Learning-to-Rank with Partitioned Preference: Fast Estimation for the Plackett-Luce Model

Jiaqi Ma
jiaqima@umich.edu
University of Michigan

Xinyang Yi
xinyang@google.com
Google AI

Weijing Tang
weijtang@umich.edu
University of Michigan

Zhe Zhao
zhezhao@google.com
Google AI

Lichan Hong
lichan@google.com
Google AI

Ed H. Chi
edchi@google.com
Google AI

Qiaozhu Mei
qmei@umich.edu
University of Michigan

Abstract

We investigate the Plackett-Luce (PL) model based listwise learning-to-rank (LTR) on data with *partitioned preference*, where a set of items are sliced into ordered and disjoint partitions, but the ranking of items within a partition is unknown. Given N items with M partitions, calculating the likelihood of data with partitioned preference under the PL model has a time complexity of $O(N+S!)$, where S is the maximum size of the top $M-1$ partitions. This computational challenge restrains most existing PL-based listwise LTR methods to a special case of partitioned preference, *top-K ranking*, where the exact order of the top K items is known. In this paper, we exploit a random utility model formulation of the PL model, and propose an efficient numerical integration approach for calculating the likelihood and its gradients with a time complexity $O(N+S^3)$. We demonstrate that the proposed method outperforms well-known LTR baselines and remains scalable through both simulation experiments and applications to real-world eXtreme Multi-Label classification tasks.

1 Introduction

Ranking is a core problem in many information retrieval systems, such as recommender systems, search engines, and online advertising. The industry-scale ranking systems are typically applied to millions of items in a personalized way for billions of users. To meet the need of scalability and to exploit a huge amount of user feedback data, *learning-to-rank* (LTR) has been the most popular paradigm for building the ranking system. Existing LTR approaches can be categorized into three groups: *pointwise* (Gey, 1994), *pairwise* (Burges et al., 2005), and *listwise* (Cao et al., 2007; Taylor et al., 2008) methods. The pointwise and pairwise LTR methods convert the ranking problem into regression or classification tasks on single or pairs of items respectively. As the real-world ranking data are often presented as (partially) ordered lists of items, the listwise LTR methods instead directly optimize objective functions defined on ranked lists of items, in order to preserve more information of the interrelations among items in a list.

One of the most well-known group of listwise LTR methods (Cao et al., 2007; Xia et al., 2008) are based on the Plackett-Luce (PL) model (Plackett, 1975; Luce, 1959). These methods define their objective functions as the likelihood of the observed ranked list under a PL model. Despite being useful in many cases, a major limitation of such methods comes from the fact that, evaluating the likelihood of general partial rankings under a PL model is usually intractable for a large number of items. This computational challenge restricts the application of existing PL-based listwise LTR methods to limited special cases of partial rankings, such as *top-K ranking*, where the exact order of the top K items is known.

In this paper, we extend PL-based listwise LTR to a more general class of partial rankings, the *partitioned preference* (Lebanon and Mao, 2008; Lu and Boutilier, 2014), defined as following: given N items, partitioned preference slices the items into M disjoint

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

partitions, where order of items within each partition are unknown while the M partitions have a global order. Partitioned preference not only is a strictly more general class of partial rankings compared to top-K ranking, but also better characterizes real-world ranking data. For example, in a page of recommended items, we usually only observe binary clicks or a small number of ordinal ratings (e.g., 5-star rating) as user feedback but do not know the exact order among the clicked items or items with the same rating scale. However, computing the exact likelihood of data with partitioned preference under the PL model requires an intractable time complexity[1] of $O(N+S!)$, where S is the size of the largest partition among the top $M-1$ partitions. While there exist sampling-based methods (Liu et al., 2019) that approximate the PL likelihood of partial rankings that are even more general than partitioned preferences, they cannot be directly adapted to the LTR setup where we usually need the gradients of the likelihood with respect to (w.r.t.) learnable parameters of a ranking model.

