Efficient and Accurate Learning of Mixtures of Plackett-Luce Models

Duc Nguyen[1], **Anderson Y. Zhang**[2]

[1] Depart of Computer and Information Science, University of Pennsylvania
[2] Department of Statistics and Data Science, University of Pennsylvania
mdnguyen@seas.upenn.edu, ayz@wharton.upenn.edu

Abstract

Mixture models of Plackett-Luce (PL), one of the most fundamental ranking models, are an active research area of both theoretical and practical significance. Most previously proposed parameter estimation algorithms instantiate the EM algorithm, often with random initialization. However, such an initialization scheme may not yield a good initial estimate and the algorithms require multiple restarts, incurring a large time complexity. As for the EM procedure, while the E-step can be performed efficiently, maximizing the log-likelihood in the M-step is difficult due to the combinatorial nature of the PL likelihood function. Therefore, previous authors favor algorithms that maximize surrogate likelihood functions. However, the final estimate may deviate from the true maximum likelihood estimate as a consequence. In this paper, we address these known limitations. We propose an initialization algorithm that can provide a provably accurate initial estimate and an EM algorithm that maximizes the true log-likelihood function efficiently. Experiments on both synthetic and real datasets show that our algorithm is competitive in terms of accuracy and speed to baseline algorithms, especially on datasets with a large number of items.

Introduction

Learning to rank is an active area of research with wide-ranging applications in recommendation systems, information retrieval, crowdsourcing and the social sciences. The Plackett-Luce (PL) model (Plackett 1975; Luce 1959) is one of the most fundamental ranking models. In a universe of n items, the PL model posits that item i has a latent *utility* $\theta_i^* \in \mathbb{R}$. The probability of observing a full ranking π given by the user (most preferred item first) is given as

$$\mathbb{P}^{PL}(\pi = [\pi_1, .., \pi_n] \,|\theta^*) = \prod_{i=1}^{n-1} \frac{\exp\left(\theta^*_{\pi_i}\right)}{\sum_{j=i}^{n} \exp\left(\theta^*_{\pi_j}\right)}. \quad (1)$$

The maximum likelihood estimate (MLE) can be obtained using iterative algorithms such as the Minorize-Maximize (MM) algorithm of Hunter (2004) and enjoys favorable theoretical properties (Hajek, Oh, and Xu 2014). In recent years, an algorithm known as Luce spectral ranking (LSR) (Maystre and Grossglauser 2015) has become the method of choice for maximum likelihood inference for PL models. LSR outputs the MLE just like MM but is often much faster.

The PL model is closely connected to the Bradley-Terry-Luce (BTL) model (Luce 1959) for *pairwise comparisons*. For two items $i \neq j$, the probability that i is ranked ahead of j *in a ranking* is equal to the probability that i beats j in a *pairwise comparison* under the BTL model. That is,

$$\mathbb{P}^{PL}(\pi(i) < \pi(j)) = \mathbb{P}^{BTL}_{ij} = \frac{1}{1 + \exp\left(-(\theta^*_i - \theta^*_j)\right)}, \quad (2)$$

where $\pi(i)$ is the position of item i in ranking π.

The classical PL model assumes that there is a universal preference ordering of the items according to their utilities. However, in practice, there might be multiple subpopulations of users with different preference profiles which cannot be fully captured by a single PL model. In such settings, a mixture of PL models is a more appropriate modeling assumption.

Problem Descriptions. Consider a mixture model with K components and n items for some constant K. Let $\beta^* = [\beta^*_1, \ldots, \beta^*_K]^\top$, ${\beta^*}^\top \mathbf{1} = 1$ denote the mixing distribution. For component $k \in [K]$ (where $[a]$ denotes $[1, \ldots, a]$), the utility parameters for the items are

$$\theta^{*k} = [\theta^{*k}_1, \ldots, \theta^{*k}_n]^\top \in \mathbb{R}^n.$$

Let $\boldsymbol{\theta}^* = [\theta^{*1}, \ldots, \theta^{*K}] \in \mathbb{R}^{n \times K}$ denote the concatenation of the K sets of parameters. A *ranking dataset* Π is a collection of full rankings.

Consider the following generative model for a ranking dataset of size m. For $l \in [m]$, let $z^*_l \in [K]$ denote the mixture component membership where $\mathbb{P}(z^*_l = k) = \beta^*_k$. Then a permutation π_l is drawn from the PL distribution parametrized by θ^{*z_l}. That is,

$$\mathbb{P}^{PL}(\pi_l = [\pi_{l,1}, \ldots, \pi_{l,n}] \,|\, z^*_l, \boldsymbol{\theta}^*) = \prod_{i=1}^{n-1} \frac{\exp\left(\theta^{*z^*_l}_{\pi_{l,i}}\right)}{\sum_{j=i}^{n} \exp\left(\theta^{*z^*_l}_{\pi_{l,j}}\right)}, \quad (3)$$

where $\pi_{l,i}$ denote the i-th item in permutation π_l. The reader may recognize two identifiability issues here. The first is parameter translation. For each component, the distributions parametrized by θ^{*k} and $\theta^{*k} + c \cdot \mathbf{1}_n$ are the same for any $c \in \mathbb{R}$. The second is mixture components (columns of $\boldsymbol{\theta}^*$)

relabeling. To account for these issues, we consider the following error metric.

$$\text{dist}(\boldsymbol{\theta}, \boldsymbol{\theta}^*) := \min_{R \in \mathcal{O}^{K \times K}} \|N(\boldsymbol{\theta})R - N(\boldsymbol{\theta}^*)\|_F \,, \quad (4)$$

where $\mathcal{O}^{K \times K}$ is the set of all permutation matrices (Strang et al. 1993, Chapter 2) of size $K \times K$ and N is the normalization operator (i.e., $N(\boldsymbol{\theta})_{\cdot k} = \theta^k - \frac{1}{n}(\theta^k)^\top \mathbf{1}_n$).

