Learning Multinomial Logits in $O(n \log n)$ Time

Flavio Chierichetti
Reddit, San Francisco, CA, USA

Mirko Giacchini
Department of Computer Science, Sapienza University of Rome, Italy

Ravi Kumar
Google Research, Mountain View, CA, USA

Silvio Lattanzi
Google Research, Barcelona, Spain

Alessandro Panconesi
Department of Computer Science, Sapienza University of Rome, Italy

Erasmo Tani
Department of Computer Science, Sapienza University of Rome, Italy

Andrew Tomkins
Google Research, Mountain View, CA, USA

Abstract

A Multinomial Logit (MNL) model is composed of a finite universe of items $[n] = \{1, \ldots, n\}$, each assigned a positive weight. A query specifies an admissible subset – called a *slate* – and the model chooses one item from that slate with probability proportional to its weight. This query model is also known as the *Plackett–Luce model* or *conditional sampling* oracle in the literature. Although MNLs have been studied extensively, a basic computational question remains open: given query access to slates, how efficiently can we learn weights so that, for *every* slate, the induced choice distribution is within total variation distance ε of the ground truth? This question is central to MNL learning and has direct implications for modern recommender system interfaces.

We provide two algorithms for this task, one with adaptive queries and one with non-adaptive queries. Each algorithm outputs an MNL $\hat{M}$ that induces, for each slate S, a distribution $\hat{M}_S$ on S that is within ε total variation distance of the true distribution. Our adaptive algorithm makes $O\left(\frac{n}{\varepsilon^3} \log n\right)$ queries, while our non-adaptive algorithm makes $O\left(\frac{n^2}{\varepsilon^3} \log n \log \frac{n}{\varepsilon}\right)$ queries. Both algorithms query only slates of size two and run in time proportional to their query complexity.

We complement these upper bounds with lower bounds of $\Omega\left(\frac{n}{\varepsilon^2} \log n\right)$ for adaptive queries and $\Omega\left(\frac{n^2}{\varepsilon^2} \log n\right)$ for non-adaptive queries, thus proving that our adaptive algorithm is optimal in its dependence on the support size n, while the non-adaptive one is tight within a $\log n$ factor.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Sample complexity and generalization bounds; Theory of computation → Theory and algorithms for application domains

Keywords and phrases Multinomial Logits, Conditional Samples, Discrete Choice Models, Recommender Systems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.63

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2601.04423

Funding *Flavio Chierichetti*: Work done in part while at Sapienza University of Rome. Supported in part by BiCi – Bertinoro international Center for informatics, by a Google Focused Research Award and by the PRIN project 20229BCXNW (funded by the European Union - Next Generation EU, Mission 4 Component 1 CUP B53D23012910006).
Alessandro Panconesi: Supported in part by BiCi – Bertinoro international Center for informatics and by a Google Focused Research Award.

Acknowledgements We wish to thank the anonymous reviewers for helpful feedback.

1 Introduction

Multinomial Logit models (MNLs), also known as softmax or Plackett–Luce models, are widely used to model choice behavior in machine learning and economics. They describe winning distributions over alternatives parameterized by item weights: given a universe U of items, an MNL M assigns each $i \in U$ a weight w_i, and for any non-empty subset $S \subseteq U$, defines $M_S(i) = w_i / \sum_{j \in S} w_j$ as the probability of selecting i from S. Such models underlie diverse applications, from token prediction in large language models to content selection in recommender systems, where they capture how preferences depend on the available slate.

Most prior work focuses on estimating MNL parameters or the induced distribution on the universal slate $S = U$, which suffices for identifying the globally most preferred items or obtaining a top-k ranking. In contrast, we address the more challenging task of approximating the MNL distribution for *all* slates, motivated by practical needs in modern recommender systems. Consider a platform such as Netflix offering choices of movies. It is now broadly understood that simply displaying a very long list of top titles does not provide a compelling user experience. Instead, these platforms define a large and rapidly changing number of relevant subsets of the entire movie catalog: action movies, foreign movies, movies similar to a particular anchor movie the user recently watched, and so forth. The interface then shows a sequence of carousels, perhaps a carousel of "top movies" followed by "movies similar to Ponyo" then "new arrivals", each one ordered to show the user's best options from the class. To drive such an interface, it is important to approximate the winning distribution for *every* one of these subsets simultaneously, to be ready to display it when needed. As the possible subsets of interest are constantly updated by the platform, it is critical to approximate the MNL's output on all possible subsets $S \subseteq U$.

Furthermore, for a particular subset, such as that containing all action movies, the platform will not show a single option, but will instead show a carousel with a moderate number of suggestions. While some previous work focused exclusively on ranking the items, practical recommender systems require scoring them for at least two key reasons. First, the number of items shown should depend on their scores: if there are four high-scoring movies, it might be better to only display those, rather than adding the next six, which may have very little chance of being selected. Second, the interface might have more richness than just the carousel itself. For instance, if the top movie of a carousel has a much higher score than the next one, the platform might feature this movie more prominently, for example, by using a specialized rendering or by allocating more space to it. Hence, it is crucial to obtain estimates of the weights that provide accurate winning distributions on all slates.

1.1 MNL and MNL Learning

A *multinomial logit (MNL) model* supported on the universe $U = [n] = \{1, \dots, n\}$ is specified by a set $\{w_1, \dots, w_n\}$ of n positive values called *weights*. A *slate* is a non-empty subset of $[n]$. An MNL M, for any given slate $S \subseteq [n]$, induces a conditional distribution denoted M_S whose support is S and where the probability of each item $i \in S$ is given by:[1]

$$M_S(i) = \frac{w_i}{\sum_{j \in S} w_j}.$$

[1] The terminology we adopt comes from the Economics literature [42], this is called a *logit* model because if we let $w_i = e^{\theta_i}$, then $M_S(i) = \mathrm{softmax}(S)_i = e^{\theta_i} / \sum_{j \in S} e^{\theta_j}$.

An MNL M can be accessed by a Sample oracle, which operates as follows: given a slate S, Sample(S) returns $i \in S$ chosen according to the distribution M_S. Given MNLs M and M', we define two notions of distance between them:

$$d_\infty(M, M') := \max_{\substack{S \subseteq [n] \\ S \neq \emptyset}} \|M_S - M'_S\|_\infty \quad \text{and} \quad d_1(M, M') := \max_{\substack{S \subseteq [n] \\ S \neq \emptyset}} \|M_S - M'_S\|_1.$$

In this paper we obtain algorithms that approximate an unknown MNL M in d_1 distance, while our lower bounds apply even to the less challenging problem of obtaining estimates with small d_∞ distance.