To overcome this computational challenge, we propose a novel numerical integration method. The key insight of our method is that, by exploiting a random utility model formulation of the PL model with Gumbel distribution (Yellott Jr, 1977; McFadden, 1978), we find that both the log-likelihood and its gradients can be rewritten as the summation of multiple one-dimensional integrals. This finding enables the proposed numerical integration approach, which efficiently approximates the log-likelihood and the gradients. We formally demonstrate that, as the number of items grows, the overall time complexity of the proposed numerical approach is $O(N + \frac{1}{\epsilon}S^3)$ in order to maintain a constant level of numerical error ϵ, which is much more efficient than the naive approach with the complexity $O(N + S!)$. We also discuss how our proposed approach might improve the generalized rank-breaking methods (Khetan and Oh, 2018).

We evaluate the effectiveness of the proposed method through both simulation and experiments with real-world datasets. For simulation, we show that the proposed method can better recover the ground-truth parameters of a PL model compared to baseline methods, including a method (Hino et al., 2010) that approximates the PL likelihood with a tractable lower bound. We also test the proposed method on real-world extreme multilabel (XML) classification datasets (Bhatia et al., 2016). We show that the proposed method can efficiently train neural network ranking models for items at million-level, and outperforms other popular listwise and pairwise LTR baselines.

[1]The exact versions of time complexity measures mentioned in this section can be found in Section 3.2.

2 Related Work

2.1 Learning-to-Rank

Our work falls in the area of LTR (Liu, 2009). The goal of LTR is to build machine learning models to rank a list of items for a given context (e.g., a user) based on the feature representation of the items and the context. The choice of the ranking objective plays an important role in learning the ranking models. Existing ranking objectives can be generally categorized in to three groups: pointwise (Gey, 1994), pairwise (Joachims, 2002; Burges et al., 2005), and listwise (Cao et al., 2007; Xia et al., 2008; Taylor et al., 2008; Christakopoulou and Banerjee, 2015; Ai et al., 2018; Wang et al., 2018; Bruch et al., 2020). The PL model has been widely used in listwise LTR methods (Cao et al., 2007; Xia et al., 2008; Schäfer, 2018). However, to our best knowledge, existing PL-based listwise methods cannot be applied to partitioned preference data, due to the aforementioned computational complexity of evaluating the likelihood. Our work tackles the computational challenge with a novel numerical approach. Beyond the computational challenge, another major limitation of the PL-based listwise methods is that, the underlying independence of irrelevant alternatives (IIA) assumption of the PL model, is sometimes overly strong in real-world applications (Seshadri and Ugander, 2019; Wilhelm et al., 2018; Christakopoulou and Banerjee, 2015). But more detailed discussions on the IIA assumption is out of the scope of this paper.

XML classification as a ranking problem. Given features of each sample, the XML classification task requires a machine learning model to tag the most relevant subset of an extremely large label set. The XML classification tasks were initially established as a reformulation of ranking problems (Agrawal et al., 2013; Prabhu and Varma, 2014), and the performance of which is primarily evaluated by various ranking metrics such as Precision@k or nDCG@k (Bhatia et al., 2016). The XML classification tasks are special cases of ranking with partitioned preference, where the class labels are considered items, and for each document its relevant labels form one partition and irrelevant labels form a second, lower-ranked partition. In this work, we apply the proposed method for ranking with partitioned preference to the XML classification datasets, and we find it achieving the state-of-the-art performance on datasets where the first partition, i.e., the set of relevant labels, is relatively large.

2.2 Rank Aggregation

Rank aggregation aims to integrate multiple partial or full rankings into one ranking. The multiple rankings are considered as noisy samples from underlying ground truth ranking preferences. Rank aggregation is a broader research area that includes LTR as a subproblem. Statistical modeling is a popular approach for rank aggregation. Various statistical models (Mallows, 1957; Luce, 1959; Plackett, 1975) are proposed to model the rank generation process in the real world. Among them, the PL model (Luce, 1959; Plackett, 1975) is one of the most widely-used. Evaluating the likelihood of the PL model on various types of partial rankings has been widely studied (Hunter et al., 2004; Maystre and Grossglauser, 2015; Liu et al., 2019; Yıldız et al., 2020; Zhao and Xia, 2020). However, we note that many of these studies (Maystre and Grossglauser, 2015; Liu et al., 2019) are designed for ranking data without any features, and thus are not suitable for LTR tasks. The ones (Yıldız et al., 2020; Zhao and Xia, 2020) that can leverage sample features are not directly applicable to large-scale partitioned-preference data. It is worth noting that, our proposed method shares the motivation of approximating the intractable PL likelihood using sampling methods (Liu et al., 2019), as numerical integration is a special case of sampling. However, the integral form of the PL likelihood inspired by the connection between the PL model and Gumbel distribution makes our method more efficient than a general sampling method.