Prior Works. Generalizing the PL model to mixtures adds a layer of complexity to the inference problem. In general, the likelihood function is non-convex in the model parameters. Most previously proposed algorithms instantiate the EM algorithm (Dempster, Laird, and Rubin 1977). As a general recipe, an EM algorithm is initialized with some parameter $\boldsymbol{\theta}^{(0)}$ (e.g., using random initialization). It then repeats the following two steps for $t = 1, 2, \ldots$ until convergence.

The E-step computes the posterior class probability conditioned on the current estimate:

$$q_l^k = \mathbb{P}(z_l^* = k \,|\, \pi_l, \boldsymbol{\theta}^{(t-1)}) \propto \beta_k \cdot \mathbb{P}^{PL}(\pi_l \,|\, z_l^* = k, \boldsymbol{\theta}^{(t-1)}) \quad (5)$$

for $l \in [m], k \in [K]$ where $\mathbb{P}^{PL}$ is given in Equation (3) and β the prior class probability. Thanks to the closed form of the PL likelihood function, the E-step can be done efficiently. The M-step obtains the next estimate $\boldsymbol{\theta}^{(t)}$ by maximizing the joint log-likelihood function which decomposes into K *weighted log-likelihood functions*. Namely,

$$\boldsymbol{\theta}^{(t)} = \arg\max_{\boldsymbol{\theta}} \sum_{k=1}^{K} \left(\sum_{l=1}^{m} q_l^k \log \mathbb{P}^{PL}(\pi_l, z_l^* = k \,|\, \boldsymbol{\theta}) \right). \quad (6)$$

Due to the combinatorial nature of the PL likelihood function, the derivative of the log likelihoood function has a complicated form. As a result, maximizing the (weighted) log-likelihood via gradient-based algorithms quickly becomes inefficient as n grows.

The first practical approach towards solving the M-step uses the Minorize-Maximize algorithm of Hunter (2004), yielding the EMM algorithm of Gormley and Murphy (2008). While guaranteed to solve the M-step, it has been observed that the MM subroutine converges slowly even for datasets with a moderate number of items (e.g., Figure 2). Motivated by practical concerns, researchers have developed *pseudo-likelihood* estimators that optimize, instead of the true log-likelihood function, *alternative objective functions*. Two such algorithms are the Generalized Method of Moments (GMM) of Azari Soufiani et al. (2013) and Composite Marginal Likelihood (CML) of Zhao and Xia (2018). It has been observed experimentally that GMM is considerably faster than MM and CML is even faster than GMM with comparable accuracy. Besides maximum likelihood (ML) inference methods, previous authors have also proposed Bayesian inference algorithms (Guiver and Snelson 2009; Mollica and Tardella 2017). In this paper, we focus primarily on ML algorithms but include additional experiments with Bayesian methods in the supplementary materials.

Using GMM and CML to solve the M-step gives us the EM-GMM algorithm (Zhao, Villamil, and Xia 2018) and the EM-CML algorithm (Zhao, Liu, and Xia 2020), respectively. The only non-EM algorithm for learning PL mixtures that we are aware of is a GMM-based algorithm proposed in Zhao, Piech, and Xia (2016); Zhao and Xia (2019). However, the construction of the algorithm is quite ad-hoc and the authors did not show extension of the algorithm to more than 2 mixture components. In addition, previous authors primarily restrict their experiments to datasets with a small number of items such as the SUSHI datasets (Kamishima 2003) with $n = 10$. It is unknown how the previous methods perform when n is large. Recent works have also studied PL mixtures learning with features and partial rankings (Tkachenko and Lauw 2016; Liu et al. 2019). While we include possible extensions of our algorithm in the supplementary materials, our main focus in this paper is an *improved algorithm for the classical setting*.

Our Contributions. We propose a new EM algorithm for learning mixtures of PL models that

- Has a provably accurate initialization procedure with a finite sample error guarantee, the first of its kind in the literature;
- Efficiently maximizes the weighted log-likelihood function in the M-step without using a surrogate likelihood or objective function, thus returning the true maximum likelihood estimate;
- Performs competitively with the previously proposed algorithms in terms of accuracy and speed, and is scalable to datasets with $n \geq 100$.

The Spectral EM Algorithm

In this section, we present our algorithmic contributions. Section describes the spectral initialization algorithm and Section describes the EM refinement procedure.

Spectral Initialization

The initialization for our algorithm is delegated to spectral clustering (Algorithm 1) and a least squares minimization algorithm (Algorithm 2). To apply spectral clustering, we first embed each ranking π_l into a 'pairwise vector' – $X_l \in \{0,1\}^{\binom{n}{2}}$ where each entry corresponds to a pair of items. As an overload of notation, we use $d = (d_1, d_2)$ where $d_1 < d_2$ to denote the entry corresponding to the pair (d_1, d_2). Define

$$X_{l,d}(\pi) = \begin{cases} 1 & \text{if } \pi_l(d_1) < \pi_l(d_2) \\ 0 & \text{otherwise} \end{cases}. \quad (7)$$

Let $X \in \mathbb{R}^{m \times \binom{n}{2}}$ denote the concatenation of the embeddings of m rankings in dataset Π. Given a target number of components K, Algorithm 1 can then be applied to the rows of X to obtain K clusters, $\{\hat{C}^k\}_{k=1}^K \subseteq [m]$.

For each cluster of rankings $\hat{C}^k$, we estimate the preference probability for a pair (i, j) as

$$\hat{P}_{ij}^k = \frac{1}{|\hat{C}^k|} \sum_{l \in \hat{C}^k} \mathbf{1}[\pi_l(i) < \pi_l(j)] \,. \quad (8)$$

From the preference probability estimates for all pairs, Algorithm 2 recovers the utility parameter $\hat{\theta}^k$. It applies the

Algorithm 1 Spectral Clustering with Adaptive Dimension Reduction

Input: Dataset $\Pi = \{\pi_1, \ldots, \pi_m\}$, number of mixture components K and threshold T.