▶ **Definition 1** (MNL Learning Problem). *Given* Sample *oracle access to an MNL M and an $\varepsilon \in (0, 1)$, the* MNL learning problem *is to output an MNL $\hat{M}$ such that $d_1(M, \hat{M}) \leq \varepsilon$. The MNL produced in output is represented using the logarithms of its weights.*[2]

1.2 Main Results

In this paper, we study algorithms for the MNL learning problem. We obtain two algorithms, one using adaptive queries and the other using non-adaptive queries. Both algorithms query only slates of size two and run in time proportional to their query complexity. Our adaptive algorithm makes $O\left(\frac{n}{\varepsilon^3} \log n\right)$ queries; we give a lower bound of $\Omega(\frac{n}{\varepsilon^2} \log n)$ queries. Summarizing:

▶ **Theorem 2** (Informal). *For any constant $\varepsilon > 0$, the complexity of learning an MNL within d_1-error ε by making* Sample *queries adaptively is $\Theta(n \log n)$.*

Our non-adaptive algorithm makes $O\left(\frac{n^2}{\varepsilon^3} \log n \log \frac{n}{\varepsilon}\right)$ queries; this is complemented by a lower bound of $\Omega(\frac{n^2}{\varepsilon^2} \log n)$. Summarizing:

▶ **Theorem 3** (Informal). *For any constant $\varepsilon > 0$, the complexity of learning an MNL within d_1-error ε by making* Sample *queries non-adaptively is between $O(n^2 \log^2 n)$ and $\Omega(n^2 \log n)$.*

As we mentioned above, the lower bounds described also hold for the weaker d_∞ distance.

Our results are surprising: for a constant ε, our seemingly harder problem can be solved as fast as (noisy) sorting. Furthermore, our lower bounds hold for unit-time oracle queries of any slate size. Hence, restricting the algorithms to slates of size two incurs no loss in efficiency.

1.3 Technical Challenges

Existing methods, especially ones that approximate the winning distribution over the universal slate $[n]$, do not seem to apply to our problem. As a simple example of the difficulty, consider an algorithm that guarantees an ℓ_1-estimate of the full slate distribution within an error of $\varepsilon \in (0, 1/2)$. Consider now the MNL on $\{1, 2, 3\}$ with weights $w_1 = 1 - \varepsilon$, $w_2 = w_3 = \varepsilon/2$. Suppose the algorithm returns the estimate $\hat{w}_1 = 1 - \varepsilon$, $\hat{w}_2 = \frac{3\varepsilon}{4}$, $\hat{w}_3 = \frac{\varepsilon}{4}$;

[2] Representing an MNL using the logarithms of its weights is standard in the ML and economics community [37, 42]. Moreover, with a full representation of the $\hat{w}_i$'s, the weights could require $\Omega(n^2)$ bits just to be stored. For instance, consider the MNL on $[n]$ with weights $w_i = 2^i$. Since $\frac{w_{i+1}}{w_i + w_{i+1}} = \frac{2}{3}$, any MNL $\hat{M}$ solving the MNL learning problem must satisfy $\hat{w}_{i+1} \geq \hat{w}_i \cdot (2 - 9\varepsilon)$. Therefore, each weight $\{\hat{w}_{n/2}, \ldots, \hat{w}_n\}$ requires $\Omega(n)$ bits for a total of $\Omega(n^2)$ bits. Hence, requiring that an algorithm outputs the weights, rather than their logarithms, would rule out the possibility of constructing any algorithm that runs in time $o(n^2)$. Other compact representations of the weights are also possible.

clearly, $\|w - \hat{w}\|_1 = \frac{\varepsilon}{2} \leq \varepsilon$. But, $\left| \frac{w_2}{w_2+w_3} - \frac{\hat{w}_2}{\hat{w}_2+\hat{w}_3} \right| \geq \frac{1}{4}$, and therefore the algorithm cannot guarantee small error on the slate $\{2, 3\}$. Similarly, prior work that additively estimates the winning distributions on all size-two slates cannot be used to obtain a good approximation on every slate. Moreover, noisy sorting algorithms also do not imply our guarantees. Indeed, while our final algorithm will perform a noisy sorting step at the beginning, the key challenge lays in performing the additional comparisons that allow the accurate estimate of each conditional distribution, as we detail below.

It is not difficult to obtain a cubic time algorithm for our problem. Indeed, consider the naive algorithm that works as follows. For each pair $\{i, j\}$ of items in the universe, estimate w_i/w_j to within a $(1 \pm \varepsilon)$-multiplicative error, or declare that their ratio (or its inverse) is larger than $\frac{n}{\varepsilon}$. One can easily show that this algorithm will need to query each pair $\approx \frac{n \log n}{\varepsilon^3}$ times to guarantee these bounds; the total cost would then be $\approx \frac{n^3 \log n}{\varepsilon^3}$. From the output of this algorithm, it is easy to approximate the output distribution for any slate.[3]

With some effort, this algorithm can be improved. The idea is to carefully control the pairs of items, querying only pairs that are nearby in the order induced by the weights (which can be approximately inferred via a noisy sorting algorithm). To avoid querying too many nearby pairs, one can first cluster items whose weights are within a constant factor, in a query-efficient manner, and then select a center from each cluster. One can determine the weight ratio of each item to its cluster center, and the weight ratios of successive items in the sorted list of cluster centers. This method can be shown to produce an algorithm that compares $O(n)$ pairs of items, with each such comparison performing $\approx \frac{n}{\varepsilon^3} \log^3 n$ queries. While this yields a quadratic time algorithm, it is unclear how this can be further improved to being quasi-linear.

1.4 Overview of Methods

We construct our quasi-linear adaptive algorithm by building on the clustering idea described above. We first partition the universe into clusters of similar-weight items and select a center for each cluster. We then estimate the ratio of the weights w_i/w_c for every item i in the cluster with center c. Finally, we construct a forest on the cluster centers, where every edge is labeled with an estimate of the ratio between the weights of the two centers it connects. We call this data structure the *estimation-forest*. This allows us to obtain estimates of the ratio of the weights for arbitrary pairs of items by combining these estimates along paths in the forest.

Following this strategy, the error compounds multiplicatively along the paths. To circumvent this issue without requiring more accurate ratio estimates – which would lead to a higher complexity – we design the forest so that any two centers whose weights the algorithm might want to compare are at a short distance from each other. To achieve this property, the topology of the forest is constructed adaptively.

Additionally, to further improve the sample complexity (and runtime), we dynamically adjust the number of queries required to approximate the weight ratio between two centers. In particular, if the total weight of items lighter than a given item i is not large enough with respect to the weight of i, then it becomes unimportant to estimate the ratio of the weight of i over the weight of any of these items.

[3] Indeed, if the items of this slate have weights that are within a $\frac{n}{\varepsilon}$ factor of each other, the $(1 \pm \varepsilon)$-approximation error will make it possible to approximate the winning probability of any item to within a $(1 \pm O(\varepsilon))$-factor (so that the total variation error will be at most $O(\varepsilon)$). If, instead, the slate contains pairs $\{i, j\}$ of items such that $w_i/w_j < \frac{\varepsilon}{n}$, then the lighter item i will have a probability of winning in the slate not larger than $O(\frac{\varepsilon}{n})$, and hence we can estimate its winning probability to be zero – given that there are at most $n - 1$ such light items in a slate, the total variation error is no larger than $O(\varepsilon)$.

Our estimation-forest data structure dynamically determines query sequences for weight-ratio estimation and enables all cluster-center–cluster-center and cluster-item–cluster-center comparisons in $O\left(\frac{n}{\varepsilon^3}\log n\right)$ queries.

