $t = 1, 2, \ldots, T$ **do**
5:     Choose action $i_t \leftarrow \begin{cases} 1 + (t-1) \bmod K, & \text{if } t \leqslant KN, \\ argmax_{i\in[K]}\widehat{B}(i), & \text{if } t > KN. \end{cases}$
6:     Choose action $j_t = \begin{cases} \text{Uniform}([K]), & \text{if } t \leqslant KN, \\ argmax_{i\in[K]}\widehat{B}(i), & \text{if } t > KN. \end{cases}$
7:     query pair $(i_t, j_t)$ and receive feedback $r_t \sim \text{Bernoulli}(p_{i_t,j_t})$
8:     **if** $t \leqslant N$ **then**
9:         $\mathbf{n}_{i_t} = \mathbf{n}_{i_t} + 1$, $\mathbf{w}_{i_t} = \mathbf{w}_{i_t} + r_t$, $\widehat{B}(i_t) = \frac{\mathbf{w}_{i_t}}{\mathbf{n}_{i_t}}$
10:    **end if**
11: **end for**
```

4.11.4 The Frank-Wolfe Algorithm for Approximate G-optimal Design

In order to find a solution for the G-optimal design problem, we resort to the Frank-Wolfe algorithm to find an approximate solution. The detailed procedure is listed in Algorithm 20. In Line 4, each outer product costs d^2 multiplications, K^2 such matrices are scaled and summed into a d-by-d matrix $\mathbf{V}(\pi)$, which costs $O(K^2d^2)$ operations in total. In Line 5, one matrix inversion costs approximately $O(d^3)$. The weighted norm requires $O(d^2)$ and the maximum is taken over K^2 such calculated values. The scaling and update in the following lines only require $O(K^2)$. In summary, the algorithm is dominated by the calculation in Line 5 which costs $O(d^2K^2)$.

In experiments, the G-optimal design $\pi(i, j)$ is approximated by running 20 iterations of Frank-Wolfe algorithm, which is more than enough for its convergence given our particular problem instance. (See Note 21.2 in (Lattimore and Szepesvári, 2020)).

Algorithm 20 G-OPTIMAL DESIGN BY FRANK-WOLFE

```
1: Input: number of items $K$, contextual vectors $\boldsymbol{\phi}_{i,j}, i \in [K], j \in [K]$, number of iterations $R$
2: Initialize:  $\pi_1(i, j) = 1/K^2$
3: for $r = 1, 2, \cdots, R$ do
4:     $\mathbf{V}(\pi_r) = \sum_{i,j} \pi_r(i, j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$
5:     $i_r^*, j_r^* = \mathrm{a}rgmax_{(i,j)\in[K]\times[K]}||\boldsymbol{\phi}_{i,j}||_{\mathbf{V}(\pi_r)^{-1}}$
6:     $g_r = ||\boldsymbol{\phi}_{i_r^*, j_r^*}||_{\mathbf{V}(\pi_r)^{-1}}$
7:     $\gamma_r = \frac{g_r - 1/d}{g_r - 1}$
8:     $\pi_{r+1}(i, j) = (1 - \gamma_r)\pi_r(i, j) + \gamma_r \mathbb{1}(i_r^* = i) \mathbb{1}(j_r^* = j)$
9: end for
10: Output: Approximate G-optimal design solution $\pi_{R+1}(i, j)$
```

4.12 BEXP3 with Many Arms

In this section, we discuss how our algorithm BEXP3 can be modified to obtain a $\widetilde{O}(d^{2/3}T^{2/3})$ regret, when the number of arms K is exponentially large ($\log K = \omega(d)$).

The idea is to first construct an ϵ-cover $\mathcal{S}$ over the set of mean vectors $\big\{\frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}\big\}_{i\in[K]}$, so that for any $i \in [K]$, there exist a vector $\boldsymbol{\psi}_i \in \mathcal{S}$ such that $\big\|\frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j} - \boldsymbol{\psi}_i\big\|_2 \leqslant \epsilon$. In the d-dimensional space, it suffices to choose $|\mathcal{S}| = O((1/\epsilon)^d)$ representative vectors to cover all the vectors.

Algorithm 21 BEXP3 with large K

```
1: Input: time horizon $T$, number of items $K$, feature dimension $d$, feature vectors $\boldsymbol{\phi}_{i,j}$ for
   $i \in [K]$, $j \in [K]$, learning rate $\eta$, exploration parameter $\gamma$, covering radius $\epsilon$.
2: Initialize: $q_1(i) = \frac{1}{K}$.
3: Calculate an $\epsilon$-cover $\mathcal{S}$ of $\big\{\frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}\big\}_{i\in[K]}$, with $|\mathcal{S}| = O((1/\epsilon)^d)$.
4: Denote the vector in $\mathcal{S}$ closest to $\frac{1}{K}\sum_{j\in[K]} \boldsymbol{\phi}_{i,j}$ as $\boldsymbol{\psi}_i$ for all $i \in [K]$.
5: for $t = 1, \ldots, T$ do
6:     Sample items $i_t \sim q_t$, $j_t \sim q_t$.
7:     Query pair $(i_t, j_t)$ and receive feedback $r_t$ .
8:     Calculate $Q_t = \sum_{i\in[K]}\sum_{j\in[K]} q_t(i)q_t(j)\boldsymbol{\phi}_{i,j}\boldsymbol{\phi}_{i,j}^\top$, $\widehat{\boldsymbol{\theta}}_t = Q_t^{-1}\boldsymbol{\phi}_{i_t,j_t} r_t$.
9:     Calculate the (shifted) Borda score estimates $\widehat{B}_t(i) = \langle \boldsymbol{\psi}_i, \widehat{\boldsymbol{\theta}}_t \rangle$.
10:    Update for all $i \in [K]$, set
       $$\widetilde{q}_{t+1}(i) = \frac{\exp(\eta\sum_{l=1}^t \widehat{B}_l(i))}{\sum_{j\in[K]}\exp(\eta\sum_{l=1}^t \widehat{B}_l(j))}; \qquad q_{t+1}(i) = (1-\gamma)\widetilde{q}_{t+1}(i) + \frac{\gamma}{K}.$$
11: end for
```

Now BEXP3 will be performed as the original case except that at Line 9 in Algorithm 21, we replace the average contextual vectors $\frac{1}{K}\sum_{j=1}^K \boldsymbol{\phi}_{i,j}$ with those nearest representative vectors $\boldsymbol{\phi}_i$. Since there are only $|\mathcal{S}| = O((1/\epsilon)^d)$ unique rewards, the EXP3 argument (4.9.14) can be performed on $|\mathcal{S}|$ unique arms instead of K arms:

$$\exp(-\eta \widehat{S}_{T,a}) \leqslant \sum_{\phi \in \mathcal{S}} \exp(-\eta \widehat{S}_{T,\phi}) = \omega_T = \omega_0 \cdot \prod_{t=1}^{T} \frac{\omega_t}{\omega_{t-1}},$$

where $\omega_0 = |\mathcal{S}|$ instead of K.

Eventually the algorithm suffers a regret of $\widetilde{O}(d^{2/3}T^{2/3}\log^{1/3}(1/\epsilon))$. The ϵ-net incurs an additional approximation error of order $O(\epsilon T)$. Setting $\epsilon = T^{-1}$ will improve the regret to $d^{2/3}T^{2/3}$ up to log factors.

4.13 Additional Comparison between BETC and BEXP3

In Figure 4.4 we show that under the same experimental setting, tuning the error tolerance ϵ in BETC can further reduce its total regret up to a constant factor, showing that under suitable hyper-parameter choices, BETC can outperform BEXP3.

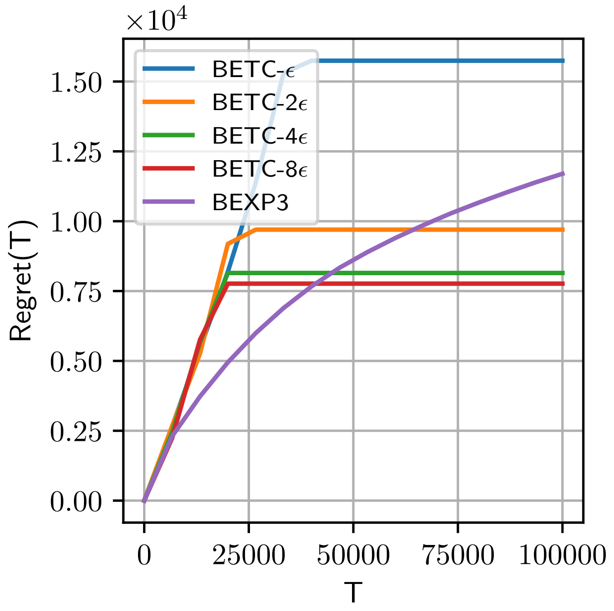


Figure 4.4: The performance of BETC under different choices of error tolerance ϵ, compared with BEXP3. We examined BETC with ϵ, 2ϵ, 4ϵ, 8ϵ where $\epsilon = d^{1/6}T^{-1/3}$.

CHAPTER 5

Self-Play Preference Optimization for Language Model Alignment

5.1 Introduction

Large Language Models (LLMs) (e.g., Ouyang et al., 2022; OpenAI et al., 2023), have shown remarkable capabilities in producing human-like text, fielding questions, and coding. Despite their advancements, these models encounter challenges in tasks requiring high levels of reliability, safety, and ethical alignment. To address these challenges, Reinforcement Learning from Human Feedback (RLHF), also known as Preference-based Reinforcement Learning (PbRL), presents a promising solution. This framework for policy optimization, highlighted in works by Christiano et al. (2017) and recently in Ouyang et al. (2022), has led to significant empirical success in fine-tuning instruction-following LLMs, making them more aligned with human preferences and thus more helpful.

Most existing approaches to RLHF rely on either explicit or implicit reward models. Taking InstructGPT (Ouyang et al., 2022) as an example, a reference policy π_{ref} is first established, typically from supervised pre-training or instruction-based (supervised) fine-tuning. An explicit reward function is obtained by training a reward model based on human preference feedback data, employing the Bradley-Terry (BT) model (Bradley and Terry, 1952). Subsequently, reinforcement learning algorithms such as Proximal Policy Optimization (Schulman et al., 2017, PPO) are used to fine-tune the reference LLM π_{ref} by maximizing the expected reward function. The reward model provides a "reward score" $r(\mathbf{y}; \mathbf{x})$ for the given response $\mathbf{y}$ and prompt $\mathbf{x}$, approximately reflecting how humans value these responses. More recently, methods like Direct Preference Optimization (Rafailov et al., 2024, DPO) have been introduced. These methods forgo the training of a separate reward model, instead using the log-likelihood ratio to implicitly represent the reward score, which is then integrated into the same Bradley-Terry model to directly optimize the LLM. Nonetheless, both the two-step RLHF algorithms and the one-step direct preference fundamentally adhere to the reward maximization objective and are determined by parametric models such as the BT model.

Parametric preference models such as the Bradley-Terry model (Bradley and Terry, 1952) and the Thurstone model (Thurstone, 1927a) provide reasonable approximations of human preferences, yet they fall short of fully capturing the complexity of human behavior. These models presuppose a monotonous and transitive relationship among preferences for different choices. However, empirical evidence suggests otherwise. For instance, Tversky (1969) observed human decisions can be influenced by different factors and exhibit inconsistency. Such observations indicate that human preferences do not always adhere to a single, value-based hierarchy and can even appear irrational, such as exhibiting loops in preference relations. For LLMs, another motivating evidence is that Munos et al. (2023) has empirically shown that directly predicting the pairwise preference can achieve higher accuracy than predicting the preference via a BT-based reward model.

To address the inconsistency in human preference, researchers have proposed to work directly with the preference probability and design algorithms that can more flexibly represent human preferences (Lou et al., 2022; Wu et al., 2023) in the ranking or bandit setting. Recently, an emerging line of work (Wang et al., 2024; Munos et al., 2023; Swamy et al., 2024) also proposed to study RLHF for LLMs under such general preference $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})$, where $\mathbf{y}$ and $\mathbf{y}'$ are two different responses and $\mathbf{x}$ is prompt. The goal is to identify the Nash equilibrium or von Neumann winner of the two-player constant-sum game

$$(\pi^*, \pi^*) = \arg\max_{\pi} \min_{\pi'} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\Big[\mathbb{E}_{\mathbf{y}\sim\pi(\cdot|\mathbf{x}),\mathbf{y}'\sim\pi'(\cdot|\mathbf{x})}\big[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})\big]\Big],$$

where each player is an LLM that outputs responses and aims to maximize its probability of being preferred over its opponent. The preference $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})$ is assumed given by an external source, such as human annotators or a strong language model. Munos et al. (2023) studied the KL-regularized version of this game and proposed to approximate the Nash equilibrium using an on-policy mirror descent algorithm.