Output: K clusters of rankings.

1: Embed the rankings as the rows of a matrix $X \in \{0,1\}^{m \times \binom{n}{2}}$ according to Equation (7).
2: Perform SVD: $X = USV^\top$, where the singular values are arranged from largest to smallest.
3: Let $\hat{r}$ be largest index in $[K]$ such that the difference between the successive singular values is greater than T, i.e., $\hat{r} = \max\{a \in [K] : S_{aa} - S_{(a+1)(a+1)} \geq T\}$.
4: Run k-means on the rows of $XV_{1:\hat{r}}$ with K clusters:

$$\left(\hat{z}, \{\hat{c}_k\}_{k=1}^K\right) = \underset{\substack{z \in \{1,\ldots,K\}^m \\ \{c_k\} \in \mathbb{R}^{\hat{r}}}}{\arg\min} \sum_{l=1}^{m} \|V_{1:\hat{r}}^\top X_l - c_{z_l}\|_2^2 .$$

5: Return clusters $\hat{C}^k = \{l \in [m] : \hat{z}_l = k\}$ for $k \in [K]$.

Algorithm 2 Least Squares Parameter Estimation

Input: Pairwise preference matrix $\hat{P} \in \mathbb{R}^{n \times n}$.

Output: Normalized parameter estimate $\hat{\theta}$.

1: Solve the least squares optimization problem

$$\hat{\theta} = \arg \max_{\theta : \theta^\top \mathbf{1} = 0} \sum_{i \neq j} (\hat{\phi}_{ij} - (\theta_i - \theta_j))^2 ,$$

where $\hat{\phi}_{ij} = \ln(\hat{P}_{ij}/(1 - \hat{P}_{ij}))$.

logit function on the pairwise probabilities and solves a constrained least squares minimization problem, which can be efficiently done using off-the-shelf solvers (Virtanen et al. 2020). Algorithm 3 summarizes the spectral initialization algorithm.

Remarks. The application of spectral clustering to mixtures of PL models has also appeared in a manuscript by Shah and Song (2018). There, the authors apply the classical spectral clustering algorithm – clustering the rows of $XV_{1:K}$ – and their analysis requires a spectral gap condition which is hard to verify. We use spectral clustering with adaptive dimension reduction and our analysis does not require any spectral gap condition (Zhang and Zhou 2022). Furthermore, we focus on *parameter estimation* while Shah and Song (2018) only focus on clustering, resulting in different theoretical guarantees. The choice of threshold T in Algorithm 3 is to satisfy a mild technical condition in the analysis of spectral clustering. In our experiments, the performance of the EM algorithm does not seem to critically depend on this threshold.

Intuition behind Algorithm 2. Recall the connection between the PL model and the BTL model in Equation (2). Suppose we observe a large sample drawn from a single PL distribution. Then $\hat{P}_{ij} \approx P^*_{ij} = 1/\left(\exp\left(-(\theta^*_i - \theta^*_j)\right)\right)$ and $\hat{\phi}_{ij} = \ln(\hat{P}_{ij}/1 - \hat{P}_{ij}) \approx \theta^*_i - \theta^*_j$. Solving the least squares optimization problem recovers $\hat{\theta} \approx \theta^*$. In the mixture setting, if the estimates $\hat{P}^k$'s are accurate, we obtain good parameter estimates (e.g., Theorem 1). Rajkumar and Agarwal (2016) apply a similar idea in their algorithm for *ranking from comparisons of $O(n \log n)$ pairs under a single BTL model*. They first apply the logit function on the pairwise preference probabilities, followed by a low rank matrix completion algorithm (Keshavan, Montanari, and Oh 2009). Their algorithm *produces a ranking*. On the other hand, our goal is mixture learning and the resulting theoretical analysis is different.

Algorithm 3 Spectral Initialization

Input: Dataset $\Pi = \{\pi_1, \ldots, \pi_m\}$, number of mixture components K.

Output: Parameter estimates for K mixture components $\hat{\boldsymbol{\theta}} = [\hat{\theta}^1, \ldots, \hat{\theta}^K] \in \mathbb{R}^{n \times K}$.

1: Run Algorithm 1 on Π with $T = \sqrt{n}\sqrt{m+n}\sqrt{\log n}$ to obtain K clusters $\hat{C}^1, \ldots, \hat{C}^K$.
2: Estimate the pairwise preference probabilities $\hat{P}^k_{ij}$ per Equation (8) for each cluster.
3: Run Algorithm 2 on $\{\hat{P}^k\}_{k=1}^K$ and return the parameter estimates for K mixture components.

Iterative Refinement via EM

The Weighted LSR Algorithm. As noted before, we wish to maximize the weighted log-likelihood (6) efficiently. Towards this goal, we generalize the Luce spectral ranking (LSR) algorithm (Maystre and Grossglauser 2015) to incorporates sample weights. The original LSR algorithm produces the MLE. Our generalized algorithm outputs the weighted MLE (see Theorem 2).

The intuition behind LSR is an interpretation of the *PL ranking generative process as a sequence of choices* (Plackett 1975). Given a ranking π_l, define its *choice breaking* as

$$\mathcal{B}_{\pi_l} = \{(\pi_{l,1}, \{\pi_{l,1}, \ldots, \pi_{l,n}\}, l), (\pi_{l,2}, \{\pi_{l,2}, \ldots, \pi_{l,n}\}, l) \ldots, (\pi_{l,n-1}, \{\pi_{l,n-1}, \pi_{l,n}\}, l)\} .$$

Each tuple $(i, A, l) \in \mathcal{B}(\pi_l)$ is a *choice enumeration* of the ranking π_l. Given a ranking dataset $\Pi = \{\pi_1, \ldots, \pi_m\}$, define the *choice breaking* of Π as the union of all ranking-level choice breakings:

$$\mathcal{B}_\Pi = \mathcal{B}_{\pi_1} \cup \ldots \cup \mathcal{B}_{\pi_m} . \tag{9}$$

Note that $|\mathcal{B}_\Pi| = m(n-1)$. When the dataset Π is clear from context, we simply use $\mathcal{B}$ to denote the dataset-level choice breaking.