3 Approach

3.1 Problem Formulation: Learning PL Model from Partitioned Preference

Suppose there are N different items in total and we denote the set $\{1, \cdots, N\}$ by $\lfloor N \rfloor$. The PL model and the partitioned preference are formally defined below.

Definition 1 (Plackett-Luce Model (Plackett, 1975; Luce, 1959))**.** *Given the utility scores of the N items, $\boldsymbol{w} = [w_1, w_2, \cdots, w_N]^T$, the probability of observing a certain ordered list of these items, $(i_1, i_2, \cdots, i_N)$, is defined as*

$$p((i_1, i_2, \cdots, i_N); \boldsymbol{w}) = \prod_{j=1}^{N} \frac{\exp(w_{i_j})}{\sum_{l=j}^{N} \exp(w_{i_l})}. \quad (1)$$

Definition 2 (Partitioned Preference (Lebanon and Mao, 2008; Lu and Boutilier, 2014))**.** *A group of M disjoint partitions of $\lfloor N \rfloor$, $S_1, S_2, \cdots, S_M$, is called a partitioned preference if (a) $S_1 \succ \cdots \succ S_M$, where $S_m \succ S_{\tilde{m}}$ indicates that any item in m-th partition has a higher rank than items in the $\tilde{m}$-th partition; (b) the rank of items within the same partition is unknown.*

Clearly, $\cup_{m=1}^{M} S_m = \lfloor N \rfloor$ and $S_m \cap S_{\tilde{m}} = \emptyset$ for any $1 \le m \ne \tilde{m} \le M$. We also denote the size of each partition S_m as n_m, $m = 1, \cdots, M$. Under a PL model parameterized by $\boldsymbol{w}$ as defined in Eq. (1), the probability of observing such a partitioned preference $S_1 \succ \cdots \succ S_M$ is given by

$$P(S_1 \succ \cdots \succ S_M; \boldsymbol{w}) = \sum_{(i_1, \cdots, i_N) \in \Omega(S_1 \succ \cdots \succ S_M; \lfloor N \rfloor)} \prod_{l=1}^{N} \frac{\exp(w_{i_l})}{\sum_{r=l}^{N} \exp(w_{i_r})}, \quad (2)$$

where $\Omega(\cdot; \lfloor N \rfloor)$ is a function that maps a partial ranking to the set of all possible permutations of $\lfloor N \rfloor$ that are consistent with the given partial ranking.

Typically, the utility scores $\boldsymbol{w}$ are themselves parameterized functions, e.g. neural networks, of the feature representation of the items and the context of ranking (e.g., a particular user). Suppose the features for item i are denoted as v_i and the item-independent context features are denoted as x. Then the utility score of i for a given context (e.g., user) can be written as $w_i(x, v_i; \theta)$, where θ represents the neural network parameters[2]. The problem of learning a PL model from data with partitioned preference can be formulated as maximizing the likelihood in Eq. (2) over θ.

However, evaluating the likelihood function naively by Eq. (2) requires a time complexity of $O(N + n_1! + \cdots + n_{M-1}!)$, which is made clear in the form of Eq. (3) given by Lemma 1. This implies that the likelihood becomes intractable as long as one partition is mildly large (e.g., $20! > 10^{18}$).

Lemma 1. *Let $\sigma(\cdot)$ be a function that maps a set of items to the set of all possible permutations of these items. Then Eq. (2) can be re-written as*

$$\prod_{m=1}^{M-1} \left(\sum_{(i_1, \cdots, i_{n_m}) \in \sigma(S_m)} \prod_{l=1}^{n_m} \frac{e^{w_{i_l}}}{\sum_{j \in R_m} e^{w_j} - \sum_{r=1}^{l-1} e^{w_{i_r}}} \right), \quad (3)$$

which happens to be

$$\prod_{m=1}^{M-1} P(S_m \succ R_{m+1}),$$

where R_m is the set of items that do not belong to the top $m-1$ partitions, i.e. $R_m = \cup_{r=m}^{M} S_r$.