Very recently, Swamy et al. (2024) proposed Self-play Preference Optimization (SPO)[1] for the same (unregularized) two-player constant-sum game. They provide a general reduction of preference optimization to no-regret online learning for the multi-step Markov Decision Process. When constrained to the bandit setting for LLMs, their proposed algorithmic framework reduces to the famous Hedge algorithm (Freund and Schapire, 1997), which admits the exponential update rule as described in (5.4.1). To approximately solve the exponential update, Swamy et al. (2024) then proposed to employ typical policy optimization algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) or Soft Actor-Critic (SAC) (Haarnoja et al., 2018) to maximize the win rate against the reference policy and evaluated the performance of their self-play algorithms in robotic and game tasks. However, it typically requires more effort to apply PPO or SAC to large-scale fine-tuning of LLM and make them work stably. Therefore, it remains unclear how their self-play framework can be applied to large-scale language model alignment efficiently.

In this chapter, motivated by these developments mentioned above, we propose a new self-play algorithm that (1) enjoys provable guarantees to solve the two-player constant-sum game; and (2) can scale up to large-scale efficient fine-tuning of large language models. In detail, we formulate the RLHF problem as a constant-sum two-player game. Our objective

[1]The SPO framework does not pertain to the efficient fine-tuning of LLMs. Our Self-Play Preference Optimization (SPPO) focuses on LLM alignment and was developed independently. To distinguish it from the SPO framework, we use the abbreviation SPPO.

is to identify the Nash equilibrium policy, which consistently provides preferred responses over any other policy on average. To identify the Nash equilibrium policy approximately, we adopt the classic online adaptive algorithm with multiplicative weights (Freund and Schapire, 1999) as a high-level framework that solves the two-player game. Further, each step of the high-level framework can be approximated by a *self-play* mechanism, where in each round the policy is playing against itself in the previous round by fine-tuning it on synthetic data that are generated by the policy and annotated by the preference model.

Our contributions are highlighted as follows:

- Starting from the exponential weight update algorithm which provably converges to the Nash equilibrium of the two-player constant-sum game, we propose the *Self-Play Preference Optimization* (SPPO) algorithm for large language model alignment. The algorithm converges to an approximate Nash equilibrium provably and admits a simple form of loss function for easy optimization.

- We compare our method with the state-of-the-art method including DPO, Identity Preference Optimization (IPO) (Azar et al., 2023), and Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024), and show that our SPPO loss function can effectively increase the log-likelihood of the chosen response and decrease that of the rejected response, which cannot be trivially achieved by symmetric pairwise loss such as DPO and IPO. Our experiments also confirm that SPPO outperforms iterative DPO and IPO on various benchmarks.

- Empirically, SPPO significantly enhances the well-aligned Mistral-7B-Instruct-v0.2 and Llama-3-8B-Instruct model, achieving an increase of over 11% on the length-controlled win rate against GPT-4-Turbo on the AlpacaEval 2.0 (Dubois et al., 2024a) test set. Additionally, SPPO exhibits strong generalist abilities across different tasks, including MT-Bench, the Open LLM Leaderboard, and the PairRM score (Jiang et al., 2023b). Unlike iterative DPO/IPO, which tends to show performance decay on other benchmarks when optimized towards the PairRM score, SPPO's performance gain is consistent. Notably, all the strong performances are achieved without external supervision (e.g., responses, preferences, etc.) from GPT-4 or other stronger language models. Despite using only the 60k prompts (without responses) from the UltraFeedback dataset (Cui et al., 2023) and forgoing any prompt augmentation, our method achieves performance comparable to GPT-4 on the AlpacaEval 2.0 win-rate.

In this chapter, motivated by these developments mentioned above, we propose a new self-play algorithm that (1) enjoys provable guarantees to solve the two-player constant-sum game; and (2) can scale up to large-scale efficient fine-tuning of large language models. In detail, we formulate the RLHF problem as a constant-sum two-player game. Our objective is to identify the Nash equilibrium policy, which consistently provides preferred responses over any other policy on average. To identify the Nash equilibrium policy approximately, we adopt the classic online adaptive algorithm with multiplicative weights (Freund and Schapire, 1999) as a high-level framework that solves the two-player game. Further, each step of the high-level framework can be approximated by a *self-play* mechanism, where in each round the policy is playing against itself in the previous round by fine-tuning it on synthetic data that are generated by the policy and annotated by the preference model.

5.1.1 Organization of this Chapter

We review relevant work in the literature in Section 5.2. We present the backgrounds and preliminaries in Section 5.3. Starting from the exponential weight update algorithm which provably converges to the Nash equilibrium of the two-player constant-sum game, we propose the *Self-Play Preference Optimization* (SPPO) algorithm for large language model alignment in Section 5.4. In Section 5.5, we show empirically that SPPO significantly enhances the performance across various benchmarks, without external supervision (e.g., responses, preferences, etc.) from GPT-4 or other stronger language models. We conclude the chapter with

Section 5.6. We defer the detailed proof of the theorems to Section 5.7.

5.2 Related Work

RLHF with Explicit/Implicit Reward Model Originally, reinforcement learning from human feedback (RLHF) was proposed by Christiano et al. (2017) as a methodology that first learns a reward model reflecting human preferences and then uses reinforcement learning algorithms to maximize the reward. This methodology is applied by Ouyang et al. (2022) to fine-tune instruction-following large language models and leads to the popular ChatGPT.

The reward model in the works mentioned above assumes a parametric model such as the Bradley-Terry model (Bradley and Terry, 1952), which assigns a "score" representing how preferred a given response is. More recently, Rafailov et al. (2024) proposed to instead directly solve the closed-form solution of such a score implied by the Bradley-Terry model. The Direct Policy Optimization (DPO) method is claimed to be more efficient and stable, yet, still implicitly assumes such a reward model that specifies the "score". In a similar spirit, Zhao et al. (2023) proposed to calibrate the score so that the score of the winner in comparison has a margin over the score of the loser, and induces a different SLic loss. Similarly, Ethayarajh et al. (2024) derived a different loss function (called KTO) from the Kahneman-Tversky human utility function, which implicitly denotes a score of the given response. Liu et al. (2023) proposed Rejection Sampling Optimization (RSO) which utilizes a preference model to generate preference pairs with candidates sampled from the optimal policy; then preference optimization is applied on the sampled preference pairs. Hong et al. (2024) proposed Odds Ratio Preference Optimization (ORPO) algorithm that can perform supervised fine-tuning and preference alignment in one training session without maintaining an intermediate reference policy.

RLHF with General Preference Model Often, the human preference is not strictly transitive, and cannot be sufficiently represented by a single numerical score. Azar et al. (2023) proposed a general preference optimization objective based on the preference probability between a pair of responses instead of a score of a single response. They further propose a learning objective based on identity mapping of the preference probability called IPO (Preference Optimization with Identity mapping), which aims to maximize the current policy's expected winning probability over a given reference policy. Munos et al. (2023) formulated the RLHF problem with general preference as a two-player, constant-sum game, where each player is one policy that aims to maximize the probability of its response being preferred against its opponent. They aim to identify the Nash equilibrium policy of this game and propose a mirror-descent algorithm that guarantees the last-iterate convergence of a policy with tabular representations[2]. Wang et al. (2024) proposed to identify the Nash equilibrium policy for multi-step MDPs when a general preference model is present and shows that the problem can be reduced to a two-player zero-sum Markov game.

Theory of RLHF There is also a line of research to analyze RLHF and provide its theoretical guarantees. Zhu et al. (2023) studied the standard RLHF with separate reward-learning and model-tuning and proposed a pessimistic reward-learning process that provably learns a linear reward model. Wang et al. (2024) proposed a framework to reduce any RLHF problem with a reward model to a reward-based standard RL problem. Additionally, they proposed to identify the Nash equilibrium policy when a general preference model is present and show that the problem can be reduced to a two-player zero-sum Markov game. Xiong et al. (2023) studied the reverse-KL regularized contextual bandit for RLHF in different settings and proposed efficient algorithms with finite-sample theoretical guarantees. Ye et al. (2024) studied the theoretical learnability of the KL-regularized Nash-Learning from Human Feedback (NLHF) by considering both offline and online settings and proposed provably efficient

[2]Due to the tabular representation, computing the normalizing factor is prohibitive and the algorithm is approximately executed by sampling one token instead of a full response.

algorithms. Ji et al. (2024) proposed an active-query-based proximal policy optimization algorithm with regret bounds and query complexity based on the problem dimension and the sub-optimality gap.

Self-Play Fine-Tuning Most works mentioned above (Rafailov et al., 2024; Zhao et al., 2023; Azar et al., 2023; Ethayarajh et al., 2024) consider one single optimization procedure starting from some reference policy. The same procedure may be applied repeatedly for multiple rounds in a self-play manner. In each round, new data are generated by the policy obtained in the last round; these new data are then used for training a new policy that can outperform the old policy.

The self-play fine-tuning can be applied to both scenarios with or without human preference data. For example, Singh et al. (2023) proposed an Expectation-Maximization (EM) framework where in each round, new data are generated and annotated with a reward score; the new policy is obtained by fine-tuning the policy on the data with a high reward. Chen et al. (2024) proposed a self-play framework to fine-tune the model in a supervised way. In each round, new preference pairs are synthesized by labeling the policy-generated responses as losers and the human-generated responses as winners. Then DPO is applied in each round to fine-tune another policy based on these synthesized preference data. Yuan et al. (2024) proposed Self-Rewarding Language Models, where the language model itself is used to annotate preference on its own responses. Iterative DPO is applied to fine-tune language models on these annotated data. These works show iterative fine-tuning can significantly improve the performance.

Swamy et al. (2024) considered a more general multi-step Markov Decision Process (MDP) setting and proposed Self-play Preference Optimization (SPO), an RLHF framework that can utilize any no-regret online learning algorithm for preference-based policy optimization. They then instantiated their framework with Soft Policy Iteration as an idealized variant of their algorithm, which reduces to the exponential weight update rule (5.4.1) when constrained to the bandit setting. The main difference is that they focus on the multi-round Markov decision process (MDP) in robotic and game tasks rather than on fine-tuning large language models and approximating the update using policy optimization methods such as PPO.

Concurrent to our work, Rosset et al. (2024) proposed the Direct Nash Optimization (DNO) algorithm based on the cross-entropy between the true and predicted win rate gaps, and provided theoretical guarantees on the error of finite-sample approximation. However, their practical version still utilizes the iterative-DPO framework as in Xu et al. (2023) with the DPO loss instead of their derived DNO loss. Notably, in their experiments, they added the GPT-4 generated responses as their "gold sample" into their fine-tuning data, and used GPT-4 as a judge to assign a numerical score to each response for preference pair construction. In sharp contrast, our work does not require the use of any strong external supervision besides a small-sized reward model. Another concurrent work (Gao et al., 2024) proposed REBEL, an iterative self-play framework via regressing the relative reward. When applied to the preference setting, it results a similar algorithm to our algorithm SPPO, except that SPPO approximates the log-partition factor $\log Z_{\pi_t}(\mathbf{x})$ with $\eta/2$ while REBEL regresses on the win rate difference (so that $\log Z_{\pi_t}(\mathbf{x})$ is canceled). Additionally, Calandriello et al. (2024) pointed out that optimizing the IPO loss (Azar et al., 2023) iteratively with self-play generated data is equivalent to finding the Nash equilibrium of the two-player game, and they proposed the IPO-MD algorithm based on this observation, which generates data with a mixture policy similar to the Nash-MD algorithm.