We now introduce sample weights. Firstly, define the 'weight' of a choice breaking $\mathcal{B}$ with weight vector w and parameter $\theta \in \mathbb{R}^n$ as

$$\gamma(\mathcal{B}, w, \theta) = w^\top \left(\frac{1}{\sum_{j \in A} e^{\theta_j}} \right)_{(i,A,l) \in \mathcal{B}} , \tag{10}$$

where $w \in \mathbb{R}_+^{m(n-1)}$ is an arbitrary weight vector;

$\left(\frac{1}{\sum_{j\in A} e^{\theta_j}}\right)_{(i,A,l)\in\mathcal{B}}$ is also vector in $\mathbb{R}_+^{m(n-1)}$ where each entry corresponds to a choice enumeration (i, A, l).

The reader may recognize that the weight vector w has the same size as the choice breaking while sample weights are often given at the ranking level – each ranking π_l is assigned a weight q_l for $l \in [m]$ as in (6). Given sample weights $q = (q_1, \ldots, q_m)$, one simply sets

$$w = [\underbrace{q_1, \ldots, q_1}_{n-1 \text{ terms}}, \underbrace{q_2, \ldots, q_2}_{n-1 \text{ terms}}, \ldots, \underbrace{q_m, \ldots, q_m}_{n-1 \text{ terms}}]^\top . \quad (11)$$

Given a choice breaking $\mathcal{B}$ and items i, j, define the set of choice enumerations where i 'beats' j as

$$\mathcal{B}_{i\succ j} = \{(i, A, l) \in \mathcal{B} \,:\, j \in A\} .$$

As a shorthand notation, for a weight vector w corresponding to choice breaking $\mathcal{B}$, define $w_{j\succ i}$ as the sub-vector of w corresponding to $\mathcal{B}_{i\succ j}$.

Similarly to the original LSR algorithm, we construct a Markov chain (MC) and recover PL parameters from its stationary distribution. This MC has n states. Given choice breaking $\mathcal{B}$, weight vector w and parameter θ, the pairwise transition probabilities of M are given as

$$M_{ij} = \begin{cases} \frac{1}{d} \cdot \gamma(\mathcal{B}_{j\succ i}, w_{j\succ i}, \theta) & \text{if } i \neq j \\ 1 - \frac{1}{d} \cdot \sum_{k\neq i} \gamma(\mathcal{B}_{k\succ i}, w_{k\succ i}, \theta) & \text{if } i = j \end{cases} , \quad (12)$$

where d is a sufficiently large normalization constant such that M does not contain any negative entries. Intuitively, M_{ij} is proportional to the sum of the weights of all choice enumerations where j 'beats' i.

Algorithm 4 summarizes the weighted LSR algorithm. It repeatedly constructs a Markov chain based on the current estimate, computes its stationary distribution and recovers the next estimate until convergence. When sample weights are not given, the weighted LSR algorithm reduces to the original LSR algorithm.

The EM-LSR Algorithm. In the E-step, we compute the posterior class probabilities $q^k \in \mathbb{R}^m, k \in [K]$. The M step consists of K maximization problem as shown in Equation (6). These can be solved in parallel by running Algorithm 4 on Π using q^k as sample weights for $k \in [K]$. Algorithm 5 summarizes the overall algorithm.

In another EM-based approach for learning PL mixtures, Liu et al. (2019) use the *unweighted LSR* algorithm. There, the E-step remains the same. The key differences lie in initialization (they use random initialization) and in the M-step. Our algorithm maximizes the weighted log-likelihood via weighted LSR and is therefore an exact EM algorithm. On the other hand, Liu et al. use the posterior class probabilities to perform a random clustering of the rankings and then run unweighted LSR on each cluster, making their algorithm an *inexact* EM algorithm. From additional experiments in the supplementary materials, one can observe that the stochastic M-step actually leads to *worse estimates without a significant reduction in inference time*.

Algorithm 4 Weighted Luce Spectral Ranking

Input: Dataset $\Pi = \{\pi_1, \ldots, \pi_m\}$, (optional) weight vector $q \in \mathbb{R}_+^m$ and (optional) initial estimate $\hat{\theta}^{(0)} \in \mathbb{R}^n$.

Output: Normalized estimate of the item parameters $\hat{\theta} \in \mathbb{R}^n$.

1: Obtain choice breaking $\mathcal{B}$ from Π per Equation (9).
2: If the weight vector q is not given, set $q = \mathbf{1}_m$.
3: Construct w from q per Equation (11).
4: If the initial estimate is not given, set $\hat{\theta}^{(0)} = \mathbf{0}_n$.
5: For $t = 1, \ldots$ until convergence
  5.1: Construct a Markov chain M with pairwise transition probability per Equation (12) from choice breaking $\mathcal{B}$, weight vector w and parameter $\hat{\theta}^{(t-1)}$.
  5.2: Compute the stationary distribution of M (e.g., via power iteration), p and return the normalized estimate $\hat{\theta}^{(t)} = \log(p) - \left(\frac{1}{n}\sum_{i=1}^n \log(p)\right) \cdot \mathbf{1}_n$

Algorithm 5 Spectral EM (EM-LSR)

Input: Dataset $\Pi = \{\pi_1, \ldots, \pi_m\}$, number of components K, prior distribution β, (optional) initial estimate $\hat{\boldsymbol{\theta}}^{(0)} \in \mathbb{R}^{n\times K}$.