3.2 Efficient Evaluation of the Likelihood and Gradients

Next, we present how to efficiently evaluate the likelihood Eq. (5) and its gradients. We derive a numerical integral approach based on the random utility model formulation of the PL model with Gumbel distribution (Yellott Jr, 1977; McFadden, 1978).

[2] We simplify the notation $w_i(x, v_i; \theta)$ as w_i for each $i \in \lfloor N \rfloor$ when there is no ambiguity.

The random utility model formulation of PL. A random utility model assumes that, for each context, the utility of preferring the item $i \in \lfloor N \rfloor$ is a random variable $u_i = w_i + \epsilon_i$. In particular, w_i is the aforementioned parameterized utility function, and ϵ_i is a random noise term that contains all the unobserved factors affecting the individual's utility. When each ϵ_i independently follows a standard Gumbel distribution (or equivalently, each u_i independently follows Gumbel(w_i), a Gumbel distribution with the location parameter set to w_i), we have the following fact (Yellott Jr, 1977), for any permutation of items $(i_1, \cdots, i_N)$,

$$P(u_{i_1} > u_{i_2} > \cdots > u_{i_N}) = \prod_{j=1}^{N} \frac{\exp(w_{i_j})}{\sum_{k=j}^{N} \exp(w_{i_k})}.$$

It implies that, after sampling N independent Gumbel variables, the ordered indices returned by sorting the Gumbel variables follow the PL model. Following this result, we have developed Proposition 1 that characterizes the preference for a given context between any two disjoint partitions.

Proposition 1. *Given a PL model parameterized by* $\boldsymbol{w}$*, for any* $A, B \subseteq \lfloor N \rfloor$ *and* $A \cap B = \emptyset$*, the probability of* $A \succ B$ *is given by*

$$\begin{aligned} P(A \succ B; \boldsymbol{w}) &= P(\min_{a\in A} g_{w_a} > \max_{b\in B} g_{w_b}) \\ &= \int_{u=0}^{1} \prod_{a\in A} (1 - u^{\exp(w_a - w_B)}) du, \end{aligned} \quad (4)$$

where g_w *denotes a random variable following Gumbel*(w) *and* $w_B = \log \sum_{b\in B} \exp(w_b)$.

Kool et al. (2020) have shown a weaker version of Proposition 1 where $A \cup B = \lfloor N \rfloor$. Here we extend it to the case where $A \cup B \subset \lfloor N \rfloor$, whose proof is given in Appendix A.2. In particular, the second equality in Eq. (4) provides an efficient way of computing the likelihood of the preference for a given context between two disjoint partitions.

Compute the likelihood and gradients by numerical integral. Following the random utility model formulation of PL, we can compute the log-likelihood of the preference for a given context among M partitions and its gradient efficiently by one-dimensional numerical integrals. For the likelihood function, it directly follows from Lemma 1 and Proposition 1 that

$$\begin{aligned} &P(S_1 \succ \cdots \succ S_M; \boldsymbol{w}) \\ &= \prod_{m=1}^{M-1} P(S_m \succ R_{m+1}; \boldsymbol{w}) \\ &= \prod_{m=1}^{M-1} \int_{u=0}^{1} \prod_{i\in S_m} \left(1 - u^{\exp\left(w_i - w_{R_{m+1}}\right)}\right) du, \end{aligned} \quad (5)$$

where $w_{R_{m+1}} = \log \sum_{j\in R_{m+1}} \exp(w_j)$. Therefore, we can compute the log-likelihood function through $M-1$ one-dimensional numerical integration. From Lemma 2, we can see that the gradients of the log-likelihood can also be obtained by $N - n_M$ numerical integrations using Eq. (6).

Lemma 2. *The gradients of the log-likelihood w.r.t.* $\boldsymbol{w}$ *can be written as*

$$\begin{aligned} &\nabla_{\boldsymbol{w}} \log P(S_1 \succ \cdots \succ S_M; \boldsymbol{w}) \\ &= -\sum_{m=1}^{M-1} \frac{1}{P(S_m \succ R_{m+1}; \boldsymbol{w})} \sum_{i\in S_m} \nabla_{\boldsymbol{w}} \exp(w_i - w_{R_{m+1}}) \\ &\quad \cdot \int_{u=0}^{1} \Big[\prod_{j\in S_m} (1 - u^{\exp(w_j - w_{R_{m+1}})}) \Big] \\ &\quad \cdot \frac{u^{\exp(w_i - w_{R_{m+1}})} \log u}{1 - u^{\exp(w_i - w_{R_{m+1}})}} du. \end{aligned} \quad (6)$$