5.3 Preliminaries

We consider the preference learning scenario as follows. Given a text sequence (commonly referred to as prompt) $\mathbf{x} = [x_1, x_2, \ldots]$, two text sequences $\mathbf{y} = [y_1, y_2, \ldots]$ and $\mathbf{y}'$ are generated as responses to the prompt $\mathbf{x}$. An autoregressive language model π given the prompt $\mathbf{x}$ can generate responses $\mathbf{y}$ following the probability decomposition

$$\pi(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} \pi(y_i|\mathbf{x}, \mathbf{y}_{<i}).$$

Given the prompt $\mathbf{x}$ and two responses $\mathbf{y}$ and $\mathbf{y}'$, a preference oracle (either a human annotator or a language model) will provide preference feedback $o(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) \in \{0, 1\}$ indicating whether $\mathbf{y}$ is preferred over $\mathbf{y}'$. We denote $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) = \mathbb{E}[o(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})]$ as the probability of $\mathbf{y}$ "winning the duel" over $\mathbf{y}'$. The KL divergence of two probability distributions of density p and q is defined as $\mathrm{KL}(p\|q) = \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y})}\Big[\log \frac{p(\mathbf{y})}{q(\mathbf{y})}\Big]$.

5.3.1 RLHF with Reward Models

Christiano et al. (2017) first learn a reward function $r(\mathbf{y}; \mathbf{x})$ following the Bradley-Terry model (Bradley and Terry, 1952). For a prompt-response-response triplet $(\mathbf{x}, \mathbf{y}, \mathbf{y}')$, the Bradley-Terry model specifies the probability of $\mathbf{y}$ being chosen over $\mathbf{y}$ as

$$\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) = \frac{\exp(r(\mathbf{y}; \mathbf{x}))}{\exp(r(\mathbf{y}; \mathbf{x})) + \exp(r(\mathbf{y}'; \mathbf{x}))} = \sigma\big(r(\mathbf{y}; \mathbf{x}) - r(\mathbf{y}'; \mathbf{x})\big), \tag{5.3.1}$$

where $\sigma(x) = e^x/(e^x + 1)$ is the logistic function. The reward function associated with the Bradley-Terry model can be estimated by maximizing the log-likelihood $\log \mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})$. Suppose the true reward function $r(\mathbf{y}; \mathbf{x}))$ is available, Christiano et al. (2017) proposed to solve the following optimization problem with policy optimization algorithms in RL such as PPO (Schulman et al., 2017):

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}, \mathbf{y}\sim\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}[r(\mathbf{y}; \mathbf{x})] - \eta^{-1}\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\mathrm{KL}(\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})\|\pi_{\mathrm{ref}}(\cdot|\mathbf{x}))], \tag{5.3.2}$$

where $\mathcal{X}$ is the prompt distribution.

Rafailov et al. (2024) identified that the optimization problem above has a closed-form solution such that for any $\mathbf{y}$,

$$\pi^*(\mathbf{y}|\mathbf{x}) \propto \pi_{\mathrm{ref}}(\mathbf{y}|\mathbf{x}) \exp(\eta r(\mathbf{y}; \mathbf{x})),$$

which can be further converted to the DPO loss for any triplet $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$ where the winner $\mathbf{y}_w$ is chosen over the loser $\mathbf{y}_l$:

$$\ell_{\mathrm{DPO}}(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l; \boldsymbol{\theta}; \pi_{\mathrm{ref}}) := -\log \sigma\left(\eta^{-1}\left[\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_w|\mathbf{x})}\right) - \log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_l|\mathbf{x})}\right)\right]\right).$$

5.3.2 RLHF with General Preference

Following Wang et al. (2024); Munos et al. (2023), we aim to establish RLHF methods without a reward model, as the human preference can be non-transitive (Tversky, 1969). Under a general preference oracle $\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})$, we follow Dudík et al. (2015) and aim to identify the *von Neumann winner*. More specifically, the von Neumann winner π^* is the (symmetric) Nash equilibrium of the following two-player constant-sum game:

$$(\pi^*, \pi^*) = \arg\max_{\pi} \min_{\pi'} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\Big[\mathbb{E}_{\mathbf{y}\sim\pi(\cdot|\mathbf{x}), \mathbf{y}'\sim\pi'(\cdot|\mathbf{x})}\big[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})\big]\Big]. \tag{5.3.3}$$

In addition, we define the winning probability of one response $\mathbf{y}$ against a distribution of responses π as

$$\mathbb{P}(\mathbf{y} \succ \pi|\mathbf{x}) = \mathbb{E}_{\mathbf{y}'\sim\pi(\cdot|\mathbf{x})}[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})],$$

and the winning probability of one policy π against another policy π' as

$$\mathbb{P}(\pi \succ \pi'|\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi(\cdot|\mathbf{x})}\mathbb{E}_{\mathbf{y}'\sim\pi'(\cdot|\mathbf{x})}[\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x})].$$

Furthermore, we define $\mathbb{P}(\pi \succ \pi') = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}[\mathbb{P}(\pi \succ \pi'|\mathbf{x})]$, where $\mathbf{x}$ is a prompt drawn from the prompt distribution $\mathcal{X}$. The two-player constant-sum game (5.3.3) can be simplified as

$$(\pi^*, \pi^*) = \arg\max_{\pi} \min_{\pi'} \mathbb{P}(\pi \succ \pi').$$

5.4 Self-Play Preference Optimization (SPPO)

In this section, we introduce the Self-Play Preference Optimization (SPPO) algorithm, derived from the following theoretical framework.

5.4.1 Theoretical Framework

There are well-known algorithms to approximately solve the Nash equilibrium in a constant-sum two-player game. In this work, we follow Freund and Schapire (1999) to establish an iterative framework that can asymptotically converge to the optimal policy on average. We start with a theoretical framework that conceptually solves the two-player game as follows:

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) \propto \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})), \text{ for } t = 1, 2, \ldots. \tag{5.4.1}$$

(5.4.1) is an iterative framework that relies on the multiplicative weight update in each round t and enjoys a clear structure. Initially, we have a base policy π_1 usually from some supervised fine-tuned model. In each round, the updated policy π_{t+1} is obtained from the reference policy π_t following the multiplicative weight update. More specifically, a response $\mathbf{y}$ should have a higher probability weight if it has a higher average advantage over the current policy π_t.

Equivalently, (5.4.1) can be written as

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) = \frac{\pi_t(\mathbf{y}|\mathbf{x}) \exp\big(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\big)}{Z_{\pi_t}(\mathbf{x})}, \tag{5.4.2}$$

where $Z_{\pi_t}(\mathbf{x}) = \sum_{\mathbf{y}} \pi_t(\mathbf{y}|\mathbf{x}) \exp\big(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})\big)$ is the normalizing factor (a.k.a., the partition function). For any fixed $\mathbf{x}$ and $\mathbf{y}$, the ideal update policy π_{t+1} should satisfy the following equation:

$$\log\left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) = \eta \cdot \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x}). \tag{5.4.3}$$

Unlike the pair-wise design in DPO or IPO that cancels the log normalizing factor $\log Z_{\pi_t}(\mathbf{x})$ by differentiating (5.4.3) between $\mathbf{y}$ and $\mathbf{y}'$, we choose to approximate (5.4.3) directly in terms of L_2 distance:

$$\pi_{t+1} = \mathrm{a}rgmin_{\pi} \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})}\Bigg(\log\left(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right) - \Big(\eta\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \log Z_{\pi_t}(\mathbf{x})\Big)\Bigg)^2. \tag{5.4.4}$$

Estimation of the Probability The optimization objective (5.4.4) can be approximated with finite samples. We choose to sample K responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K \sim \pi_t(\cdot|\mathbf{x})$ for each prompt $\mathbf{x}$, and denote the empirical distribution by $\widehat{\pi}_t^K$. The finite-sample optimization problem can be approximated as

$$\pi_{t+1} = argmin_\pi \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})}\bigg(\log\bigg(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\bigg) - \Big(\eta\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) - \log Z_{\widehat{\pi}_t^K}(\mathbf{x})\Big)\bigg)^2. \quad (5.4.5)$$

Specifically, $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) = \sum_{k=1}^K \mathbb{P}(\mathbf{y} \succ \mathbf{y}_k|\mathbf{x})/K$ and $Z_{\widehat{\pi}_t^K}(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})}[\exp(\eta\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}))]$. $Z_{\widehat{\pi}_t^K}(\mathbf{x})$, treated as an expectation, can be further estimated by B new samples with in total $O(KB)$ queries of the preference oracle $\mathbb{P}$. (5.4.5) is an efficiently tractable optimization problem. Informally speaking, when $K \to \infty$, (5.4.5) will recover (5.4.4). We have the following guarantee on the convergence of (5.4.4):

Theorem 5.4.1. Assume the optimization problem (5.4.4) is realizable. Denote π_t as the policy obtained via (5.4.4) and the mixture policy $\bar{\pi}_T = \frac{1}{T}\sum_{t=1}^T \pi_t$. By setting $\eta = \Theta(1/\sqrt{T})$, we have that

$$\max_\pi \big[\mathbb{P}(\pi \succ \bar{\pi}_T)\big] - \min_\pi \big[\mathbb{P}(\pi \prec \bar{\pi}_T)\big] = O(1/\sqrt{T}).$$

Theorem 5.4.1 characterizes the convergence rate of the average policy across the time horizon T towards the Nash equilibrium, in terms of the duality gap. The proof is based on Theorem 1 in Freund and Schapire (1999) with slight modification. For completeness, we include the proof in Appendix 5.7.

Alternatively, we can avoid estimating $\log Z_{\widehat{\pi}_t^K}(\mathbf{x})$ by replacing it simply with $\eta/2$[3] in (5.4.5) to obtain a more clear objective:

$$\pi_{t+1} = argmin_\pi \mathbb{E}_{\mathbf{x}\sim\mathcal{X},\mathbf{y}\sim\pi_t(\cdot|\mathbf{x})}\bigg(\log\bigg(\frac{\pi(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\bigg) - \eta\bigg(\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) - \frac{1}{2}\bigg)\bigg)^2. \quad (5.4.6)$$

Intuitively, if a tie occurs (i.e., $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) = 1/2$), we prefer the model does not update weight at $\mathbf{y}$. If $\mathbf{y}$ wins over $\widehat{\pi}_t^K$ on average (i.e., $\mathbb{P}(\mathbf{y} \succ \widehat{\pi}_t^K|\mathbf{x}) > 1/2$), then we increase

[3] Assuming the winning probability between any given pair is either 1 or 0 with equal chance, when $K \to \infty$, we can show that indeed $Z_{\widehat{\pi}_t^K}(\mathbf{x}) \to e^{\eta/2}$.

the probability density at $\mathbf{y}$ to employ the advantage of $\mathbf{y}$ over $\widehat{\pi}_t^K$. In our experiments, we choose to minimize the objective (5.4.6).

5.4.2 The SPPO Algorithm

Based on the aformentioned theoretical framework, we propose the *Self-Play Preference Optimization* algorithm in Algorithm 22.

Algorithm 22 `Self-Play Preference Optimization(SPPO)`

```
1: input: base policy $\pi_{\boldsymbol{\theta}_1}$, preference oracle $\mathbb{P}$, learning rate $\eta$, number of generated samples $K$.
2: for $t = 1, 2, \dots$ do
3:     Generate synthetic responses by sampling $\mathbf{x} \sim \mathcal{X}$ and $\mathbf{y}_{1:K} \sim \pi_t(\cdot|\mathbf{x})$.
4:     Annotate the win-rate $\mathbb{P}(\mathbf{y}_k \succ \mathbf{y}_{k'}|\mathbf{x}), \forall k, k' \in [K]$.
5:     Select responses from $\mathbf{y}_{1:K}$ to form dataset $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i, \widehat{P}(\mathbf{y}_i \succ \pi_t|\mathbf{x}_i))\}_{i\in[N]}$.