Output: Normalized estimate $\hat{\boldsymbol{\theta}} = [\hat{\theta}^1, \ldots, \hat{\theta}^K]$.

1: If $\hat{\boldsymbol{\theta}}^{(0)}$ is not given, run Algorithm 3 on Π with K mixture components and set $\hat{\boldsymbol{\theta}}^{(0)}$ to the output.
2: For $t = 1, 2, \ldots$ until convergence
  2.1: E-step – Compute the class posterior probabilities $q_l^k = p(z_l^* = k|\pi_l, \hat{\boldsymbol{\theta}}^{(t-1)})$ for $l \in [m], k \in [K]$.
  2.2: M-step – Estimate $\hat{\theta}^{k(t)}$ by running Algorithm 4 on Π with sample weight vector $q^k = [q_1^k, \ldots, q_m^k]$ and initial estimate $\hat{\theta}^{k(t-1)}$ for $k \in [K]$.

Theoretical Analysis

In this section, we study the theoretical properties of EM-LSR. Specifically we present the finite sample error guarantee for the spectral initialization algorithm and study the M-step of not only our EM-LSR algorithm but also related EM algorithms.

Spectral Initialization

Central to the analysis of the spectral initialization algorithm is the accuracy of spectral clustering (Algorithm 1). Our analysis starts from the fact that, under the pairwise representation in Equation (7), the PL distribution exhibits *sub-gaussian characteristics* (Vershynin 2018; Shah and Song 2018). The detailed descriptions of these characteristics are not immediately important to our discussions so we refer the interested reader to the supplementary materials. However, we emphasize that these characteristics also appear in a broad class of ranking models known as *random utility models* (RUMs) that subsume the PL model. The spectral clustering algorithm is model-agnostic. It can be applied to mixtures of sub-gaussian distributions and enjoys high clustering accuracy if the signal-to-noise ratio (SNR) is high. We also show how, by changing

the mapping function used in Algorithm 2, we can perform parameter estimation for a general RUM, not just PL. Thanks to this flexibility, Algorithm 3 can be a useful tool for learning mixtures of general RUMs.

We now consider an expressive generative model for mixtures of K PLs where Algorithm 3 produces a provably accurate estimate. The generative model assumes that for all mixture components, only the utilities of the first L items are different while the those of the remaining $n-L$ items are the same. This model reflects the phenomenon where users from different sub-populations differ in their preference among a few items while the remaining items are essentially interchangeable. Intuitively, one would expect that when L is small, so is the difference between the subpopulations and it is harder to separate the rankings into the correct clusters. On the other hand, when L is large, the difference among the subpopulations is large and it is easier to separate the clusters. The following theorem captures this intuition.

Theorem 1. *Consider a mixture of K Plackett-Luce models with uniform mixing probabilities. Suppose that $\theta_i^{*k} = 0 \; \forall i \in [L+1:n]$ and $\theta_{1:L}^{*k} \sim \mathbb{N}(0, I_L)$ for $k \in [K]$. Fix a constant $\alpha > 0$. There exist constants c, c_1, C_1, C_2, D such that if $m \geq c \max\{K^4, Kn\}$ then the output $\hat{\boldsymbol{\theta}}$ of Algorithm 3 satisfies the following. If $L \geq c_1 \exp\left(C_1 \sqrt{\log n}\right)$, then*

$$\textit{dist}(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^*) = O\left(\exp\left(D\sqrt{\log n}\right)\left(\sqrt{\frac{K^2 n \log n}{m}} + \frac{\sqrt{Kn}}{e^{L^{0.99}}}\right)\right)$$

with probability $1 - O(\frac{K}{n^8}) - O(K^2 n^2 \exp\left(-L^{0.99}\right))$. If $L \geq C_2 n^\alpha$ and assuming that $n = \omega(\log m)$, then

$$\textit{dist}(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^*) = O\left(\exp\left(D\sqrt{\log n}\right)\sqrt{\frac{K^2 n \log n}{m}}\right)$$

with probability $1 - O(\frac{K}{n^8}) - O(K^2 n^2 \exp\left(-n^\alpha\right))$.

The first error bound is a sum of two terms. The first is the estimation error incurred by Algorithm 2 which diminishes with increasing m. The second comes from the clustering error incurred by Algorithm 1 and is controlled by the SNR of the generative model. One can also check that $\exp\left(\sqrt{\log n}\right) = o(n^\alpha)$ for any $\alpha > 0$ and $\exp\left(\sqrt{\log n}\right) = \omega(\log n)$. When $L \approx \exp\left(O(\sqrt{\log n})\right)$ (low SNR), there is significant clustering error and the second term scales approximately as $O(\frac{\sqrt{n}}{e^L}) = O\left(\frac{1}{\text{poly}(n)}\right)$. Hence, Algorithm 3 converges to within a small radius around $\boldsymbol{\theta}^*$ given a sufficiently large m. However, when L is polynomial in n (high SNR), estimation error dominates clustering error, giving us the second error bound which diminishes with sample size m. In this regime, the spectral initialization algorithm works well as a *standalone mixture learning algorithm*. Note that this guarantee holds even for a small $\alpha > 0$, when the fraction of 'informative' items diminishes: $L/n = o(1)$. Our proposed generative model is new and could be a useful analysis framework for future works. To the best of our knowledge, the finite sample error bounds are also the *first of their kind in the literature*.

Iterative Refinement via EM

Accuracy of the M-step. The following theorem generalizes Theorem 1 of Maystre and Grossglauser (2015).