Analysis of the computation cost. Suppose the number of numerical integration intervals is set as T. Evaluating the likelihood (5) requires a time complexity $O(N + T(N - n_M))$ and evaluating the gradients (6) requires a time complexity $O(N + T(\sum_{m=1}^{M-1} n_m^2))$. Next we quantify the minimum T required by certain desired numerical error. The numerical error consists of the discretization error, which is caused by the discretization of the integral, and the round-off error, which is caused by the finite precision of the computer. While there is non-negligible round-off error if we calculate the numerical integration directly using formula (5) and (6), this can be largely alleviated using common numerical tricks (see Appendix A.5). So we mainly focus on the analysis of the discretization error, which is given in Theorem 1.

Theorem 1. *Assume that there exist some constants* $c \in \mathbb{R}$*, and* $C > 1$ *and* $C = o(N^{1/6})$*, such that, for any* $1 \le m \le M-1$*,*

$$2 < \exp(w_{R_m} + c) < C. \quad (7)$$

Then for any $\epsilon > 0$ *and* $m = 1, \cdots, M-1$*,*

(a) we need at most $\frac{C^2(n_m+1)}{2\sqrt{3}\epsilon}$ *intervals for the* m*-th integral in the likelihood (5) to have a discretization error smaller than* ϵ*;*

(b) if there exists some constant $0 < C_0 < 1$ and $\frac{1}{C_0} = o(\sqrt{N})$, such that, for each $i \in S_m$, defining $a = \exp(w_i + w_{R_{m+1}} + 2c)$ and $b = \exp(w_i + c)$, the following conditions hold,

$$a > 4, a + 2b > 5, \text{ and } b > C_0, \tag{8}$$

then we need at most $\frac{\sqrt{6}C^{11/2}n_m^2}{C_0^2\epsilon}$ intervals for the (m, i)-th integral in gradients (6) to have a discretization error smaller than ϵ/n_m.

We note that the assumptions (7) and (8) in Theorem 1 respectively require the largest and the smallest neural network output logit, plus a constant c, to be not too far away from zero[3]. These assumptions are easy to satisfy by first controlling the scale of the logits $\boldsymbol{w}$, and then choosing a proper c to center the logits (c could be either positive or negative). We provide the proof of Theorem 1 in Appendix A.4.

Theorem 1 implies that the computation cost of the proposed method is overall $O(N + \frac{1}{\epsilon}\sum_{m=1}^{M-1} n_m^3)$ to maintain a discretization error at most ϵ for both the likelihood and its gradients, which is clearly much more efficient than the naive approach with factorial terms. We further highlight several computational advantages of the proposed numerical approach. 1) The whole computation is highly parallelizable: the computation of the T integrands and the product over S_m within each integrand can all be done in parallel. 2) The number of intervals T can be adjusted to control the trade-off between computation cost and accuracy[4]. 3) In large-scale ranking data, we often have $N \simeq n_M \gg \sum_{m=1}^{M-1} n_m$, thus $\sum_{m=1}^{M-1} n_m^3$ will be negligible for large N, resulting a linear complexity w.r.t. N.

3.3 Improving the Computational Efficiency of Generalized Rank-Breaking Methods

We discuss the potential application of the proposed numerical approach to *generalized rank-breaking methods* (Khetan and Oh, 2018) as a final remark of this section.

While partitioned preference is a general class of partial rankings for a set of items, it is not able to represent the class of all possible partial rankings, which is also known as *arbitrary pairwise preference* (Lu and Boutilier, 2014; Liu et al., 2019). It is challenging to learn arbitrary pairwise preference using a listwise method. To the best of our knowledge, there is no scalable listwise method that is able to learn industry-scale PL-based ranking models. Pointwise and pairwise methods are able to deal with any types of partial rankings at the expense of lower statistical efficiency. Generalized rank-breaking methods (Khetan and Oh, 2018) are recently proposed to better trade-off the computational and statistical efficiency in LTR.