6:     Optimize $\pi_{\boldsymbol{\theta}_{t+1}}$ according to (5.4.6):
           $$\boldsymbol{\theta}_{t+1} \leftarrow argmin_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y},\widehat{P}(\mathbf{y}\succ\pi_t|\mathbf{x}))\sim\mathcal{D}_t}\bigg(\log\bigg(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\bigg) - \eta\bigg(\widehat{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - \frac{1}{2}\bigg)\bigg)^2. \quad (5.4.7)$$
7: end for
```

In each round t, Algorithm 22 will first generate K responses $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K$ according to $\pi_t(\cdot|\mathbf{x})$ for each prompt $\mathbf{x}$ (Line 3). Then, the preference oracle $\mathbb{P}$ will be queried to calculate the win rate among the K responses (Line 4). At Line 5, certain criteria can be applied to determine which response should be kept in the constructed dataset $\mathcal{D}_t$ and construct the prompt-response-probability triplet $(\mathbf{x}, \mathbf{y}, \widehat{P}(\mathbf{y} \succ \pi_t|\mathbf{x}))$. We will discuss the design choices later in Section 5.5. One straightforward design choice is to include all K responses into $\mathcal{D}_t$ and each $\widehat{P}(\mathbf{y}_i \succ \pi_t|\mathbf{x})$ is estimated by comparing $\mathbf{y}_i$ to all K responses. In total, $O(K^2)$ queries will be made. Then the algorithm will optimize (5.4.6) on the dataset $\mathcal{D}_t$ (Line 6).

Comparison with DPO, IPO, and KTO In practice, we utilize mini-batches of more than 2 responses to estimate the win rate of a given response, while the DPO and IPO loss focus on a single pair of responses. When only a pair of responses $\mathbf{y}_w$ and $\mathbf{y}_l$ is available, we have the pair-wise symmetric loss based on the preference triplet $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$ defined as:

$$\begin{aligned}\ell_{\mathrm{SPPO}}(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l; \boldsymbol{\theta}; \pi_{\mathrm{ref}}) := &\left(\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_w|\mathbf{x})}\right) - \eta\Big(\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x}) - \frac{1}{2}\Big)\right)^2 \\ &+ \left(\log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_l|\mathbf{x})}\right) - \eta\Big(\mathbb{P}(\mathbf{y}_w \prec \mathbf{y}_l|\mathbf{x}) - \frac{1}{2}\Big)\right)^2, \quad (5.4.8)\end{aligned}$$

where $\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})$ can be either a soft probability within $[0, 1]$ or a hard label 1 indicating $\mathbf{y}_w \succ \mathbf{y}_l$.

We now compare the SPPO loss to other baselines assuming a hard label $\mathbf{y}_w \succ \mathbf{y}_l$ is given. For the ease of comparison, let

$$a = \beta \log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_w|\mathbf{x})}\right), b = \beta \log\left(\frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x})}{\pi_{\mathrm{ref}}(\mathbf{y}_l|\mathbf{x})}\right), c = \beta \mathrm{KL}(\pi_{\boldsymbol{\theta}} \| \pi_{\mathrm{ref}}),$$

then we have

$$\ell_{\mathrm{DPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = -\log \sigma(a - b), \quad (5.4.9)$$

$$\ell_{\mathrm{IPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = [(a - b) - 1]^2, \quad (5.4.10)$$

$$\ell_{\mathrm{KTO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = \sigma(-a + c) + \sigma(b - c) \text{ (simplified)}, \quad (5.4.11)$$

where $\sigma(x) = e^x/(e^x + 1)$ and the SPPO loss can be written as

$$\ell_{\mathrm{SPPO}}(\mathbf{y}_w, \mathbf{y}_l, \mathbf{x}) = (a - 1/2)^2 + (b + 1/2)^2.$$

It can be seen that SPPO not only pushes the gap between a and b to be 1, but also attempts to push value of a to be close to $1/2$ and the value of b to be close to $-1/2$ such that $\pi_{\boldsymbol{\theta}}(\mathbf{y}_w|\mathbf{x}) > \pi_{\mathrm{ref}}(\mathbf{y}_w|\mathbf{x})$ and $\pi_{\boldsymbol{\theta}}(\mathbf{y}_l|\mathbf{x}) < \pi_{\mathrm{ref}}(\mathbf{y}_l|\mathbf{x})$. We believe this is particularly important: when there are plenty of preference pairs, DPO and IPO can ensure the policy will converge to the target policy, but when the preference pairs are scarce (e.g., one pair for each prompt), there is no guarantee that the estimated reward of the winner a will increase and the estimated reward of the loser b will decrease. Instead, only the reward gap between the winner and the loser (i.e., $a-b$) will increase. This phenomenon is observed by Pal et al. (2024) that DPO only drives the loser's likelihood to be small, but the winner's likelihood barely changes. We believe that fitting $\beta \log\left(\frac{\pi_{t+1}(\mathbf{y}|\mathbf{x})}{\pi_t(\mathbf{y}|\mathbf{x})}\right)$ directly to $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x}) - 1/2$ is more effective than IPO which attempts to fit $\beta \log\left(\frac{\pi_{t+1}(\mathbf{y}_w|\mathbf{x})}{\pi_t(\mathbf{y}_w|\mathbf{x})}\right) - \beta \log\left(\frac{\pi_{t+1}(\mathbf{y}_l|\mathbf{x})}{\pi_t(\mathbf{y}_l|\mathbf{x})}\right)$ to $\mathbb{P}(\mathbf{y}_w \succ \pi_t|\mathbf{x}) - \mathbb{P}(\mathbf{y}_l \succ \pi_t|\mathbf{x})$. In addition, SPPO shares a similar spirit as KTO. The KTO loss pushes a to be large by minimizing $\sigma(-a+c)$ and pushes b to be small by minimizing $\sigma(b-c)$. In contrast, SPPO pushes a to be as large as $1/2$ and b to be as small as $-1/2$.

On the other hand, we would like to comment that although DPO and KTO can be extended to their iterative variants, they are not by nature iterative algorithms and do not have provable guarantees that they can reach the Nash equilibrium. In contrast, SPPO and IPO are by design capable to solve the Nash equilibrium iteratively. SPPO is superior to IPO because its design explicitly alleviates the data sparsity issue, as discussed above and detailed in Pal et al. (2024).

5.5 Experiments

We conduct extensive experiments to show the performance of our method and compare it with other baselines.

5.5.1 Experiment Setup

Base Model and Datasets We follow the experimental setup of Snorkel[4], a model that utilizes iterative DPO to achieve state-of-the-art performance on AlpacaEval benchmarks. Specifically, we use Mistral-7B-Instruct-v0.2 as our base model[5]. Mistral-7B-Instruct-v0.2

[4]`https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO`

[5]`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`

is an instruction fine-tuned version of Mistral-7B-v0.2 model (Jiang et al., 2023a). We also adopt Ultrafeedback (Cui et al., 2023) as our source of prompts which includes around 60k prompts from diverse resources. During generation, we follow the standard chat template of Mistral-7B. To avoid overfitting during the fine-tuning, we split the dataset into three portions and use only one portion per iteration. These settings were also adopted by training the model Snorkel-Mistral-PairRM-DPO[6] (Snorkel). We follow the splitting in Snorkel for a fair comparison. Additionally, we use Llama-3-8B-Instruct[7] as a stronger base model along with the same preference dataset and data splitting.

Preference Model We employ PairRM (Jiang et al., 2023b), an efficient pair-wise preference model of size 0.4B. PairRM is based on DeBERTA-V3 (He et al., 2021) and trained on high-quality human-preference datasets. Results on benchmarks like Auto-J Pairwise dataset (Li et al., 2023a) show that it outperforms most of the language-model-based reward models and performs comparably with larger reward models like UltraRM-13B (Cui et al., 2023). We refer the readers to the homepage on Huggingface[8] for detailed benchmark results. We therefore keep PairRM as our ranking model following Snorkel for a balance between accuracy and efficiency.

Specifically, PairRM will output a "relative reward" $s(\mathbf{y}, \mathbf{y}'; \mathbf{x})$ that reflects the strength difference between $\mathbf{y}$ and $\mathbf{y}'$, i.e.,

$$\mathbb{P}(\mathbf{y} \succ \mathbf{y}'|\mathbf{x}) = \frac{\exp(s(\mathbf{y}, \mathbf{y}'; \mathbf{x}))}{1 + \exp(s(\mathbf{y}, \mathbf{y}'; \mathbf{x}))}.$$

Unlike the Bradley-Terry-based reward model, PairRM only assigns the relative reward which is not guaranteed to be transitive (i.e., $s(\mathbf{y}_1, \mathbf{y}_2; \mathbf{x}) + s(\mathbf{y}_2, \mathbf{y}_3; \mathbf{x}) \neq s(\mathbf{y}_1, \mathbf{y}_3; \mathbf{x})$). So it indeed models the general preference.

[6]`https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO`

[7]`https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`

[8]`https://huggingface.co/llm-blender/PairRM`

Response Generation and Selection During the generation phase in each iteration, we use top $p = 1.0$ and temperature 1.0 to sample from the current policy. We sample with different random seeds to get $K = 5$ different responses for each prompt. Previous works utilizing Iterative DPO choose 2 responses to form a pair for each prompt. For a fair comparison, we do not include all $K = 5$ responses in the preference data but choose two responses among them. Following Snorkel, we choose the winner $\mathbf{y}_w$ and loser $\mathbf{y}_l$ to be the response with the *highest* and *lowest* PairRM score, which is defined for each response $\mathbf{y}_i$ as:

$$s_{\text{PairRM}}(\mathbf{y}_i; \mathbf{x}) := \frac{1}{K} \sum_{k=1}^{K} s(\mathbf{y}_i, \mathbf{y}_k; \mathbf{x}).$$

Probability Estimation We then estimate the win rate over the distribution by the average win rate over all the sampled responses as explained in (5.4.5):

$$\widehat{P}(\mathbf{y}_i \succ \pi_t | \mathbf{x}_i) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{P}(\mathbf{y}_i \succ \mathbf{y}_k | \mathbf{x}), \forall i \in [K].$$

Hyperparameter Tuning The experiments are conducted on 8 $\times$ Nvidia A100 GPUs. For SPPO, we trained three iterations in total. In each iteration, we selected the model trained on the first epoch of the 20k prompts from UltraFeedback to proceed to the next iteration. For both Mistral-7B-Instruct-v0.2 and Llama-3-8B-Instruct, the global training batch size is set to 64, and η is set to $1e3$. The learning rate schedule is determined by the following hyperparameters: learning rate=5.0e-7, number of total training epochs=1, warmup ratio=0.1, linear schedule. The best hyper-parameters for each model is selected by the average win-rate (judged by PairRM-0.4B) on a hold-out subset of Ultrafeedback as the metric. For more details on the win-rate comparison using PairRM as a judge, please refer to Section 5.5.2 and Figure 5.3.

Baselines We evaluate the following base models as well as baseline methods for fine-tuning LLMs:

- Mistral-7B-Instruct-v0.2: Mistral-7B-Instruct-v0.2 is an instruction fine-tuned version of Mistral-7B-v0.2 model (Jiang et al., 2023a). It is the starting point of our algorithm.

- Snorkel (Mistral-PairRM-DPO): We directly evaluate the uploaded checkpoint on HuggingFace[9]. This model is obtained by three rounds of iterative DPO from Mistral-7B-Instruct-v0.2.

- (Iterative) DPO: We also implement the iterative DPO algorithm by ourselves. The experimental settings and model selection schemes align with those used for SPPO, except for the adoption of the DPO loss function as defined in (5.4.9). Hyperparameters are optimized to maximize the average win-rate assessed by PairRM at each iteration. Note that the practical algorithm in Rosset et al. (2024) is essentially the same as iterative DPO.

- (Iterative) IPO: We implement the iterative IPO algorithm by ourselves. The experimental setting and the model selection scheme is the same as iterative DPO, except that the loss function is the IPO loss (5.4.10). For fair comparison, hyperparameters for IPO is also selected by evaluation using the average PairRM win-rate on the hold-out subset of Ultrafeedback.

- Self-rewarding LM: Yuan et al. (2024) proposed to prompt the LLM itself as a preference judge to construct new preference pairs and iteratively fine-tune the LLM with the DPO algorithm. We use the AlpacaEval 2.0 win rate reported by Yuan et al. (2024) for comparison. Note that Self-rewarding LM is a trained from Llama 2 70B.

- Llama-3-8B-Instruct: Llama-3-8B-Instruct is an instruction-tuned model optimized for dialogue use cases and outperforms many of the available open-source chat models on common industry benchmarks.

[9]https://huggingface.co/snorkelai/Snorkel-Mistral-PairRM-DPO

Benchmarks Following previous works, we use AlpacaEval 2.0 (Dubois et al., 2024a), MT-Bench (Zheng et al., 2024), and Open LLM Leaderboard (Beeching et al., 2023a) as our evaluation benchmarks.