Theorem 2. *The output of weighted LSR (Algorithm 4) is the maximum weighted log-likelihood estimate:*

$$\theta_q^{MLE} := \arg\max_\theta \sum_{l=1}^m q_l \cdot \log \mathbb{P}^{PL}(\pi_l, z_l \,|\, \theta)\,.$$

As noted before, the EMM algorithm is an alternative approach that exactly solves the M-step using the (weighted) MM algorithm. In other words, *assuming perfect numerical precision and the same initialization*, EMM and EM-LSR will produce the same final estimate. However, our EM-LSR algorithm is often much faster than EMM (e.g., Figure 1).

Convergence of EM. It is well known that the EM algorithm converges to a stationary point (Wu 1983). There is, unfortunately, no guarantee how close such a point is to the global optimum. However, assuming correct model specification and that the initial estimate falls within a neighborhood around θ^* which satisfies certain high SNR conditions, the EM algorithm will converge to θ^* (Wang et al. 2015; Wu et al. 2016; Balakrishnan, Wainwright, and Yu 2017). The area around θ^* where this desirable behaviour occurs is referred to as the *basin of attraction*. We leave the detailed characterization of the basin of attraction as a subject of future studies.

True Likelihood versus Surrogate Likelihood. For two other commonly used EM algorithms in the literature – EM-CML and EM-GMM – previous authors use random initialization. On the other hand, ours uses spectral initialization. However, initialization is not the only differentiating characteristic of our algorithm. In fact, our algorithm, EM-CML and EM-GMM are *fundamentally different EM-based algorithms*. To see why, one needs to inspect the objective function of the M-step. Suppose that all three algorithms are initialized at some $\hat{\boldsymbol{\theta}}^{(0)}$. Let $\{q_l^k\}_{l\in[m]}^{k\in[K]}$ denote the posterior class probabilities conditioned on $\hat{\boldsymbol{\theta}}^{(0)}$ per Equation (5).

In the first iteration, EM-LSR and EMM maximize the weighted log-likelihood.

$$\hat{\boldsymbol{\theta}}_{\text{LSR}}^{(1)} = \arg\max_{\boldsymbol{\theta}} \sum_{l=1}^m \sum_{k=1}^K \left[q_l^k \cdot \log \mathbb{P}^{PL}(\pi_l, z_l \,|\, \theta^k)\right].$$

On the other hand, EM-CML maximizes the *composite (surrogate) marginal likelihood*. $\hat{\boldsymbol{\theta}}_{\text{CML}}^{(1)} = \arg\max_{\boldsymbol{\theta}}$

$$\sum_{l=1}^m \sum_{k=1}^K \left[\sum_{\substack{i,j:\\ \pi_l(i)<\pi_l(j)}} q_l^k \log\left(\frac{1}{1+\exp\left(-(\theta_i^k - \theta_j^k)\right)}\right)\right].$$

Lastly, EM-GMM *minimizes* the following function.

$$\hat{\boldsymbol{\theta}}_{\text{GMM}}^{(1)} = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^K \sum_{i\neq j} \left(\hat{F}_{ij}^k - \frac{1}{1+\exp\left(-(\theta_i^k - \theta_j^k)\right)}\right)^2,$$

where $\hat{F}_{ij}^k = \frac{\sum_{l=1}^m \mathbf{1}[\pi_l(i)<\pi_l(j)]\, q_l^k}{\sum_{l=1}^m q_l^k}$.

One can see that the objective functions are different and so are their solutions. Hence, even if we initialize all three algorithms with the same estimate, their trajectories will be different in general. While EM-LSR and EMM converges to the true MLE when initialized within the basin of attraction, this *may not be true for EM-GMM and EM-CML*. This difference is supported by our experiments, where even with the same initialization, the algorithms produce different final estimates.

Experiments

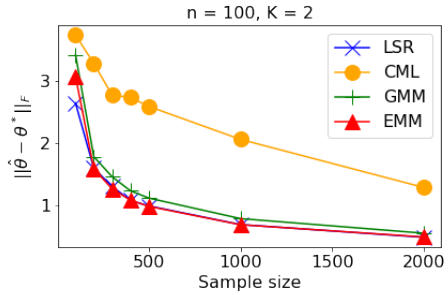
We compare our spectral EM algorithm to the following baselines: EMM, EM-GMM and EM-CML.

Synthetic Datasets. We simulate data from the generative model as described in Theorem 1. We set $n = 100$ and $L = 5$ while varying the number of mixture components K for different experiments. Figure 1 shows estimation error and total inference time against the sample size m, averaged over 25 trials. Spectral initialization consistently gives better initial estimates than both random initialization and GMM initialization (Zhao, Piech, and Xia 2016). To keep a fair comparison, we use spectral initialization for all algorithms. When K is small (e.g., Figures 1a and 1b) all four methods are quite accurate. When the number of mixture components are moderate (e.g., Figures 1c and 1d), the advantages that EM-LSR enjoys over the other methods become more apparent. While EMM becomes too inefficient for practical purposes, EM-LSR remains relatively efficient and produces more accurate estimates than both EM-CML and EM-GMM.

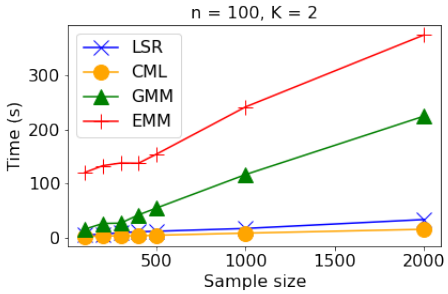
Real Datasets. We include commonly used datasets in previous works such as APA, Irish Elections (West, North, Meath) and SUSHI all with $n < 15$. We partition all the rankings with a 80-20 training-testing split; and the train rankings into 80% for inference and 20% for validation. K is chosen using Bayesian Information Criterion (Gelman, Hwang, and Vehtari 2014) on the validation set and the log-likelihood of the final model is evaluated using the test set. For these datasets, EM-LSR and EMM are the most accurate while EM-CML is the fastest, especially on datasets with a large m such as the Irish election datasets. To understand the relative speed between EM-LSR and EM-CML, note that the bottle neck in these EM algorithms is the M-step. The most time-consuming procedure in the M-step of EM-LSR is constructing the Markov chain in Algorithm 4 with time complexity $O(mn^2)$. For EM-CML, it is solving a constrained concave maximization problem via SLSQP (Virtanen et al. 2020) and may scale at least as $\Omega(n^3)$. [1]. Therefore, EM-CML tends to be faster for datasets with a small n and a large m. However, its inference time could grow significantly with n.