While the above algorithm is adaptive, we also obtain a non-adaptive version. The idea is to first design a new adaptive algorithm that queries each pair of items only $O\left(\frac{1}{\varepsilon^3}\log n \log\frac{n}{\varepsilon}\right)$ times. We then query every pair a fixed number of times and then simulate this new adaptive algorithm on the precomputed answers; this leads to a non-adaptive algorithm with $O\left(\frac{n^2}{\varepsilon^3}\log n\log\frac{n}{\varepsilon}\right)$ queries.

The two lower bounds in our paper are proved using a reduction from the problem of identifying the k coins with the highest heads probability in a collection of n biased coins.

In particular, we construct an MNL supported on an even-sized universe whose items are divided into pairs, each pair representing the two sides of a coin. We then order the pairs so that the weights of the items in a pair are much larger than those in preceding pairs. This way, we can assume without loss of generality that any learning algorithm is only querying slates corresponding to our original pairs.

1.5 Organization

In Section 2 we review related work. Section 3 introduces key tools and notation that we use throughout the paper. Section 4 gives an overview of our adaptive algorithm, while Section 5 gives an overview of our non-adaptive algorithm. Section 6 gives an overview of our adaptive and non-adaptive lower bounds. Finally, in Section 7 and Section 8, we conclude with some open questions.

All the proofs and algorithms missing from the paper can be found in the full version.

2 Related Work

The problem of learning MNLs on *all* slates from Sample queries arises naturally from several perspectives. Our work is related to, yet distinct from, the existing literature. First, prior work on MNL fitting has provided approximation guarantees only for the full slate or for pairs of items – both of which are strictly weaker than the guarantees we obtain. Second, our framework extends beyond classical MNL ranking and selection by capturing quantitative relationships among items, revealing how much and where certain items dominate, while preserving the $O(n\log n)$ efficiency of the best known ranking algorithms. Third, our problem can be viewed as a natural strengthening of distribution learning under conditional sampling, extending the "testing by learning" paradigm to recover all conditional distributions simultaneously in a more expressive and challenging setting. Finally, our results also strengthen prior work on learning Random Utility Models (RUMs), specialized to the MNL case. We elaborate on these connections below.

2.1 MNL Fitting

A large body of the literature focuses on finding MNL weights maximizing the likelihood of a collected dataset. In this setting, usually, the queries are either fixed [45, 19, 15, 31] or sampled from a distribution [32, 29, 30, 27]. When the dataset actually comes from a hidden MNL model, some of these algorithms guarantee that the estimated (normalized) weights approximate the hidden (normalized) weights [29, 30, 27, 37, 39, 36].

These works are not directly applicable to our setting because of the following two main issues.

(i) Approximately recovering the normalized weights is equivalent to providing a good estimate of the winning distribution of the full slate. However, this is insufficient to accurately estimate the winning distribution for smaller slates, as we discussed in the Introduction.

(ii) Most of these works assume that the maximum ratio between two weights is upper bounded by a constant [29]. Note that such an assumption would greatly simplify our problem given that we could accurately estimate the weights with respect to any anchor item. Therefore, the interesting setting is one where there is no *a priori* bound on the ratio of the weights. Furthermore, as these algorithms are non-adaptive, they are subject to our lower bound of $\Omega(n^2 \log n)$ queries for our problem.

Stepping outside the task of fitting the weights themselves, Falahatgar et al. [17] adaptively query $O\left(\frac{n \cdot \log(n) \cdot \min\{n, 1/\varepsilon\}}{\varepsilon^2}\right)$ slates of size two (i.e., pairs) and produce an *additive* estimate within ε for all other *pairs*, in a family of models that are more powerful than MNLs [44]. However, in order for an additive approximation of the slates of size two to generalize to all other slates with an ℓ_∞-error of ε', one needs $\varepsilon \leq \frac{\varepsilon'}{n}$. Therefore, applying their algorithm as a blackbox would require $\Omega(n^4 \log n)$ queries. Moreover, their algorithm heavily relies on providing an additive approximation – specifically, each estimated probability is rounded to the closest multiple of ε. Hence, it appears hard to generalize their work to larger slates even in a non-blackbox manner.

Other works considered minimizing the ℓ_∞-difference of the logarithm of the weights, or approximating the weight ratios of all the pairs [43, 20, 40, 24]. These guarantees would be sufficient to obtain a good estimate for all slates, but are not necessary. Indeed, if $w_1 = e^{1000}$, $w_2 = e^{\varepsilon}$ and $\hat{w}_1 = e^{100}$, $\hat{w}_2 = e^{\varepsilon}$, then $\|\ln(w) - \ln(\hat{w})\|_\infty$ can be made arbitrarily large by increasing the constant 1000, yet, $\hat{w}$ provides good predictions: $|\frac{w_1}{w_1+w_2} - \frac{\hat{w}_1}{\hat{w}_1+\hat{w}_2}| \leq \varepsilon$. Moreover, these works analyze non-adaptive algorithms and assume a bound on the ratio of the weights, and therefore suffer our $\Omega(n^2 \log n)$ lower bound.

We also mention a separate line of work that focuses on developing statistical tests to determine whether a given dataset of comparisons is consistent with an MNL model [38, 25, 34]. These works are complementary to ours in that they address model validation rather than estimation, and they do not provide algorithms for learning the underlying MNL parameters.

2.2 MNL Ranking/Selection

Other classical problems involving MNLs include: (i) sorting/ranking the weights [18, 17, 9, 41, 35], (ii) finding the top-k items with largest weight [21, 10, 11, 12, 22], and (iii) finding the item of maximum weight [16, 26]. Some works make assumptions about the weights (e.g., adjacent weights are sufficiently separated) and seek an exact output with high probability [21], while others do not make further assumptions but only require a probably approximately correct (PAC) output [41]. These problems have also been explored in the "dueling bandits" literature [3].

While we will use an $O(\frac{n \log n}{\varepsilon^2})$ approximate sorting algorithm by Falahatgar et al. [17] as a first step in our algorithm, these results are not sufficient by themselves to learn an MNL under our definition. Our lower bound will be proved by showing that the top-$\frac{n}{2}$ problem (and the ranking problem) reduces to our MNL learning problem. Interestingly, despite this, we obtain an $O(\frac{n \log n}{\varepsilon^3})$ learning algorithm that is only $O(\frac{1}{\varepsilon})$-factor worse than the best possible algorithm for MNL ranking [17].

2.3 Distribution Testing with Conditional Samples

Our problem can also be described in the context of conditional sampling. Let μ be a hidden distribution over $[n]$. Algorithms can, adaptively, make the following types of queries: chosen a set $S \subseteq [n]$, an oracle returns an item of S sampled according to distribution μ conditioned on S.[4] The goal in distribution testing is usually to make the smallest number of queries to establish whether μ satisfies certain properties, such as, e.g., uniformity [5]. However, the problem of estimating the probability $\mu(i)$, for $i \in [n]$, has also been considered [8, 6, 2].