- **AlpacaEval 2.0** is an LLM-based automatic evaluation benchmark. It employs AlpacaFarm (Dubois et al., 2024b) as its prompts set composed of general human instructions. The model responses and the reference response generated by GPT-4-Turbo are fed into a GPT-4-Turbo-based annotator to be judged. We follow the standard approach and report the win rate over the reference responses.

- **MT-Bench** (Zheng et al., 2024) is a collection of 80 high-quality multi-turn open-ended questions. The questions cover topics like writing, role-playing, math, coding, etc.. The generated answer is judged by GPT-4 and given a score directly without pairwise comparison.

- **Open LLM Leaderboard** (Beeching et al., 2023a) consists of six datasets, each of which focuses on a facet of language model evaluation. In detail, the evaluation rubric includes math problem-solving, language understanding, human falsehood mimicking, and reasoning. We follow the standard evaluation process and use in-context learning to prompt the language model and compute the average score over six datasets to measure the performance.

5.5.2 Experimental Results

We evaluate the models on the three benchmarks described above. We also compare models based on the pre-trained preference model PairRM.

Evaluation using GPT-4 as a judge In the assessment of AI chatbots, human evaluation remains the benchmark for quality and accuracy (Askell et al., 2021; Ouyang et al., 2022). However, due to its limitations in scalability and reproducibility, we explore the alternative approach of using the advanced capabilities of GPT-4 (OpenAI et al., 2023) as an automatic evaluation tool. We conduct GPT-4-based automatic evaluation on AlpacaEval 2.0 (Li et al., 2023b) and MT-Bench (Zheng et al., 2023) to measure the chatbot capability of our model. The results can be found in Table 5.1 for AlpacaEval 2.0 and Figure 5.2 (left) for MT-Bench. We also provide a radar chart analyzing the MT-Bench results in Figure 5.2 (right). We found that the performance of SPPO models consistently improves along with the iterative alignment iterations.

Table 5.1 (AlpacaEval 2.0) shows the win rate over the GPT-4-Turbo baseline of different models on 805 prompts. We also include one column indicating the length-controlled win rate, and one column on the average length of each model, to account for the tendency of the LLM-based judge to favor longer sequence outputs — an issue colloquially termed the "reward hacking" phenomenon. According to the table, Mistral-7B-SPPO Iter3 has the highest win rate, 28.52% for the length-controlled version, and 31.02% for the overall win rate. The performance gains over previous iterations are 7.69% (Mistral-7B-Instruct → Iter1), 2.10% (Iter1 → Iter2), and 1.64% (Iter2 → Iter3), respectively, indicating steady improvements across iterations, as illustrated in Figure 5.1. We also apply SPPO to a stronger baseline model, i.e., Llama-3-8B-Instruct, and the fine-tuned model Llama-3-8B-SPPO has a higher length-controlled win rate 38.77% and overall win rate 39.85%. The performance gains are more significant: 8.81% (Llama-3-8B-Instruct → Iter1), 3.42% (Iter1 → Iter2), and 3.62% (Iter2 → Iter3), summing up to a total gain of 15.85%.

Additionally, the data indicates that SPPO achieves superior performance compared to the iterative variants of DPO and IPO. The length-controlled win rate for SPPO reaches 28.53%, outperforming the DPO's best rate of 26.39% (by Snorkel) and IPO's rate of 25.45% . Notably, while DPO and IPO training tend to significantly increase the average output length—2736 and 2654, respectively—SPPO shows a more moderate length increase, moving from 1676 in the base model to 2163 at the third iteration. This suggests that SPPO improves performance while more effectively controlling the tendency towards longer output

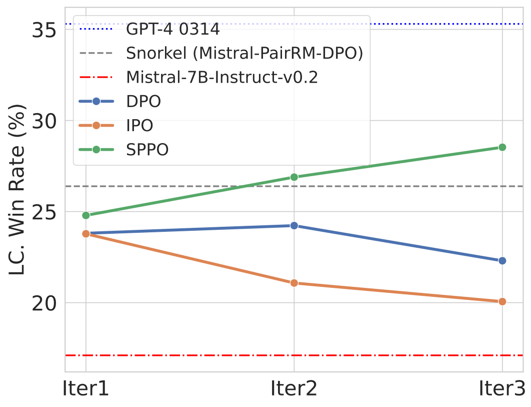
Table 5.1: AlpacaEval 2.0 evaluation of various models (detailed in Baselines) in terms of both normal and length-controlled (LC) win rates in percentage (%). Mistral-7B-SPPO Iter3 model achieves the highest LC win rate of 28.53% and a normal win rate of 31.02%. SPPO demonstrates steady performance gains across iterations and outperforms other baselines which show a tendency to produce longer responses. Additionally, re-ranking with the PairRM reward model (best-of-16) at test time consistently enhances the performance across all models and SPPO (best-of-16) achieves high win rate *without strong external supervision like GPT-4*. We additionally include the results obtained from fine-tuning Llama-3-8B-Instruct, which also show steady performance improvement.

Model	AlpacaEval 2.0		
	LC Win Rate	Win Rate	Avg. Len
Mistral-7B-Instruct-v0.2	17.11	14.72	1676
Mistral-7B-Instruct-v0.2 (best-of-16)	22.45	17.94	1529
Snorkel (Mistral-PairRM-DPO)	26.39	30.22	2736
Snorkel (Mistral-PairRM-DPO best-of-16)	29.97	34.86	2616
Self-Rewarding 70B Iter1	-	9.94	1092
Self-Rewarding 70B Iter2	-	15.38	1552
Self-Rewarding 70B Iter3	-	20.44	2552
Mistral-7B-DPO Iter1	23.81	20.44	1723
Mistral-7B-DPO Iter2	24.23	24.46	2028
Mistral-7B-DPO Iter3	22.30	23.39	2189
Mistral-7B-IPO Iter1	23.78	20.77	1693
Mistral-7B-IPO Iter2	21.08	23.38	2660
Mistral-7B-IPO Iter3	20.06	22.47	2760
Mistral-7B-SPPO Iter1	24.79$_{(+7.69)}$	23.51$_{(+8.79)}$	1855
Mistral-7B-SPPO Iter2	26.89$_{(+2.10)}$	27.62$_{(+4.11)}$	2019
Mistral-7B-SPPO Iter3	**28.53**$_{(+1.64)}$	**31.02**$_{(+3.40)}$	2163
Mistral-7B-SPPO Iter1 (best-of-16)	28.71$_{(+6.26)}$	27.77$_{(+9.83)}$	1901
Mistral-7B-SPPO Iter2 (best-of-16)	31.23$_{(+2.52)}$	32.12$_{(+4.35)}$	2035
Mistral-7B-SPPO Iter3 (best-of-16)	**32.13**$_{(+0.9)}$	**34.94**$_{(+2.82)}$	2174
Llama-3-8B-Instruct	22.92	22.57	1899
Llama-3-8B-SPPO Iter1	31.73$_{(+8.81)}$	31.74$_{(+9.17)}$	1962
Llama-3-8B-SPPO Iter2	35.15$_{(+3.42)}$	35.98$_{(+4.24)}$	2021
Llama-3-8B-SPPO Iter3	**38.77**$_{(+3.62)}$	**39.85**$_{(+3.87)}$	2066

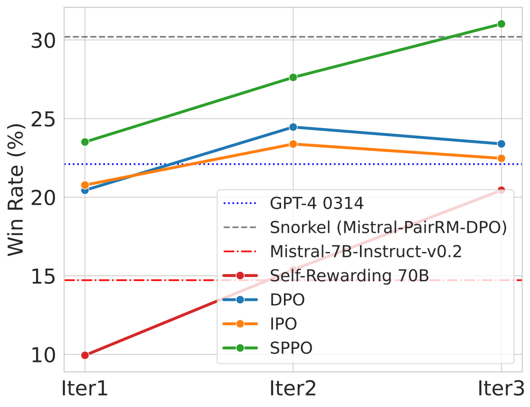
Table 5.2: AlpacaEval 2.0 leaderboard results of both normal and length-controlled (LC) win rates in percentage (%). Mistral-7B-SPPO can outperform larger models and Mistral-7B-SPPO (best-of-16) can outperform proprietary models such as GPT-4(6/13). Llama-3-8B-SPPO exhibits even better performance.

Model	AlpacaEval 2.0	
	LC. Win Rate	Win Rate
GPT-4 Turbo	50.0	50.0
Claude 3 Opus	40.5	29.1
Llama-3-8B-SPPO Iter3	38.8	39.9
GPT-4 0314	35.3	22.1
Llama 3 70B Instruct	34.4	33.2
Mistral-7B-SPPO Iter3 (best-of-16)	32.1	34.9
GPT-4 0613	30.2	15.8
Snorkel (Mistral-PairRM-DPO best-of-16)	30.0	34.9
Mistral Medium	28.6	21.9
Mistral-7B-SPPO Iter3	28.5	31.0
Claude 2	28.2	17.2
Snorkel (Mistral-PairRM-DPO)	26.4	30.2
Gemini Pro	24.4	18.2
Mistral 8×7B v0.1	23.7	18.1
Llama 3 8B Instruct	22.9	22.6
GPT-3.5 Turbo 0613	22.7	14.1
Vicuna 33B v1.3	17.6	12.7

lengths compared to DPO and IPO. Finally, we present the best-of-16 results for each model, selected using the PairRM reward model. We find that re-ranking with the preference model at test time can consistently improve the performance of base model (Mistral-7B-Instruct-v0.2), DPO (Snorkel), and SPPO (Iter3) by 5.34%, 3.57%, and 3.6%, respectively. Notably,



(a)



(b)

Figure 5.1: Win Rate against GPT-4-Turbo with (a) and without (b) Length Controlling (LC) on AlpacaEval 2.0. SPPO demonstrates steady improvements on both LC and raw win rates.

this shows that while SPPO significantly enhances model alignment using PairRM-0.4B as the sole external supervision, it has not resulted in over-optimization against the preference model (Gao et al., 2023). Future work will explore further improvements in model alignment, potentially through additional iterations beyond the current three (following Snorkel's methodology).

In Table 5.2, we compare SPPO on the AlpacaEval 2.0 leaderboard with other state-of-the-art AI chatbots. We found our SPPO model outperforms many competing models trained on proprietary alignment data (e.g., Claude 2, Gemini Pro, & Llama 3 8B Instruct). When applied to Llama 3 8B Instruct, our Llama-3-8B-SPPO exhibits an even higher win rate. With test-time reranking, Mistral-7B-SPPO Iter3 (best-of-16) is even competitive to GPT-4 0613 and Llama 3 70B Instruct.

In Figure 5.2 (left), we evaluate the performance of SPPO on MT-Bench. We can see that Mistral-7B-SPPO Iter3 outperforms all baseline models, achieving an average score of 7.59. While we are not certain why the MT-Bench performance drops at the first two iterations, the performance of SPPO at the final iteration still improves over the base model. Since the length-controlled AlpacaEval 2.0 has a 98% Pearson correlation with human evaluations

Model	MT-Bench 1st Turn	MT-Bench 2nd Turn	MT-Bench Average
Mistral-7B-Instruct-v0.2	7.78	7.25	7.51
Snorkel (Mistral-PairRM-DPO)	7.83	7.33	7.58
Mistral-7B-DPO Iter1	7.45	6.58	7.02
Mistral-7B-DPO Iter2	7.57	6.56	7.06
Mistral-7B-DPO Iter3	7.49	6.69	7.09
Mistral-7B-SPPO Iter1	7.63	6.79	7.21
Mistral-7B-SPPO Iter2	7.90	7.08	7.49
Mistral-7B-SPPO Iter3	7.84	7.34	**7.59**

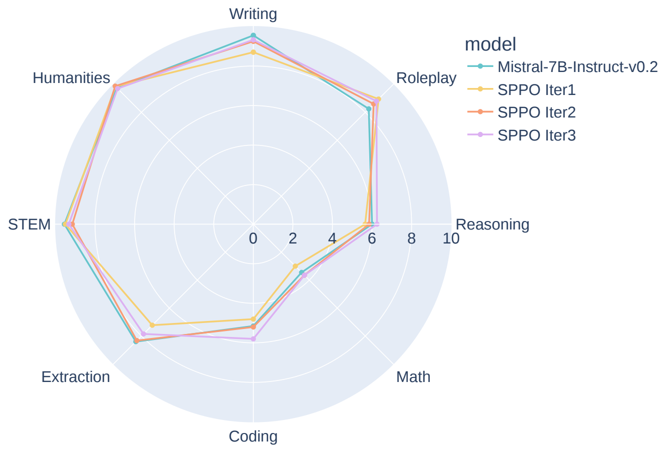


Figure 5.2: **MT-Bench Evaluation.** Left: Mistral-7B-SPPO Iter3 outperforms all baseline models by achieving an average score of 7.59. Despite initial drops in performance in the first two iterations, SPPO Iter3 improves upon the base model by the final iteration. Right: Radar chart of MT-Bench results for Mistral-7B-SPPO. SPPO Iter3's improves across different MT-Bench categories, showing significant gains in RolePlay, Reasoning, Math, and Coding tasks.