Indeed, the setting where EM-LSR outperforms the baselines is when n is large. We perform additional experiments on the ML-10M movie ratings datasets (Harper and Konstan 2015). To generate rankings, we first run a low rank matrix completion algorithm (Zitnik and Zupan 2012) on the user-

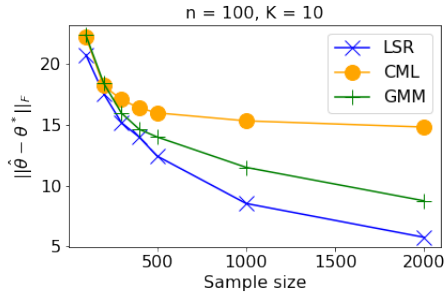
[1]SLSQP solves a sequence of quadratic optimization problems with n variables. Each solves a linear system with n variables and n equations and generally takes $O(n^3)$ (Strang et al. 1993).



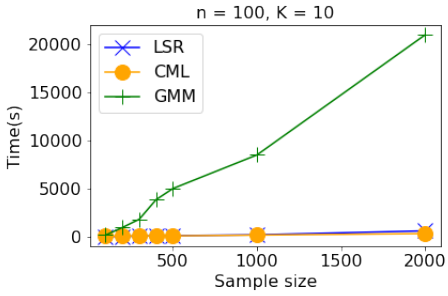
(a) For a small number of mixture components, all methods are quite accurate. As the theory implies, EMM and EM-LSR produce similar estimates.



(b) EM-LSR is comparatively efficient (figure shows total inference time).



(c) For a moderate number of mixture components, EM-LSR is the most accurate method (EMM not shown due to timeout).



(d) EM-LSR and EM-CML are the only two methods that are efficient for a moderate number of mixture components.

Figure 1. ℓ_2 error and inference time on synthetic datasets. EM-LSR is competitive in terms of accuracy and speed to the baseline algorithms.

item rating matrix to fill in the missing entries. We then select n movies from the set of all movies and the rankings are obtained from the completed matrix. Figure 2 shows the performance of the four methods on two versions of the ML-10M datasets with $n = 25$ and $n = 100$ given increasing training data up to 14k. In the supplementary materials (Nguyen and Zhang 2023), we also include additional experiments, strategies to extend EM-LSR to handle partial rankings with ties and comparisons to a Bayesian method (Mollica and Tardella 2017). Table 1 summarizes the experimental results on real datasets.

	Test log-likelihood (inference time)			
Dataset	LSR	CML	GMM	EMM
APA	-4.6 (600)	-4.6 (**33**)	-4.6 (2.2K)	**-4.6** (9.24K)
West	-11.9 (810)	-12.0 (**200**)	-11.9 (5.8K)	**-11.9** (25K)
Sushi	**-13.6** (746)	-14.0 (**24.6**)	-13.8 (489)	-13.8 (1.2K)
North	**-18.7** (1.5K)	-18.9 (**120**)	-18.7 (3.1K)	-18.8 (14K)
Meath	-23.6 (1.5K)	-23.9 (**497**)	-23.7 (30K)	**-23.6** (70K)
ML (25)	-49.2 (3.7K)	-50.1 (**2.5K**)	-49.8 (26K)	**-49.2** (63K)
ML (50)	**-131 (5.8K)**	-132 (6.8K)	-132 (125K)	NA
ML (100)	**-326 (12K)**	-330 (27K)	-332 (490K)	NA
ML (200)	**-787 (25K)**	-799 (81K)	NA	NA

Table 1. Test log-likelihood and inference time on real datasets. 'NA' denotes not available due to timeout. Best performances are in bold.

Conclusion

We have proposed an accurate and efficient algorithm for learning a mixture of Plackett-Luce models. For future works, we would like to consider other initialization methods such as the method of moments or tensor decomposition. Detailed characterization of the basin of attraction within which the EM algorithm converges to the true parameter is also a challenging open question. On a more practical note, incorporating the representation power of deep neural networks into our algorithm will further increase its utility for large scale recommendation systems applications.

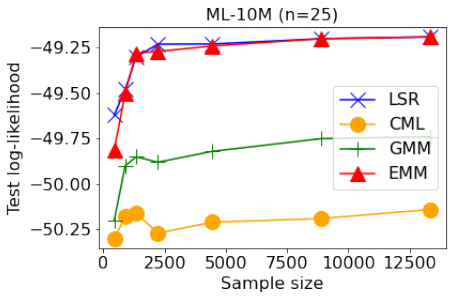
Acknowledgements

The authors thank the anonymous reviewers for their thoughtful suggestions and comments. The authors are supported by NSF Grant DMS-2112099. A.Z. acknowledges financial support from the Alfred H. Williams Faculty Scholar award.

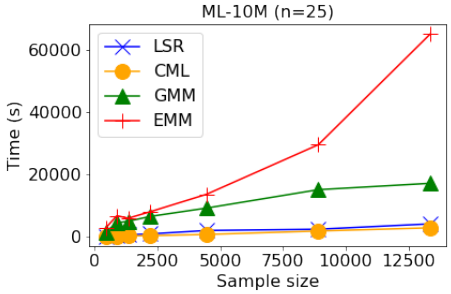
References

Azari Soufiani, H.; Chen, W.; Parkes, D. C.; and Xia, L. 2013. Generalized method-of-moments for rank aggregation. *Advances in Neural Information Processing Systems*, 26.