Our problem, on the other hand, asks for the minimum number of queries to accurately estimate $\mu(i \mid S)$ for each $i \in S \subseteq [n]$.[5] Indeed, the distribution μ can be seen as the weights of an MNL M and therefore, $\mu(i \mid S) = M_S(i)$ for $i \in S \subseteq [n]$. Observe that having an estimate only for $\mu(i)$ is equivalent to an estimate of the winning distribution on the full slate – insufficient to estimate the winning distribution for smaller slates. Some algorithms provide a multiplicative $(1 \pm \varepsilon)$ estimate for $\mu(i)$ for $i \notin B$ where B is a set such that $\sum_{b \in B} \mu(b) \leq \varepsilon$; this is a stronger property than an additive approximation of the full slate. However, it still cannot provide accurate estimates for slates that are either subsets of B or that span across B and $[n] \setminus B$. Note also that it can be $|B| = \Theta(n)$ meaning that the distribution of most slates cannot be estimated. From a technical standpoint, Chakraborty et al. [8] achieve this guarantee by building a complete binary tree where edges are labeled with probabilities. This idea bears some high level similarities with our estimation-forest, but details differ. Indeed, their tree is static while the topology of our forest is adaptively chosen, which is crucial for a tight $O(n \log n)$ bound. Moreover, their tree is populated by querying slates of arbitrary size, while we only query slates of size two. Finally, as argued above, the guarantees provided by their tree are insufficient to estimate the winning distributions of all slates. Recent work has also focused on different query models [1, 33, 28]; however, their results are incomparable to ours.

We remark that a common paradigm for designing distribution testing algorithms in the traditional (unconditional) setting is that of *testing by learning* (see, e.g. [7]), in which a property is tested by first approximately learning the underlying distribution, and then checking whether the learned distribution has the property in question. Our work serves as a conditional counterpart to this paradigm that works for the more challenging case in which the property being tested requires approximating the behavior of all conditional distributions.

2.4 RUM Learning

MNLs are a special case of RUMs; hence, algorithms for learning RUMs on all slates could be used to learn an MNL. However, the best known algorithms for general RUM learning require exponentially many queries to slates of size $\Theta(\sqrt{n})$ [13]. In contrast, we show that MNLs can be learned using only $O(n \log n)$ queries to slates of size two.

3 Technical Preliminaries

Let $U = [n] = \{1, \ldots, n\}$ be a universe of items. For a probability distribution P over $[n]$, let $P(i)$ denote the probability of the item $i \in [n]$. For distributions P, Q, let $\|P - Q\|_1 := \sum_{i \in [n]} |P(i) - Q(i)|$ be the ℓ_1-*distance*, which is also twice the total variation distance, and let $\|P - Q\|_\infty := \max_{i \in [n]} |P(i) - Q(i)|$ be the ℓ_∞-*distance*. Let $X \sim \mathrm{Ber}(\mu)$ denote a random

[4] If S has probability 0, the oracle returns a uniform at random item from S.
[5] In our proofs, we will assume that the weights are strictly positive for simplicity. However, the same algorithms also work when weights of zero are allowed.

variable following a Bernoulli distribution with mean μ and let $X \sim \mathrm{Bin}(n,p)$ denote a random variable following a binomial distribution with n trials and head probability p. Also let $X \sim \mathrm{Geom}(p)$ denote a random variable following a geometric distribution with parameter $p \in (0,1]$; in particular, $\Pr_{X\sim\mathrm{Geom}(p)}[X=k] = (1-p)^{k-1}p$, for $k \geq 1$. For any $x, y \in \mathbb{R}$, we denote by $x \pm y$ the interval $[x-y, x+y]$ and for any $x \in \mathbb{R}, \varepsilon \in (0,1)$, we denote by $(1\pm\varepsilon)x$ the interval $[(1-\varepsilon)x, (1+\varepsilon)x]$. We will use standard concentration results (see, e.g., [14, 4]).

3.1 Ordered Clusterings and Directed Weightings

An *ordered clustering* of $[n]$ is given by an ordered partition $(C_1, \ldots, C_T)$ of $[n]$ and a corresponding list $(c_1, \ldots, c_T)$ of *centers* such that $c_i \in C_i$ for each i. In our algorithms, the centers can always be thought to be sorted in increasing order of weight. Here, for $v \in [n]$, let $\gamma(v) \in [T]$ the unique index such that $v \in C_{\gamma(v)}$; we call $\gamma(v)$ the *cluster index* of v.

Let $F = ([n], E)$ be an undirected forest supported on $[n]$. For $u, v \in [n]$ in the same connected component of F, let $P(u,v)$ be the (unique) path in F from u to v. We use $d(u,v)$ to denote the (unweighted/hop) distance in F between vertices u and v, where if u and v are in different connected components, we define $d(u,v) = \infty$.

Let $\vec{E} := \{(u,v) \in V^2 \mid \{u,v\} \in E\}$. A *directed weighting* of the edges of F is a function $r : \vec{E} \to \mathbb{R}_{>0}$ such that $r(u,v) = 1/r(v,u)$. For a path $P = u_1, \ldots, u_t$ in F define $r(P) = \prod_{i=1}^{t-1} r(u_i, u_{i+1})$, and if $t = 1$, let $r(P) = 1$.

4 Learning MNLs Adaptively

Our first result is an algorithm to learn an MNL M by making $O\left(\frac{n}{\varepsilon^3}\log n\right)$ adaptive Sample queries to output the weights of an MNL $\hat{M}$ such that $d_1(M, \hat{M}) \leq \varepsilon$. Observe that $M_S(i) = \frac{w_i}{\sum_{s\in S} w_s} = \frac{1}{\sum_{s\in S} \frac{w_s}{w_i}}$. Therefore, if we had access to a multiplicative estimate of the ratio w_i/w_j for each pair $i, j \in [n]$, we could provide a good estimate for M_S for each slate S, in ℓ_1-error. Unfortunately, this has two issues. (i) In general, this ratio can be unbounded and therefore, producing a multiplicative estimate could in principle cost an unbounded number of queries. (ii) If we aim to obtain an algorithm with query complexity $o(n^2)$, we simply cannot afford to query all the pairs.

To circumvent these issues, we instead construct a *sparse* graph on the items of $[n]$ that contains estimates of the ratio w_i/w_j along each edge $\{i,j\}$, and then use this graph to compute $\hat{M}$. At a high level, we produce a forest F such that: (i) if two items are close to each other in F, we can get an estimate of their ratios, (ii) if two items are far away in F, then their ratio is negligible. We will also need some technical properties to ensure that we can obtain a valid MNL $\hat{M}$ from the forest. The following definition formalizes the properties we need.

▶ **Definition 4** ((t,ε)-Estimation-Forest). *Let $t \in \mathbb{Z}, t \geq 2$ and let $\varepsilon \in (0,1)$. A (t,ε)-estimation-forest for an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$ is a tuple $\mathcal{F} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$, where $F = ([n], E)$ is an undirected forest, r is a directed weighting on F, and $(C_1, \ldots, C_T), (c_1, \ldots, c_T)$ is an ordered clustering over $[n]$. For any $u, v \in [n]$ such that $\gamma(u) \geq \gamma(v)$:*

1. *if $d(u,v) \leq t$, then $r(P(u,v)) \in (1\pm\varepsilon)\cdot\frac{w_u}{w_v}$ and $r(P(v,u)) \in (1\pm\varepsilon)\cdot\frac{w_v}{w_u}$.*
2. *If $d(u,v) \in (t,\infty)$, then:*

$$\sum_{\substack{s\in[n]\\ \gamma(s)\leq\gamma(v)}} \frac{w_s}{w_u} \leq \varepsilon \quad \text{and} \quad \sum_{\substack{s\in\mathcal{C}\\ \gamma(s)\leq\gamma(v)}} r(P(s,u)) \leq \varepsilon,$$

where $\mathcal{C}$ is the connected component containing both u and v.