and 10× more evaluation prompts, it likely provides a more reliable evaluation than MT-Bench. To gain a deeper understanding on MT-Bench performance, we plot the improvement in Figure 5.2 (right), broken down by question prompt category. Mistral-7B-SPPO Iter3 demonstrates notable gains in RolePlay, Reasoning, Math, and Coding tasks.

Open LLM Leaderboard We further evaluate the capabilities of SPPO models using Huggingface Open LLM Leaderboard (Beeching et al., 2023b). This leaderboard encompasses 6 different datasets, each focusing on a specific capability of LLMs: Arc (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2021), and GSM8k (Cobbe et al., 2021). The models are prompted with zero or few-shot exemplars. The results, presented in Table 5.3, demonstrate that SPPO can enhance the performance of the base model on Arc, TruthfulQA, and GSM8k, and achieve the state-of-the-art performance with an averagte score of 66.75. However, these

Table 5.3: **Open LLM Leaderboard Evaluation**. SPPO fine-tuning improves the base model's performance on different tasks, reaching a state-of-the-art average score of 66.75 for Mistral-7B and 70.29 for Llama-3-8B. For Mistral-7B, subsequent iterations of DPO, IPO, and SPPO see a decline in performance. It is possible that aligning with human preferences (simulated by the PairRM preference model in our study) may not always enhance, and can even detract from, overall performance.

Models	Arc	TruthfulQA	WinoGrande	GSM8k	HellaSwag	MMLU	Average
Mistral-7B-Instruct-v0.2	63.65	66.85	77.98	41.93	84.89	59.15	65.74
Snorkel	66.04	70.86	77.74	36.77	85.64	60.83	66.31
Mistral-7B-DPO Iter1	63.14	68.39	77.19	40.33	85.25	59.41	65.62
Mistral-7B-DPO Iter2	64.16	67.84	76.09	39.95	85.23	59.03	65.38
Mistral-7B-DPO Iter3	65.19	67.89	77.27	32.30	85.49	59.00	64.52
Mistral-7B-IPO Iter1	64.68	68.60	77.98	43.75	85.08	59.04	66.52
Mistral-7B-IPO Iter2	62.12	66.30	77.51	39.20	83.15	59.70	64.66
Mistral-7B-IPO Iter3	62.97	67.12	77.51	37.45	83.69	59.57	64.72
Mistral-7B-SPPO Iter1	65.02	69.40	77.82	43.82	85.11	58.84	66.67
Mistral-7B-SPPO Iter2	65.53	69.55	77.03	44.35	85.29	58.72	**66.75**
Mistral-7B-SPPO Iter3	65.36	69.97	76.80	42.68	85.16	58.45	66.40
Llama-3-8B-Instruct	62.29	51.65	76.09	75.89	78.73	65.59	68.37
Llama-3-8B-SPPO Iter1	63.82	54.96	76.40	75.44	79.80	65.65	69.35
Llama-3-8B-SPPO Iter2	64.93	56.48	76.87	75.13	80.39	65.67	69.91
Llama-3-8B-SPPO Iter3	65.19	58.04	77.11	74.91	80.86	65.60	**70.29**

improvements do not hold in subsequent alignment iterations: DPO, IPO, and SPPO's performance declines after the first or second iterations. This limitation may be attributed to the "alignment tax" phenomenon (Askell et al., 2021), which suggests that aligning with human preferences (simulated by PairRM preference in our study) might not improve or even hurt the general performance. Improving language model capabilities through alignment iterations remains a topic for future research, and we posit that incorporating high-quality SFT annotations (Chen et al., 2024) could play a significant role in this endeavor.

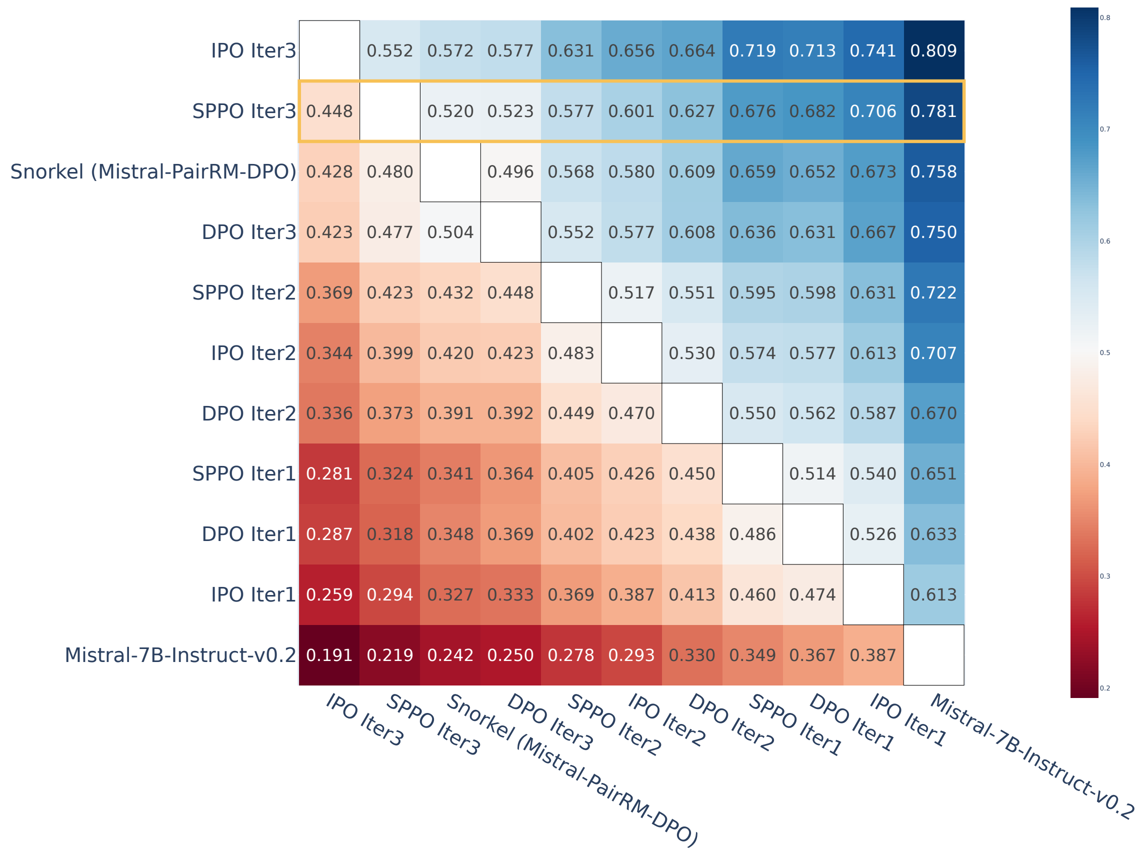


Figure 5.3: Pairwise win rates among base model (Mistral-7B-Instruct-v0.2), DPO models, IPO models, and SPPO models using **PairRM-0.4B** as a judge, which may favor models with longer outputs. On benchmarks with more powerful judge models (e.g., GPT-4), such as AlpacaEval 2.0 and MT-Bench, SPPO outperforms other baseline algorithms by a large margin.

Evaluation using PairRM as a judge As SPPO identifies the von Neumann winner (see (5.3.3)) in a two-player constant-sum game, we examine the pairwise preferences among SPPO models and other baselines. The pairwise win rates, measured by PairRM, are depicted in Figure 5.3. We observe that in all algorithms—namely DPO, IPO, and SPPO—the newer model iterations surpass the previous ones. For example, SPPO Iteration 3 outperforms SPPO Iteration 2. Both SPPO and IPO consistently outperform DPO across all iterations. While SPPO is superior to IPO in the first two iterations, IPO exceeds SPPO in performance during the final iteration. Considering the superior performance of SPPO in

Mini-Batch Size	Iteration	AlpacaEval 2.0 Win Rate LC.	AlpacaEval 2.0 Win Rate Raw	AlpacaEval 2.0 Avg. Len (chars)
$K = 2$	Iter1	23.85	23.53	1948
	Iter2	26.91	27.24	1999
	Iter3	28.26	28.22	1961
$K = 5$	Iter1	24.79	23.51	1855
	Iter2	26.89	27.62	2019
	Iter3	**28.53**	**31.02**	2163

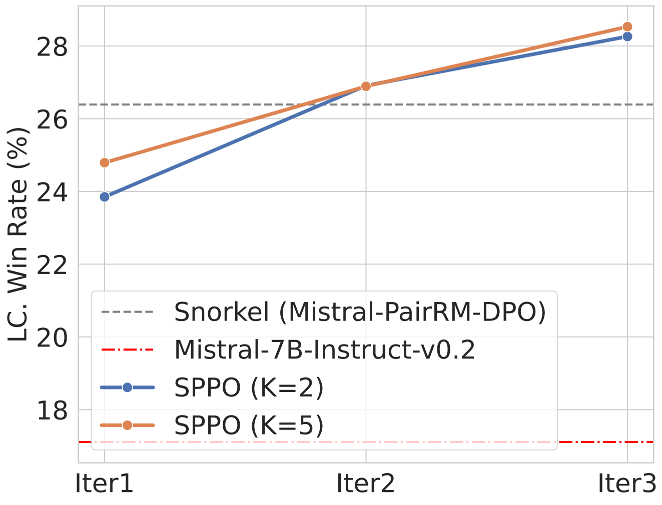


Figure 5.4: AlpacaEval 2.0 evaluation on SPPO of different mini-batch size in terms of both normal and length-controlled (LC) win rates in percentage (%). $K = 2, 5$ denote different mini-batch sizes when estimating the win rate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$.

standard benchmarks evaluated by GPT-4 or against ground-truth answers (e.g., AlpacaEval 2.0, MT-Bench, and Open LLM Leaderboard), along with IPO's tendency to produce longer sequence outputs (see Avg. Len in Table 5.1), we believe this is due to IPO exploiting the length bias in PairRM that favors longer sequences. Conversely, SPPO models benefit from a more robust regularization within a multiplicative weight update framework.

5.5.3 Ablation Study

We study the effect of mini-batch size when estimating the win rate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$. Specifically, for each prompt, we still generate 5 responses and choose the winner $\mathbf{y}_w$ and loser $\mathbf{y}_l$ according to the PairRM score. When estimating the probability, we varies the batch size to be $K = 2, 3, 5$. For $K = 2$, we estimate $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$ with only 2 samples $\mathbf{y}_w$ and $\mathbf{y}_l$:

$$\widehat{P}(\mathbf{y}_w \succ \pi_t|\mathbf{x}) = \frac{\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_w|\mathbf{x}) + \mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})}{2} = \frac{1/2 + \mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l|\mathbf{x})}{2},$$

and $\widehat{P}(\mathbf{y}_l \succ \pi_t|\mathbf{x})$ similarly. $K = 5$ indicates the original setting we use.

We compare the results on AlpacaEval 2.0, as shown in Figure 5.4. We find that the performance of SPPO is robust to the noise in estimating $\mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})$. While $K = 5$ initially outperforms $K = 2$ in the first iteration, the difference in their performance diminishes in subsequent iterations. Additionally, we observe that $K = 2$ exhibits a reduced tendency to increase output length.

5.6 Conclusions

This chapter introduced Self-Play Preference Optimization(SPPO), an approach to fine-tuning Large Language Models (LLMs) from Human/AI Feedback. SPPO has demonstrated significant improvements over existing methods such as DPO and IPO across multiple benchmarks, including AlpacaEval 2.0, MT-Bench, and the Open LLM Leaderboard. By integrating a preference model and employing a batched estimation process, SPPO aligns LLMs more closely with human preferences and avoids common pitfalls such as "length bias" reward hacking.