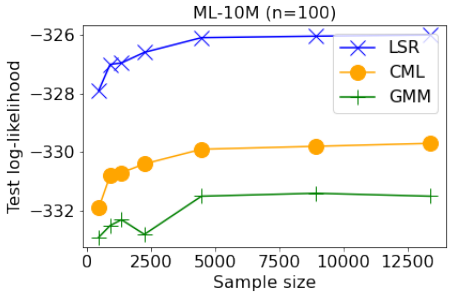
Balakrishnan, S.; Wainwright, M. J.; and Yu, B. 2017. Statistical guarantees for the EM algorithm: From population



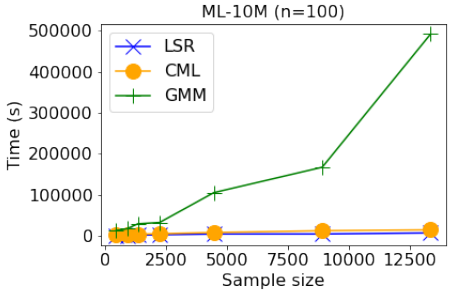
(a) With a small number of items, EMM and EM-LSR are more accurate.



(b) However, EM-LSR is also competitive in terms of efficiency.



(c) For a larger set of items, EM-LSR is the most accurate method (EMM not shown due to timeout).



(d) EM-LSR is comparatively scalable for larger datasets.

Figure 2. Test log-likelihood and inference time on ML-10M datasets. For larger datasets, EM-LSR is more accurate while being competitive in speed to the baseline algorithms.

to sample-based analysis. *The Annals of Statistics*, 45(1): 77–120.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22.

Gelman, A.; Hwang, J.; and Vehtari, A. 2014. Understanding predictive information criteria for Bayesian models. *Statistics and computing*, 24(6): 997–1016.

Gormley, I. C.; and Murphy, T. B. 2008. Exploring voting blocs within the Irish electorate: A mixture modeling approach. *Journal of the American Statistical Association*, 103(483): 1014–1027.

Guiver, J.; and Snelson, E. 2009. Bayesian inference for Plackett-Luce ranking models. In *proceedings of the 26th annual international conference on machine learning*, 377–384.

Hajek, B.; Oh, S.; and Xu, J. 2014. Minimax-optimal inference from partial rankings. *Advances in Neural Information Processing Systems*, 27.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4): 1–19.

Hunter, D. R. 2004. MM algorithms for generalized Bradley-Terry models. *The annals of statistics*, 32(1): 384–406.

Kamishima, T. 2003. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 583–588.

Keshavan, R.; Montanari, A.; and Oh, S. 2009. Matrix completion from noisy entries. *Advances in neural information processing systems*, 22.

Liu, A.; Zhao, Z.; Liao, C.; Lu, P.; and Xia, L. 2019. Learning plackett-luce mixtures from partial preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4328–4335.

Luce, R. D. 1959. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

Maystre, L.; and Grossglauser, M. 2015. Fast and accurate inference of Plackett–Luce models. *Advances in neural information processing systems*, 28.

Mollica, C.; and Tardella, L. 2017. Bayesian Plackett–Luce mixture models for partially ranked data. *Psychometrika*, 82(2): 442–458.

Nguyen, D.; and Zhang, A. Y. 2023. Efficient and Accurate Learning of Mixtures of Plackett-Luce Models. *arXiv preprint arXiv:2302.05343*.

Plackett, R. L. 1975. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2): 193–202.

Rajkumar, A.; and Agarwal, S. 2016. When can we rank well from comparisons of O (n\log (n)) non-actively chosen pairs? In *Conference on Learning Theory*, 1376–1401. PMLR.

Shah, D.; and Song, D. 2018. Learning RUMs: Reducing Mixture to Single Component via PCA. *arXiv preprint arXiv:1812.11917*.

Strang, G.; Strang, G.; Strang, G.; and Strang, G. 1993. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA.

Tkachenko, M.; and Lauw, H. W. 2016. Plackett-luce regression mixture model for heterogeneous rankings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 237–246.

Vershynin, R. 2018. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Wang, Z.; Gu, Q.; Ning, Y.; and Liu, H. 2015. High dimensional em algorithm: Statistical optimization and asymptotic normality. *Advances in neural information processing systems*, 28.

Wu, C.; Yang, C.; Zhao, H.; and Zhu, J. 2016. On the convergence of the em algorithm: A data-adaptive analysis. *arXiv preprint arXiv:1611.00519*.

Wu, C. J. 1983. On the convergence properties of the EM algorithm. *The Annals of statistics*, 95–103.

Zhang, A. Y.; and Zhou, H. H. 2022. Leave-one-out Singular Subspace Perturbation Analysis for Spectral Clustering. *arXiv preprint arXiv:2205.14855*.

Zhao, Z.; Liu, A.; and Xia, L. 2020. Learning Mixtures of Plackett-Luce Models with Features from Top-l Orders. *arXiv preprint arXiv:2006.03869*.

Zhao, Z.; Piech, P.; and Xia, L. 2016. Learning mixtures of Plackett-Luce models. In *International Conference on Machine Learning*, 2906–2914. PMLR.

Zhao, Z.; Villamil, T.; and Xia, L. 2018. Learning mixtures of random utility models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhao, Z.; and Xia, L. 2018. Composite marginal likelihood methods for random utility models. In *International Conference on Machine Learning*, 5922–5931. PMLR.

Zhao, Z.; and Xia, L. 2019. Learning mixtures of plackett-luce models from structured partial orders. *Advances in Neural Information Processing Systems*, 32.

Zitnik, M.; and Zupan, B. 2012. Nimfa: A Python Library for Nonnegative Matrix Factorization. *Journal of Machine Learning Research*, 13: 849–853.