3. *if* $d(u,v) = \infty$, *then:*

$$\sum_{\substack{s\in[n]\\ \gamma(s)\le\gamma(v)}} \frac{w_s}{w_u} \le \varepsilon.$$

   Also, for any u' *(resp.* v'*) in the same connected component of* u *(resp.* v*), it holds that* $\gamma(u') > \gamma(v')$.

4. *if* $\gamma(u) = \gamma(v)$, *then* $d(u,v) \le t$.

The ordering of the centers can intuitively be thought to be in increasing order of weight. In the full version, we show that we can use a (t,ε)-estimation-forest for an MNL M to obtain an MNL $\hat{M}$ such that $d_1(M,\hat{M}) \le O(\varepsilon)$.

4.1 On Choosing the Estimation-Forest Topology

Interestingly, for the purpose of constructing $\hat{M}$, it turns out that the specific value of t is irrelevant. This observation allows us to reduce the problem of learning M to that of constructing a (t,ε)-estimation-forest for a single, arbitrary choice of t. The central challenge then lies in designing an efficient topology for the estimation-forest.

The most natural topology would be a path on the items (after a noisy-sorting step). However, along a path, two items of comparable weight can be separated by a super-constant distance $d = \omega(1)$. To preserve property 2 of Definition 4, one would then need to construct a (d,ε)-estimation-forest, which would incur a query cost of $\Omega(nd^2 \log n)$. Since d can be as large as $\Theta(n)$, this is clearly suboptimal, suggesting the need for a topology with low diameter. Note that even a complete binary tree also can yield super-constant length paths, implying an $\omega(n \log n)$ query cost.

On the other hand, to achieve a very small diameter, one might consider a star or a tree topology with unbounded arity. However, in these cases, one would need to estimate extremely large weight ratios, leading to high query complexity. This, in fact, explains why a disconnected graph is required.

Another natural direction would be to consider general (non-acyclic) graphs. In fact, we could consider a path with skips to decrease the diameter (perhaps exploiting modern shortcutting results [23]). The difficulty is that, in a cyclic graph, the weight of an item depends on the particular path chosen, and different paths can yield inconsistent estimates. Thus, acyclicity of the topology is essential to ensure that an explicit MNL can be extracted from it.

In the full version, we provide an efficient algorithm for constructing an $(O(1),\varepsilon)$-estimation-forest. The resulting topology takes the form of a forest of *lobster graphs*: items are first clustered together, as described below, and a forest of unbounded-arity trees is then constructed over the resulting cluster centers. The arity of each tree is not predetermined but is instead adaptively chosen as the algorithm progresses, in order to balance estimation accuracy and query efficiency. Interestingly, the diameter of the trees in our forest can be super-constant. However, each tree will have the property that if two items are at distance more than $O(1)$, then one of the two is so much larger than the other that their ratio can be taken to be infinite without incurring a large error. Thanks to this property, from the perspective of any single item, one can consider the tree to have constant diameter and lose at most $O(\varepsilon)$ in the final estimate.

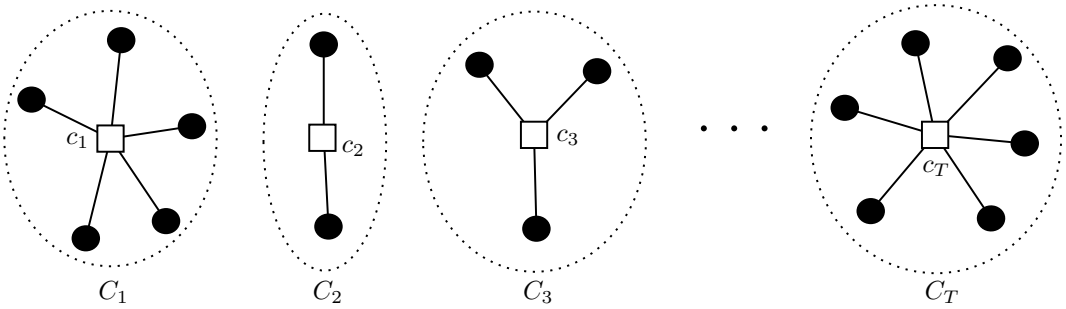

Figure 1 The structure of an (A_1, A_2, ε)-cluster graph. The vertices of the graph are the items $[n]$ of the MNL, the cluster centers are depicted as white-filled squares, while the other items are represented by black circles. Items in the same cluster have similar weight (within a factor of A_1 of each other). Clusters further to the right contain items of higher weights. Associated with each edge $\{u, v\}$, and each direction (say, $u \to v$), is an estimate $r(u, v)$ of the ratio w_u/w_v.

4.2 Building the Estimation-Forest

We now go more into the details of our solution to efficiently build an estimation-forest. When constructing the estimation-forest, some ratio estimates might be costlier to obtain than others. In order to maintain a low query complexity, we leverage the fact that if two items have similar weights, fewer queries are required to estimate the ratio of their weights. In the first step to build our estimation-forest, we exploit this observation via a pre-processing step, which sorts the items in approximately increasing order of weights, and produces clusters of similar items resulting in a *cluster graph*, defined as follows.

▶ **Definition 5** (Cluster Graph). *An (A_1, A_2, ε)-cluster graph for an MNL supported on $[n]$ with weights $\{w_1, \ldots, w_n\}$, is a tuple $\mathcal{G} = (F, r, (C_1, \ldots, C_T), (c_1, \ldots, c_T))$, where $F = ([n], E)$ is an undirected forest, r is a directed weighting on F, and $(C_1, \ldots, C_T), (c_1, \ldots, c_T)$ is an ordered clustering over $[n]$, satisfying:*

1. *For any $i \in [T]$ and any item u in the cluster C_i we have:*

$$\frac{1}{A_1} \leq \frac{w_u}{w_{c_i}} \leq A_1.$$

2. *For any $i, j \in [T]$ with $i > j$ we have:*

$$\frac{w_{c_i}}{w_{c_j}} \geq A_2.$$

3. *The edge set E consists of all the edges of the form $\{c_i, u\}$ for all choices of i and of $u \in C_i$. Moreover the weight $r(u, v)$ of any edge $\{u, v\} \in E$ satisfies:*

$$r(u, v) \in (1 \pm \varepsilon)\frac{w_u}{w_v} \quad \text{and} \quad r(v, u) = \frac{1}{r(u, v)} \in (1 \pm \varepsilon)\frac{w_v}{w_u}.$$

In the full version we provide an algorithm to efficiently build a cluster graph. Our algorithms employs a noisy sorting procedure of Falahatgar et al. [17] as a subroutine and builds on it to partition the vertices and compute the edge weights r. We show in Figure 1 a cluster graph produced by our algorithm.