5.7 Omitted Proof

Proof of Theorem 5.4.1. Suppose the optimization problem is realizable, we have exactly that

$$\pi_{t+1}(\mathbf{y}|\mathbf{x}) \propto \pi_t(\mathbf{y}|\mathbf{x}) \exp(\eta \mathbb{P}(\mathbf{y} \succ \pi_t|\mathbf{x})), \text{ for } t = 1, 2, \ldots. \tag{5.7.1}$$

To prove that the exponential weight update can induce the optimal policy, we directly invoke a restated version of Theorem 1 in Freund and Schapire (1999):

Lemma 5.7.1 (Theorem 1 in Freund and Schapire (1999), restated)**.** For any oracle $\mathbb{P}$ and for any sequence of mixed policies $\mu_1, \mu_2, \ldots, \mu_T$, the sequence of policies $\pi_1, \pi_2, \ldots, \pi_T$

produced by (5.7.1) satisfies:

$$\sum_{t=1}^{T} \mathbb{P}(\pi_t \prec \mu_t) \leqslant \min_{\pi} \left[\frac{\eta}{1-e^{-\eta}} \sum_{t=1}^{T} \mathbb{P}(\pi \prec \mu_t) + \frac{\mathrm{KL}(\pi \| \pi_0)}{1-e^{-\eta}} \right].$$

By setting $\mu_t = \pi_t$, we have that

$$\frac{T}{2} \leqslant \min_{\pi} \left[\frac{\eta T}{1-e^{-\eta}} \mathbb{P}(\pi \prec \bar{\pi}_T) + \frac{\mathrm{KL}(\pi \| \pi_0)}{1-e^{-\eta}} \right],$$

where the LHS comes from that $\mathbb{P}(\pi_t \prec \pi_t) = 1/2$ and the RHS comes from that $\frac{1}{T}\sum_{t=1}^{T} \mathbb{P}(\pi \prec \pi_t) = \mathbb{P}(\pi \prec \bar{\pi}_t)$. Now rearranging terms gives

$$\frac{1-e^{-\eta}}{2\eta} \leqslant \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) + \frac{\mathrm{KL}(\pi \| \pi_0)}{\eta T} \right].$$

We can naively bound the KL-divergence $\mathrm{KL}(\pi \| \pi_0) \leqslant \| \log \pi_0(\cdot) \|_\infty$, which can be seen as a (large) constant.

By choosing $\eta = \frac{\| \log \pi_0(\cdot) \|_\infty}{\sqrt{T}}$, we have

$$\frac{1}{2} - \frac{\| \log \pi_0(\cdot) \|_\infty}{4\sqrt{T}} + O(T^{-1}) \leqslant \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] + \sqrt{\frac{\| \log \pi_0(\cdot) \|_\infty}{T}},$$

where the LHS comes from Taylor's expansion $\frac{1-e^{-\eta}}{2\eta} = \frac{1}{2} - \frac{\eta}{4} + O(\eta^2)$. Notice that 1/2 at the LHS is already the value of the symmetric two-player constant-sum game. This shows that for appropriately chosen η and T, the mixture policy $\bar{\pi}_T$ is close to the minimax optimal policy (Nash equilibrium).

The optimality gap is thus bounded by

$$\begin{aligned} & \max_{\pi} \left[\mathbb{P}(\pi \succ \bar{\pi}_T) \right] - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \\ & = \max_{\pi} \left[1 - \mathbb{P}(\pi \prec \bar{\pi}_T) \right] - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \\ & = 2 \left(\frac{1}{2} - \min_{\pi} \left[\mathbb{P}(\pi \prec \bar{\pi}_T) \right] \right) \\ & = O\left(\frac{1}{\sqrt{T}} \right). \end{aligned}$$

□

CHAPTER 6

Conclusions and Future Directions

This dissertation addressed several key concerns in learning from preference feedback, including the learning-to-rank problem and the problem of learning with general preference feedback from the perspective of both theoretical analysis and empirical evaluation. Several practical algorithms were proposed to achieve competitive performance with theoretical guarantees. Specifically, the dissertation made the following key contributions:

- Proposed an adaptive sampling algorithm for heterogeneous rank aggregation that can efficiently identify accurate users and utilize this information to perform ranking. Theoretical guarantees showed the algorithm is comparable to an oracle that knows the best user, with only a sublinear gap in sample complexity.
- Developed an active ranking algorithm called Probe-Rank that does not require the strong stochastic transitivity assumption. Probe-Rank was shown both theoretically and empirically to be more sample-efficient than existing methods when comparing nonadjacent items is more difficult.
- Introduced a new problem setting for identifying the Borda winner in generalized linear dueling bandits and proposed algorithms with matching upper and lower bounds on the Borda regret.
- Formulated preference learning as a two-player constant-sum game and developed a self-play algorithm called SPPO for large language model alignment. SPPO was

demonstrated to significantly improve performance across multiple benchmarks compared to existing methods.

This dissertation also suggests several potential directions and open questions for future research. The first direction is to develop more flexible yet meaningful assumptions to characterize human preferences from multiple sources. The current assumption of consistent ranking across all users is often not fully satisfied in practice. One potential direction is to assume each human user is independent from others, and the consensus preference between any two items is reached by majority voting from all users. This would allow more flexible preference behaviors while still providing some structure for learning algorithms. Also, it is desirable to develop more efficient and scalable algorithms for preference-based reinforcement learning in high-dimensional state and action spaces. Applying preference learning to complex sequential decision making problems remains challenging.

Bibliography

ABBASI-YADKORI, Y., PÁL, D. and SZEPESVARI, C. (2011). Improved algorithms for linear stochastic bandits. In *NIPS*.

AILON, N., KARNIN, Z. and JOACHIMS, T. (2014). Reducing dueling bandits to cardinal bandits. In *International Conference on Machine Learning*. PMLR.

ASKELL, A., BAI, Y., CHEN, A., DRAIN, D., GANGULI, D., HENIGHAN, T., JONES, A., JOSEPH, N., MANN, B., DASSARMA, N. ET AL. (2021). A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861* .

AUDIBERT, J.-Y., BUBECK, S. and MUNOS, R. (2010). Best arm identification in multi-armed bandits. In *COLT*.

AUER, P., CESA-BIANCHI, N. and FISCHER, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine Learning* **47** 235–256.

AZAR, M. G., ROWLAND, M., PIOT, B., GUO, D., CALANDRIELLO, D., VALKO, M. and MUNOS, R. (2023). A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036* .

BALSUBRAMANI, A., KARNIN, Z., SCHAPIRE, R. E. and ZOGHI, M. (2016). Instance-dependent regret bounds for dueling bandits. In *Conference on Learning Theory*. PMLR.

BALTRUNAS, L., MAKCINSKAS, T. and RICCI, F. (2010). Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*.

BEECHING, E., FOURRIER, C., HABIB, N., HAN, S., LAMBERT, N., RAJANI, N., SANSEVIERO, O., TUNSTALL, L. and WOLF, T. (2023a). Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L. and Wolf, T. (2023b). Open llm leaderboard. *Hugging Face* .

Bengs, V., Busa-Fekete, R., El Mesaoudi-Paul, A. and Hüllermeier, E. (2021). Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research* **22** 7–1.

Bengs, V., Haddenhorst, B. and Hüllermeier, E. (2023). Identifying copeland winners in dueling bandits with indifferences. *arXiv preprint arXiv:2310.00750* .

Bengs, V., Saha, A. and Hüllermeier, E. (2022). Stochastic contextual dueling bandits under linear stochastic transitivity models. In *International Conference on Machine Learning*. PMLR.

Bradley, R. A. and Terry, M. E. (1952). Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* **39** 324–345.

Brandt, J., Bengs, V., Haddenhorst, B. and Hüllermeier, E. (2022). Finding optimal arms in non-stochastic combinatorial bandits with semi-bandit feedback and finite budget. *Advances in Neural Information Processing Systems* **35** 20621–20634.

Braverman, M. and Mossel, E. (2008). Noisy sorting without resampling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*.

Busa-Fekete, R., Hüllermeier, E. and Mesaoudi-Paul, A. E. (2018). Preference-based online learning with dueling bandits: A survey. *ArXiv* **abs/1807.11398**.

Busa-Fekete, R., Szorenyi, B., Cheng, W., Weng, P. and Hüllermeier, E. (2013). Top-k selection based on adaptive sampling of noisy preferences. In *International Conference on Machine Learning*. PMLR.

Busa-Fekete, R., Szörényi, B. and Hüllermeier, E. (2014). Pac rank elicitation through adaptive sampling of stochastic pairwise preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28.

Calandriello, D., Guo, D., Munos, R., Rowland, M., Tang, Y., Pires, B. A., Richemond, P. H., Lan, C. L., Valko, M., Liu, T. et al. (2024). Human alignment of large language models through online preference optimisation. *arXiv preprint arXiv:2403.08635* .

Caplin, A. and Nalebuff, B. (1991). Aggregation and social choice: A mean voter theorem. *Econometrica: Journal of the Econometric Society* 1–23.

Chen, J., Chen, X., Zhang, Q. and Zhou, Y. (2017). Adaptive multiple-arm identification. In *International Conference on Machine Learning.*

Chen, X., Bennett, P. N., Collins-Thompson, K. and Horvitz, E. (2013). Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining.* ACM.

Chen, Z., Deng, Y., Yuan, H., Ji, K. and Gu, Q. (2024). Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335* .

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S. and Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems* **30**.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C. and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457* .

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R. et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* .

Conitzer, V. and Sandholm, T. (2005). Communication complexity of common voting rules. In *Proceedings of the 6th ACM conference on Electronic commerce*.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z. and Sun, M. (2023). Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377* .

Dani, V., Hayes, T. P. and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *Annual Conference Computational Learning Theory*.

Di, Q., Jin, T., Wu, Y., Zhao, H., Farnoud, F. and Gu, Q. (2024). Variance-aware regret bounds for stochastic contextual dueling bandits. In *The Twelfth International Conference on Learning Representations*.

Dubois, Y., Galambosi, B., Liang, P. and Hashimoto, T. B. (2024a). Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475* .

Dubois, Y., Li, C. X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P. S. and Hashimoto, T. B. (2024b). Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems* **36**.

Dudík, M., Hofmann, K., Schapire, R. E., Slivkins, A. and Zoghi, M. (2015). Contextual dueling bandits. *ArXiv* **abs/1502.06362**.

Dudík, M., Hofmann, K., Schapire, R. E., Slivkins, A. and Zoghi, M. (2015). Contextual dueling bandits. In *Conference on Learning Theory*. PMLR.

Dwork, C., Kumar, R., Naor, M. and Sivakumar, D. (2001). Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D. and Kiela, D. (2024). Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306* .

Even-Dar, E., Mannor, S. and Mansour, Y. (2002a). Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*. Springer.

Even-Dar, E., Mannor, S. and Mansour, Y. (2002b). Pac bounds for multi-armed bandit and markov decision processes. In *Annual Conference Computational Learning Theory*.

Falahatgar, M., Hao, Y., Orlitsky, A., Pichapati, V. and Ravindrakumar, V. (2017a). Maxing and ranking with few assumptions. *Advances in Neural Information Processing Systems* **30**.

Falahatgar, M., Jain, A., Orlitsky, A., Pichapati, V. and Ravindrakumar, V. (2018). The limits of maxing, ranking, and preference learning. In *International Conference on Machine Learning*.

Falahatgar, M., Orlitsky, A., Pichapati, V. and Suresh, A. T. (2017b). Maximum selection and ranking under noisy comparisons. *arXiv preprint arXiv:1705.05366* .

Farrell, R. H. (1964). Asymptotic behavior of expected sample size in certain one sided tests. *The Annals of Mathematical Statistics* 36–72.

Faury, L., Abeille, M., Calauzènes, C. and Fercoq, O. (2020). Improved optimistic algorithms for logistic bandits. In *International Conference on Machine Learning*. PMLR.

Feige, U., Raghavan, P., Peleg, D. and Upfal, E. (1994a). Computing with noisy information. *SIAM J. Comput.* **23** 1001–1018.