An arbitrary pairwise preference of N items can be represented as a directed acyclic graph (DAG) of N nodes, where each node is an item and each directed edge represents the preference over a pair of items. The generalized rank-breaking methods first apply a graph algorithm to extract a *maximal ordered partition* of $\lfloor N \rfloor$, $S_1 \succ S_2, \cdots, S_M$: a group of M disjoint partitions of $\lfloor N \rfloor$ with largest possible M, such that the item preference in the M partitions is consistent with that of the DAG. One difference between data with partitioned preference and data with arbitrary pairwise preference is that the maximal ordered partition is not unique for the latter, as the maximal ordered partition does not preserve all relationships in the DAG. With the extracted partitions, we can maximize the likelihood of these partitions under a PL model to learn the model parameters. Khetan and Oh (2018) propose to calculate the likelihood as shown in Eq. (2), which has a time complexity involves factorials of the partition sizes. To overcome this challenge for learning large-scale data, existing methods need to approximate the likelihood by dropping the top $M - 1$ partitions with large sizes. In contrast, the proposed numerical approach in this paper can be directly applied to the likelihood evaluation step of generalized rank-breaking methods to significantly improve the computational efficiency.

4 Experiments

In this section, we report empirical results on both synthetic and real-world datasets. We compare the proposed method, denoted as **PL-Partition**, with two groups of baseline methods that can be applied to large-scale partitioned preference data.

First, we consider two softmax-based listwise methods: **PL-LB** (Hino et al., 2010) and **AttRank** (Ai et al., 2018). PL-LB optimizes a lower bound of the likelihood of partitioned preference under the PL model. In particular, for each $m = 1, \cdots, M - 1$, the term $P(S_m \succ R_{m+1}; \boldsymbol{w})$ in Eq. (5) is replaced by its lower bound $n_m! \prod_{i \in S_m} \Big(\exp(w_i) / \sum_{j \in S_m \cup R_{m+1}} \exp(w_j)\Big)$. AttRank optimizes the cross-entropy between the softmax outputs and an empirical probability based on the item relevance given by training labels.

For the second group of baselines, we consider two

[3]In practice, we find ranging from -10 to 10 is close enough to give good empirical results.

[4]In our experiments in the main paper, we fix the hyper-parameters $T = 10000$ and $c = 5$ in all settings. We also provide

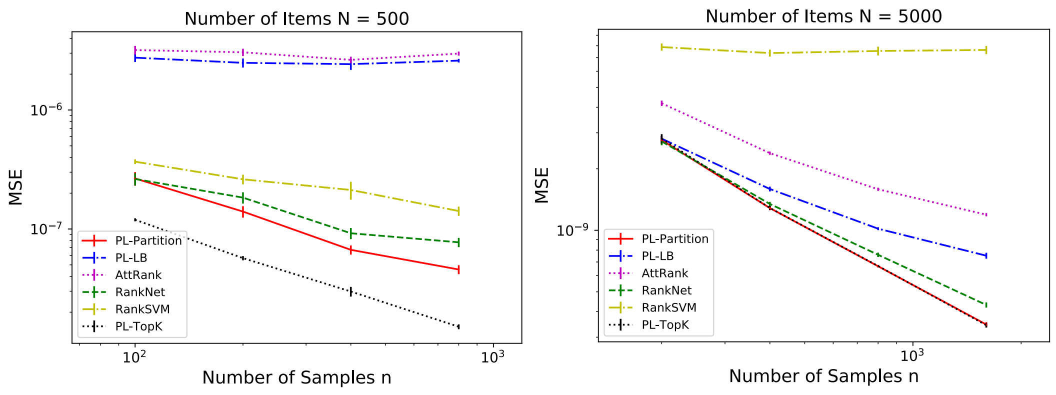

Figure 1: MSE of the estimated PL utility scores vs various numbers of items N and number of samples n. Both x-axis and y-axis are in the logarithmic scale with base 10. The results are averaged over 5 different random seeds and error bars indicate the standard error of the mean.

popular pairwise methods: **RankNet** (Burges et al., 2005) and **RankSVM** (Joachims, 2002). RankNet optimizes a logistic surrogate loss on each pair of items that can be compared. RankSVM optimizes a hinge surrogate loss on each pair of items that can be compared.

4.1 Simulation

We conduct experiments on synthetic data which is generated from a PL model. The goal of this simulation study is two-fold: 1) we investigate how accurate the proposed method can recover the ground truth utility scores of a PL model; 2) we empirically compare the computation costs of different methods over data with different scales.