Observe that a cluster graph is not yet an estimation-forest. Indeed, there might be items in different clusters (but close in the ordering) whose ratio is constant. To obtain an estimation-forest, we add extra edges between some pairs of centers. We do so in an

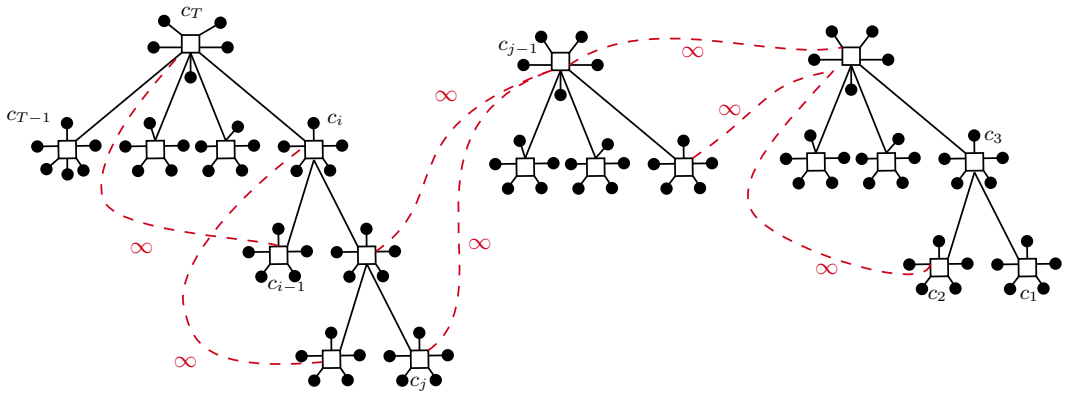

Figure 2 The structure of an estimation-forest constructed by our adaptive algorithm. White squares represent cluster centers, while black circles represent the other items of $[n]$. A new level in the forest is created when two nodes are compared and the estimate of their ratio is "∞". If this happens twice consecutively (for the parent node and the children with smallest estimated weight), then a new tree is created. In the figure, we have $i < T - 1$ and $j < i - 1$.

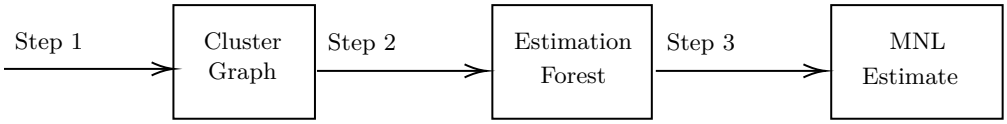

Figure 3 The structure of our algorithm to learn MNLs adaptively. The non-adaptive algorithm follows the same overall structure.

iterative way, starting from the center of the last cluster and moving backwards. *A priori*, these multiplicative estimates can potentially be costly to obtain, since the ratio between the weights of distinct cluster centers could be arbitrarily large. In order to maintain a low query complexity, we employ a careful thresholding strategy. This ensures that we only require an accurate estimate of the ratio when this is not too large to make a significant difference in the MNL winning distributions. For instance, if the ratio between two items is greater than $\Omega(\frac{n}{\varepsilon})$, then it is safe to act as if the second item's weight is infinitely larger than the first, as this approximation only causes a d_1-error of magnitude $O(\varepsilon)$. When we find two clusters that are incomparable, we restart the iteration process from the last cluster that was comparable. It can be shown that this leads to an $(O(1), \varepsilon)$-estimation-forest. We show in Figure 2 a forest that can be produced by our algorithm.

In summary, our algorithm consists of *three* phases. In the first phase, we construct a $(\Theta(1), \Theta(1), \Theta(\varepsilon))$-cluster graph. In the second phase, we extend the cluster graph to a $(\Theta(1), \Theta(\varepsilon))$-estimation-forest. Finally, in the third phase, we use the forest to recover an estimate of the MNL weights. A representation of the steps in our algorithm is in Figure 3. The first two phases require at most $O(\frac{n \log n}{\varepsilon^3})$ queries, while the last one does not make any further queries, yielding our main result:

▶ **Theorem 6.** *Choose any $\varepsilon \in (0, 1)$ and $\delta = n^{-c}$ for a constant $c > 0$. There exists an adaptive randomized algorithm that, with probability at least $1 - \delta$, makes $O\left(\frac{n \log n}{\varepsilon^3}\right)$ Sample queries and solves the MNL Learning Problem on $[n]$ with accuracy parameter ε. Moreover, the algorithm only queries pairs and runs in time proportional to the number of queries.*

5 Learning MNLs Non-Adaptively

We next present an algorithm to learn MNLs non-adaptively, i.e., by making a single batch of queries. In order to do this we leverage the following reduction.

▶ **Lemma 7.** *Given an adaptive algorithm for learning MNLs with the* Sample *oracle that queries any pair of items at most m times, one can construct a non-adaptive algorithm for the same problem that makes at most $m\binom{n}{2} = O(mn^2)$ queries.*

Proof. The non-adaptive algorithm queries each pair m times and then simulates the adaptive algorithm by replacing each Sample oracle call with a revealed response from the set of non-adaptive queries. ◀

The number of Sample queries made to any pair $\{u, v\} \subseteq [n]$ of items by the adaptive algorithm described above could be as high as $\tilde{O}(n/\varepsilon^3)$; this would naively yield an $\tilde{O}(n^3/\varepsilon^3)$-algorithm. Instead, we design an algorithm with query complexity $\tilde{O}(n^2/\varepsilon^3)$. To accomplish this, we modify the adaptive algorithm to obtain a new (adaptive) algorithm that has a worse overall query complexity than $O(n \log n)$, but allows us to uniformly bound the number of queries made to each pair of items. In particular, we show the following result.

▶ **Theorem 8.** *Choose any $\varepsilon, \delta \in (0, 1)$. There exists an adaptive randomized algorithm that, with probability at least $1 - \delta$, queries each pair at most $O\left(\frac{\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ times and solves the MNL Learning Problem on $[n]$ with accuracy parameter ε.*

To obtain this result, we use three new technical ingredients. First, we make use of a different algorithm to approximately order the items of the MNL. This algorithm, which is a straight-forward adaptation of the classical Quicksort algorithm, makes more queries than the previous one overall, but guarantees a uniform upper bound on the number of queries on each pair of items. Second, we introduce a new algorithm to construct the estimation-forest. This algorithm only needs to make $O(|C_j| \log^2 n)$ comparisons between any pair $\{c_i, c_j\}$ of cluster centers (with $i > j$) whenever the ratio w_{c_i}/w_{c_j} is estimated. Finally, we introduce a subroutine that allows one to amortize the cost of estimating the ratio w_{c_i}/w_{c_j} among all the pairs of the form $\{c_i, s\}$, where s belongs to the cluster C_j. This allows one to distribute the $O(|C_j| \log^2 n)$ cost nearly equally among all items in C_j, and hence to guarantee each pair is queried at most $O(\log^2 n)$ times.