Feige, U., Raghavan, P., Peleg, D. and Upfal, E. (1994b). Computing with noisy information. *SIAM Journal on Computing* **23** 1001–1018.

Filippi, S., Cappe, O., Garivier, A. and Szepesvári, C. (2010). Parametric bandits: The generalized linear case. *Advances in Neural Information Processing Systems* **23**.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55** 119–139.

Freund, Y. and Schapire, R. E. (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior* **29** 79–103.

Gao, L., Schulman, J. and Hilton, J. (2023). Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*. PMLR.

Gao, Z., Chang, J. D., Zhan, W., Oertell, O., Swamy, G., Brantley, K., Joachims, T., Bagnell, J. A., Lee, J. D. and Sun, W. (2024). Rebel: Reinforcement learning via regressing relative rewards. *arXiv preprint arXiv:2404.16767* .

Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR.

Haddenhorst, B., Bengs, V. and Hüllermeier, E. (2021). On testing transitivity in online preference learning. *Machine Learning* **110** 2063–2084.

He, P., Gao, J. and Chen, W. (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

Heckel, R., Shah, N. B., Ramchandran, K. and Wainwright, M. J. (2019). Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics* **47** 3099–3126.

Heckel, R., Simchowitz, M., Ramchandran, K. and Wainwright, M. (2018). Approximate ranking from pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*. PMLR.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. and Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* .

Herbrich, R., Minka, T. and Graepel, T. (2006a). Trueskilltm: A bayesian skill rating system. In *NIPS*.

Herbrich, R., Minka, T. and Graepel, T. (2006b). Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems* **19**.

Hong, J., Lee, N. and Thorne, J. (2024). Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691* .

Jamieson, K., Katariya, S., Deshpande, A. and Nowak, R. (2015). Sparse dueling bandits. In *Artificial Intelligence and Statistics*. PMLR.

Ji, K., He, J. and Gu, Q. (2024). Reinforcement learning from human feedback with active queries. *arXiv preprint arXiv:2402.09401* .

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L. et al. (2023a). Mistral 7b. *arXiv preprint arXiv:2310.06825* .

Jiang, D., Ren, X. and Lin, B. Y. (2023b). Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561* .

Jin, T., Xu, P., Gu, Q. and Farnoud, F. (2020). Rank aggregation via heterogeneous thurstone preference models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jun, K.-S., Bhargava, A., Nowak, R. and Willett, R. (2017). Scalable generalized linear bandits: Online computation and hashing. *Advances in Neural Information Processing Systems* **30**.

Katariya, S., Jain, L., Sengupta, N., Evans, J. and Nowak, R. (2018). Adaptive sampling for coarse ranking. In *International Conference on Artificial Intelligence and Statistics*. PMLR.

Komiyama, J., Honda, J. and Nakagawa, H. (2016). Copeland dueling bandit problem: Regret lower bound, optimal algorithm, and computationally efficient algorithm. In *International Conference on Machine Learning*. PMLR.

Kuleshov, V. and Precup, D. (2014). Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028* .

Kumar, A. and Lease, M. (2011). Learning to Rank from a Noisy Crowd. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11, ACM, New York, NY, USA.

Lai, T. L., Robbins, H. et al. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics* **6** 4–22.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms.* Cambridge University Press.

Lattimore, T. and Szepesvári, C. (2020). *Bandit Algorithms.* Cambridge University Press.

Li, J., Sun, S., Yuan, W., Fan, R.-Z., Zhao, H. and Liu, P. (2023a). Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470* .

Li, L., Lu, Y. and Zhou, D. (2017). Provably optimal algorithms for generalized linear contextual bandits. In *International Conference on Machine Learning*. PMLR.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P. and Hashimoto, T. B. (2023b). Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Lin, S., Hilton, J. and Evans, O. (2021). Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958* .

Liu, C., Jin, T., Hoi, S. C. H., Zhao, P. and Sun, J. (2017). Collaborative topic regression for online recommender systems: an online and bayesian approach. *Machine Learning* **106** 651–670.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J. and Liu, J. (2023). Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657* .

Lou, H., Jin, T., Wu, Y., Xu, P., Gu, Q. and Farnoud, F. (2022). Active ranking without strong stochastic transitivity. *Advances in neural information processing systems* .

Maystre, L. and Grossglauser, M. (2017). Just sort it! a simple and effective approach to active preference learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org.

Minka, T. P., Cleven, R. and Zaykov, Y. (2018). Trueskill 2: An improved bayesian skill rating system.

Mohajer, S., Suh, C. and Elmahdy, A. (2017). Active learning for top-k rank aggregation from noisy comparisons. In *International Conference on Machine Learning*. PMLR.

Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T., Michi, A. et al. (2023). Nash learning from human feedback. *arXiv preprint arXiv:2312.00886* .

OpenAI, J., Achiam, Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* .

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A. et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* **35** 27730–27744.

Pal, A., Karkhanis, D., Dooley, S., Roberts, M., Naidu, S. and White, C. (2024). Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228* .

Pfeiffer, T., Gao, X. A., Chen, Y., Mao, A. and Rand, D. G. (2012). Adaptive polling for information aggregation. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Piech, C., Huang, J., Chen, Z., Do, C., Ng, A. and Koller, D. (2013). Tuned models of peer assessment in moocs. *arXiv preprint arXiv:1307.2579* .

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S. and Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* **36**.

Ramamohan, S., Rajkumar, A. and Agarwal, S. (2016). Dueling bandits: Beyond condorcet winners to general tournament solutions. In *NIPS*.

Ren, W., Liu, J. and Shroff, N. B. (2018). Pac ranking from pairwise and listwise queries: Lower bounds and upper bounds. *arXiv preprint arXiv:1806.02970* .

Ren, W., Liu, J. K. and Shroff, N. (2019). On sample complexity upper and lower bounds for exact ranking from noisy comparisons. In *Advances in Neural Information Processing Systems*.

Resler, A. and Mansour, Y. (2019). Adversarial online learning with noise. In *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*. PMLR, Long Beach, California, USA.

Rosset, C., Cheng, C.-A., Mitra, A., Santacroce, M., Awadallah, A. and Xie, T. (2024). Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715* .

Rowland, M., Omidshafiei, S., Tuyls, K., Pérolat, J., Valko, M., Piliouras, G. and Munos, R. (2019). Multiagent evaluation under incomplete information. In *NeurIPS*.

Rusmevichientong, P. and Tsitsiklis, J. N. (2010). Linearly parameterized bandits. *Mathematics of Operations Research* **35** 395–411.

Saha, A. (2021). Optimal algorithms for stochastic contextual preference bandits. *Advances in Neural Information Processing Systems* **34** 30050–30062.

Saha, A. and Gopalan, A. (2019). Active ranking with subset-wise preferences. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR.

Saha, A. and Gopalan, A. (2020). From pac to instance-optimal sample complexity in the plackett-luce model. In *International Conference on Machine Learning*. PMLR.

Saha, A., Koren, T. and Mansour, Y. (2021a). Adversarial dueling bandits. *ArXiv* **abs/2010.14563**.

Saha, A., Koren, T. and Mansour, Y. (2021b). Adversarial dueling bandits. In *International Conference on Machine Learning*. PMLR.

Saha, A. and Krishnamurthy, A. (2021). Efficient and optimal algorithms for contextual dueling bandits under realizability. In *International Conference on Algorithmic Learning Theory*.

Sakaguchi, K., Bras, R. L., Bhagavatula, C. and Choi, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM* **64** 99–106.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Shah, N. B. and Wainwright, M. J. (2017). Simple, robust and optimal ranking from pairwise comparisons. *The Journal of Machine Learning Research* **18** 7246–7283.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A. et al. (2023). Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585* .

Slivkins, A. et al. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* **12** 1–286.

Sui, Y. and Burdick, J. (2014). Clinical online recommendation with subgroup rank feedback. In *Proceedings of the 8th ACM conference on recommender systems*.

Swamy, G., Dann, C., Kidambi, R., Wu, Z. S. and Agarwal, A. (2024). A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056* .

Szörényi, B., Busa-Fekete, R., Paul, A. and Hüllermeier, E. (2015). Online rank elicitation for plackett-luce: A dueling bandits approach. In *Advances in Neural Information Processing Systems*.

Thurstone, L. (1927a). A law of comparative judgment. *Psychological Review* **34** 273.

Thurstone, L. L. (1927b). A law of comparative judgment. *Psychological Review* **34** 273–286.

Tversky, A. (1969). Intransitivity of preferences. *Psychological review* **76** 31.

Urvoy, T., Clérot, F., Féraud, R. and Naamane, S. (2013). Generic exploration and k-armed voting bandits. In *ICML*.

Valcarce, D., Parapar, J. and Barreiro, Á. (2017). Combining top-n recommenders with metasearch algorithms. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* .

Vasile, F., Smirnova, E. and Conneau, A. (2016). Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM conference on recommender systems*.

Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B. and Lee (2018). Billion-scale commodity embedding for e-commerce recommendation in alibaba. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* .

Wang, Y., Liu, Q. and Jin, C. (2024). Is rlhf more difficult than standard rl? a theoretical perspective. *Advances in Neural Information Processing Systems* **36**.

Weng, R. C. and Lin, C.-J. (2011). A bayesian approximation method for online ranking. *Journal of Machine Learning Research* **12**.

Wu, H. and Liu, X. (2016). Double thompson sampling for dueling bandits. *ArXiv* **abs/1604.07101**.

Wu, Y., Jin, T., Di, Q., Lou, H., Farnoud, F. and Gu, Q. (2023). Borda regret minimization for generalized linear dueling bandits. In *ICML 2023 Workshop The Many Facets of Preference-Based Learning*.

Wu, Y., Jin, T., Lou, H., Xu, P., Farnoud, F. and Gu, Q. (2022). Adaptive sampling for heterogeneous rank aggregation from noisy pairwise comparisons. In *International Conference on Artificial Intelligence and Statistics*. PMLR.

Xiong, W., Dong, H., Ye, C., Zhong, H., Jiang, N. and Zhang, T. (2023). Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456* .

Xu, J., Lee, A., Sukhbaatar, S. and Weston, J. (2023). Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682* .

Ye, C., Xiong, W., Zhang, Y., Jiang, N. and Zhang, T. (2024). A theoretical analysis of nash learning from human feedback under general kl-regularized preference. *arXiv preprint arXiv:2402.07314* .

Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J. and Weston, J. (2024). Self-rewarding language models. *arXiv preprint arXiv:2401.10020* .

Yue, Y., Broder, J., Kleinberg, R. D. and Joachims, T. (2012). The k-armed dueling bandits problem. *J. Comput. Syst. Sci.* **78** 1538–1556.

Yue, Y. and Joachims, T. (2009). Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*.

Yue, Y. and Joachims, T. (2011a). Beat the mean bandit. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer.

Yue, Y. and Joachims, T. (2011b). Beat the mean bandit. In *International Conference on Machine Learning*.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A. and Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830* .

Zhang, X., Li, G. and Feng, J. (2016). Crowdsourced top-k algorithms: An experimental evaluation. *Proc. VLDB Endow.* **9** 612–623.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M. and Liu, P. J. (2023). Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425* .

Zhao, Z. and Xia, L. (2019). Learning mixtures of plackett-luce models from structured partial orders. In *NeurIPS*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* **36**.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. et al. (2024). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* **36**.

Zhu, B., Jiao, J. and Jordan, M. I. (2023). Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270* .

Zhu, J., Ahmed, A. and Xing, E. P. (2012). Medlda: maximum margin supervised topic models. *J. Mach. Learn. Res.* **13** 2237–2278.

Zoghi, M., Karnin, Z. S., Whiteson, S. and De Rijke, M. (2015a). Copeland dueling bandits. *Advances in neural information processing systems* **28**.

Zoghi, M., Karnin, Z. S., Whiteson, S. and de Rijke, M. (2015b). Copeland dueling bandits. In *NIPS*.

Zoghi, M., Whiteson, S., Munos, R. and de Rijke, M. (2014a). Relative upper confidence bound for the k-armed dueling bandit problem. *ArXiv* **abs/1312.3393**.

Zoghi, M., Whiteson, S., Munos, R. and Rijke, M. (2014b). Relative upper confidence bound for the k-armed dueling bandit problem. In *International conference on machine learning*. PMLR.