Combining Theorem 8 and Lemma 7 yields:

▶ **Corollary 9.** *Choose any $\varepsilon, \delta \in (0, 1)$. There exist a non-adaptive algorithm that, with probability at least $1 - \delta$, makes at most $O\left(\frac{n^2\cdot\log(n/\varepsilon)\cdot\log(n/\delta)}{\varepsilon^3}\right)$ queries and solves the MNL Learning Problem on $[n]$ with accuracy parameter ε.*

6 Lower Bounds

We prove lower bounds that show that our adaptive algorithm has optimal dependence on n, and that our non-adaptive algorithm has nearly-optimal (at most a $\log n$ factor away from optimal) dependence on n. Moreover, both algorithms are only a factor of $1/\varepsilon$ away from optimal in terms of their dependence on the accuracy parameter ε. We prove lower bounds on the easier task of producing an estimate $\hat{M}$ with $d_\infty(M, \hat{M}) \leq \varepsilon$, and these in turn imply lower bounds on obtaining an approximation in the d_1-distance.

For learning MNLs with adaptive queries to Sample, we show the following.

▶ **Theorem 10.** *Any (possibly randomized and adaptive) algorithm that, given in input* $\varepsilon, \delta \in (0,1)$ *and access to a* Sample *oracle for any MNL* M*, outputs an MNL* $\hat{M}$ *satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \leq \varepsilon] \geq 1 - \delta,$$

must make $\Omega(\frac{n}{\varepsilon^2} \log \frac{n}{\delta})$ *queries in the worst case.*

For the non-adaptive case, we show the following.

▶ **Theorem 11.** *Any (possibly randomized) non-adaptive algorithm that, given in input* $\varepsilon \in (0,1)$ *and access to a* Sample *oracle for any MNL* M*, outputs an MNL* $\hat{M}$ *satisfying:*

$$\Pr[d_\infty(M, \hat{M}) \leq \varepsilon] \geq \frac{9}{10},$$

must make $\Omega(\frac{n^2}{\varepsilon^2} \log n)$ *queries in the worst case.*

Both the lower bounds we provide are based on reductions from the problem of approximately identifying the $\frac{n}{2}$ coins with the largest probability of *heads* in a set of n biased coins.

7 Future Work

In this work we essentially resolved the complexity of learning MNLs via Sample queries in the adaptive setting. Future work could, however, tackle a number of technical improvements. The main question we leave open is finding the optimal dependence on ε for adaptive algorithms. We highlight here some challenges in obtaining an algorithm with a better dependence in ε.

First, we observe that the analysis of our $O(\frac{n \log n}{\varepsilon^3})$ algorithm is tight. Our algorithm constructs a forest with vertex set equal to the items, and each edge (a, b) in the forest is labeled with a $(1 \pm \varepsilon)$-estimate of the ratio w_a/w_b. It can be shown that the topology of the forest can be obtained with $O(\frac{n \log n}{\varepsilon^2})$ queries – our algorithm pays an extra ε^{-1} factor in estimating the ratios on the edges. Specifically, consider the instance $w_i = (2\varepsilon)^i$ for $i \in [n]$, $\varepsilon \in (0, 1/4)$. Since $w_i > 2w_{i+1}$ but $w_{i+1}/w_i > \varepsilon$, one can show that our algorithm will build a forest with $\Theta(n)$ edges. The ratio on each such edge is upper bounded by $O(\varepsilon)$ and therefore estimating it within $(1 \pm \varepsilon)$ with high probability would require $\Theta(\frac{\log n}{\varepsilon^3})$ queries – thus, our algorithm makes $\Omega(\frac{n \log n}{\varepsilon^3})$ queries on this instance.

We also mention that our algorithm, in general, requires estimates as accurate as $1 \pm \varepsilon$. Consider a subset of the instance containing one large item of weight $w_1 = 1$ and $\frac{1}{\varepsilon}$ small items ($w_2, \ldots, w_t$ for $t = 1/\varepsilon + 1$) of weight ε. Our algorithm would separate these items into two clusters, one containing only w_1 and the other containing $w_2, \ldots, w_t$, and then it would estimate the ratio of the two centers. If this estimate is off by significantly more than $1 \pm \varepsilon$ (say $1 \pm v$, with $v > \varepsilon$), then the ratio of the large item's weight to the total of the small items also has error $1 \pm v$, causing the estimated winning probability of the large item against all the small ones to be wrong by an additive $\Theta(v)$.

A natural direction to explore would be choosing the precision on the edges dynamically rather than always using $1 \pm \varepsilon$. However, this would require a substantially different analysis and a different estimation-forest (or estimation-graph) topology. Indeed, our current topology can create stars where an item of weight $w_1 = 1$ gets attached to two items: one of weight $w_2 = 2\varepsilon$ and the other of weight $w_3 = 5\varepsilon$. Thus, one is forced to estimate the ratios between $\{w_1, w_2\}$ and $\{w_1, w_3\}$ within $1 \pm \varepsilon$ so to maintain a good estimate for $\{w_2, w_3\}$ as well –

even though w_1 is much larger than w_2 and w_3. Note that it is easy to construct an instance where this construction appears $\Theta(n)$ times, resulting in a cost of $\Theta(\frac{n \log n}{\varepsilon^3})$ if one uses the topology produced by our algorithm. Thus, a substantially different algorithm and analysis would be required to improve the dependency on ε.

Finally, it is unclear if an $O(\frac{n \log n}{\varepsilon^2})$ algorithm exists at all. In a slightly more general model than MNLs, Falahatgar et al. [17] showed that if one wants to approximate the distributions on all pairs by querying only pairs, then $\Omega(\frac{n \log n}{\varepsilon^3})$ queries are necessary (under the assumption that $n \geq 1/\varepsilon$). While this result does not apply to our setting, since it was proved in a more general model, it provides some evidence that ε^{-2} is not necessarily achievable. On the other hand, there is a trivial $O(n2^n/\varepsilon^2)$ algorithm if we can query slates of arbitrary size – however, this is better than $O(\frac{n \log n}{\varepsilon^3})$ only when $\varepsilon < 2^{-n} \log n$. Under the natural assumption that $n \geq 1/\varepsilon$, it is not clear whether one can do better than $O(\frac{n \log n}{\varepsilon^3})$.

8 Conclusions and Open Problems

In this paper, we considered the problem of learning an unknown MNL by making queries to a Sample oracle so that the learned weights can be used to provide an estimate to the distribution of each slate within an ℓ_1-error of ε. We developed two algorithms for this task: one for the adaptive setting and one for the non-adaptive setting.

Our adaptive algorithm has a query complexity of $O(\frac{n \log n}{\varepsilon^3})$ for $\delta = \frac{1}{\mathsf{poly}(n)}$, which is nearly matched by our lower bound of $\Omega(\frac{n \log n}{\varepsilon^2})$. The main open question left by our work is to resolve the gap in the accuracy parameter ε. We have shown that the lower bound holds for ℓ_∞, while our algorithm's guarantees hold for the harder setting of ℓ_1-error; this opens up the possibility that the optimal query complexity in ε may differ for the ℓ_∞ and ℓ_1 case.

Our non-adaptive algorithm has a query complexity of $O(\frac{n^2 \cdot \log n \cdot \log(n/\varepsilon)}{\varepsilon^3})$ nearly matching our $\Omega(\frac{n^2 \log n}{\varepsilon^2})$ non-adaptive lower bound. Again, this leaves the analogue open problem of closing the gap between the upper and the lower bound.

Finally, our non-adaptive algorithm is based on an adaptive algorithm that queries each pair at most polylogarithmic many times. However, the latter is different from the $O(\frac{n \log n}{\varepsilon^3})$ algorithm we first design for the adaptive setting. A possible direction for future work would be to find a single algorithm which can be used to match the query complexity of our algorithms in both the adaptive and non-adaptive setting.

References

1 Tomer Adar. Tight simulation of a distribution using conditional samples. *arXiv*, 2506.18444, 2025. doi:10.48550/arXiv.2506.18444.

2 Tomer Adar, Eldar Fischer, and Amit Levi. Optimal mass estimation in the conditional sampling model. In *SODA*, 2026.

3 Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: a survey. *JML*, 22(7):1–108, 2021. URL: https://jmlr.org/papers/v22/18-546.html.

4 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. doi:10.1093/ACPROF:OSO/9780199535255.001.0001.

5 Clément L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Number 9 in Graduate Surveys. Theory of Computing Library, 2020.

6 Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SICOMP*, 44(3):540–616, 2015. doi:10.1137/130945508.

7 Clément L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Foundations and Trends® in Communications and Information Theory*, 19(6):1032–1198, 2022. doi:10.1561/0100000114.

8 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *ITCS*, pages 561–580, 2013. doi:10.1145/2422436.2422497.

9 Pinhan Chen, Chao Gao, and Anderson Y Zhang. Optimal full ranking from pairwise comparisons. *The Annals of Statistics*, 50(3):1775–1805, 2022.

10 Xi Chen, Sivakanth Gopi, Jieming Mao, and Jon Schneider. Competitive analysis of the top-k ranking problem. In *SODA*, pages 1245–1264, 2017. doi:10.1137/1.9781611974782.81.

11 Xi Chen, Yuanzhi Li, and Jieming Mao. A nearly instance optimal algorithm for top-k ranking under the multinomial logit model. In *SODA*, pages 2504–2522, 2018. doi:10.1137/1.9781611975031.160.

12 Yuxin Chen and Changho Suh. Spectral MLE: Top-K rank aggregation from pairwise comparisons. In *ICML*, pages 371–380, 2015. URL: http://proceedings.mlr.press/v37/chena15.html.

13 Flavio Chierichetti, Mirko Giacchini, Ravi Kumar, Alessandro Panconesi, and Andrew Tomkins. Tight bounds for learning RUMs from small slates. In *NeurIPS*, pages 105864–105886, 2024.

14 Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.

15 Otto Dykstra. Rank analysis of incomplete block designs: A method of paired comparisons employing unequal repetitions on pairs. *Biometrics*, 16(2):176–188, 1960.

16 Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *COLT*, pages 255–270, 2002. doi:10.1007/3-540-45435-7_18.

17 Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *ICML*, pages 1427–1436, 2018.

18 Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *ICML*, pages 1088–1096, 2017. URL: http://proceedings.mlr.press/v70/falahatgar17a.html.

19 Lester R. Ford Jr. Solution of a ranking problem from binary comparisons. *The American Mathematical Monthly*, 64(8P2):28–33, 1957.

20 Ruijian Han, Rougang Ye, Chunxi Tan, and Kani Chen. Asymptotic theory of sparse Bradley–Terry model. *Annals of Applied Probability*, 30:2491–2515, 2020.

21 Minje Jang, Sunghyun Kim, Changho Suh, and Sewoong Oh. Optimal sample complexity of m-wise data for top-k ranking. In *NIPS*, volume 30, 2017.

22 Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *ICML*, pages 655–662, 2012.

23 Shimon Kogan and Merav Parter. New diameter-reducing shortcuts and directed hopsets: Breaking the $O(\sqrt{n})$ barrier. In *SODA*, 2022.

24 Wanshan Li, Shamindra Shrotriya, and Alessandro Rinaldo. ℓ_∞-bounds of the MLE in the BTL model under general comparison graphs. In *UAI*, 2022.

25 Anuran Makur and Japneet Singh. Minimax hypothesis testing for the bradley–terry–luce model. *IEEE Transactions on Information Theory*, 71(12):9163–9202, 2025.

26 Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *JMLR*, 5:623–648, 2004. URL: https://jmlr.org/papers/volume5/mannor04b/mannor04b.pdf.

27 Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of Plackett–Luce models. In *NIPS*, volume 28, 2015.

28 Kuldeep S Meel, Gunjan Kumar, and Yash Pote. Distance estimation for high-dimensional discrete distributions. In *AISTATS*, pages 955–963, 2025. URL: https://proceedings.mlr.press/v258/meel25a.html.

29 Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. In *NIPS*, 2012.

30 Sahand Negahban, Sewoong Oh, and Devavrat Shah. Rank centrality: Ranking from pairwise comparisons. *Oper. Res.*, 65(1), 2017. doi:10.1287/OPRE.2016.1534.

31 Mark EJ Newman. Efficient computation of rankings from pairwise comparisons. *JMLR*, 24(238):1–25, 2023. URL: https://jmlr.org/papers/v24/22-1086.html.

32 Sam Olesker-Taylor and Luca Zanetti. An analysis of Elo rating systems via Markov chains. In *NeurIPS*, 2024.

33 Pinki Pradhan and Sampriti Roy. Distribution testing meets sum estimation. *arXiv*, 2504.15153, 2025. doi:10.48550/arXiv.2504.15153.

34 Charvi Rastogi, Sivaraman Balakrishnan, Nihar B. Shah, and Aarti Singh. Two-sample testing on ranked preference data and the role of modeling assumptions. *J. Mach. Learn. Res.*, 23(1), 2022. URL: https://jmlr.org/papers/v23/20-1304.html.

35 Wenbo Ren, Jia Liu, and Ness B. Shroff. On sample complexity upper and lower bounds for exact ranking from noisy comparisons. In *NIPS*, 2019.

36 Arjun Seshadri, Alex Peysakhovich, and Johan Ugander. Discovering context effects from raw choice data. In *ICML*, pages 5660–5669, 2019. URL: http://proceedings.mlr.press/v97/seshadri19a.html.

37 Arjun Seshadri, Stephen Ragain, and Johan Ugander. Learning rich rankings. In *NIPS*, 2020.

38 Arjun Seshadri and Johan Ugander. Fundamental limits of testing the independence of irrelevant alternatives in discrete choice. In *EC*, pages 65–66, 2019. doi:10.1145/3328526.3329656.

39 Nihar Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *JMLR*, 17(58):1–47, 2016. URL: https://jmlr.org/papers/v17/15-189.html.

40 Gordon Simons and Yi-Ching Yao. Asymptotics when the number of parameters tends to infinity in the Bradley–Terry model for paired comparisons. *The Annals of Statistics*, 27(3):1041–1060, 1999.

41 Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. Online rank elicitation for Plackett–Luce: A dueling bandits approach. In *NIPS*, 2015.

42 Kenneth E Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2003.

43 Ting Yan, Yaning Yang, and Jinfeng Xu. Sparse paired comparisons in the Bradley–Terry model. *Statistica Sinica*, 22:1305–1318, September 2012.

44 Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *JCSS*, 78(5):1538–1556, 2012. doi:10.1016/J.JCSS.2011.12.028.

45 Ernst Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29(1):436–460, 1929.