Synthetic data generated from a PL model. We first generate a categorical probability simplex $p \in \Delta^{N-1}$ as the ground truth utility scores for N items following $p = \text{softmax}(q)$ and $q_i \overset{\text{i.i.d.}}{\sim} \text{uniform}(0, \log N), 1 \leq i \leq N$. Then we draw n samples of full ranking from a PL model parameterized by p. Finally, we randomly split the full rankings into M partitions and remove the order within each partition to get the partitioned preference. We note that the synthetic data is stateless (i.e., there is no feature for each sample), as this simulation focuses on the estimation of the PL utility scores p rather than the relationship between p and sample features. Further, in large-scale real-world applications, we often can only observe the order of limited items per sample. For example, a user can only consume a limited number of recommended items. To respect this pattern, we restrict the total number of items in top $M-1$ partitions to be at most 500 regardless of N. We fix $M = 4$ and generate data with varying (N, n) and random seeds.

Experiment setups. As the synthetic data is stateless, we only need to train N free parameters, with each parameter corresponding to an item. We use the proposed method and baseline methods to respectively train the parameters using stochastic gradient descent with early-stopping. We use AdaGrad optimizer with initial learning rate of 0.1 for all methods. We report the mean squared error (MSE) between the softmax of these free parameters and the PL utility scores p as a measure of how accurately different methods recover p.

MSE of the estimated PL utility scores. Figure 1 shows the MSE of the estimated PL utility scores by different methods over various N and n. To better compare results across data with different numbers of items N, we further include an oracle reference method **PL-TopK**, which has access to the full ranking of the items in the top $M-1$ partitions and optimizes the corresponding PL likelihood. First, as expected, the proposed PL-Partition method best recovers the ground truth utility scores in terms of MSE on all data configurations, as it numerically approximates the PL likelihood of the partitioned preference data. However, it is worth noting that PL-LB, while also trying to approximate the PL likelihood with a lower bound, performs even worse than pairwise methods when N is small, which indicates the existing lower bound method is not sufficient to take the full advantage of the PL model.

Computation cost. Figure 2 shows the time and memory costs of different methods over various N. The results are obtained using a single Nvidia V100 GPU. We report the total time of running 1000 steps of stochastic gradient descent with batch size 20. We also report the peak CUDA memory. The costs of both time and memory for the pairwise methods grow faster than those for the listwise methods as N increases. The two listwise baseline methods, AttRank and PL-LB, have similar memory cost. PL-LB has

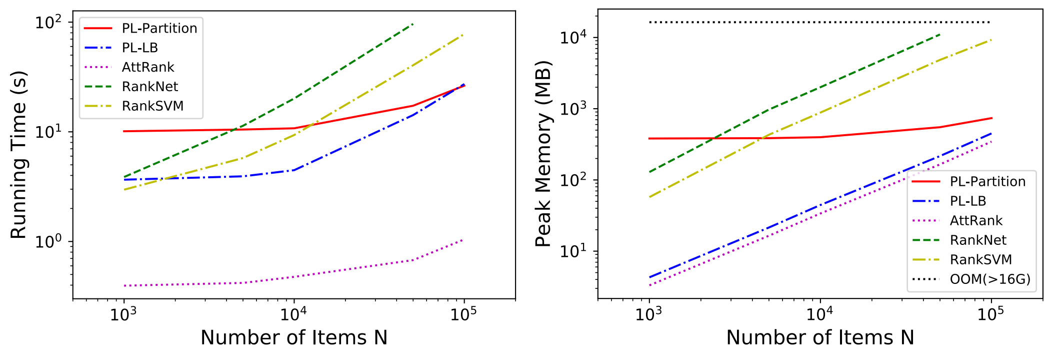

Figure 2: Time and memory cost of different methods for various number of items N (average of 5 different random seeds, each running 1000 steps of stochastic gradient descent with batch size 20). Both x-axis and y-axis are in the logarithmic scale with base 10. The standard errors are barely visible, thus omitted. RankNet result for $N = 10^5$ is missing as it ran out of memory.

Table 1: Precision@k and propensity-scored precision@k on the real-world XML datasets. Due to space limit, PL-Partition, RankNet, and RankSVM are respectively renamed as PL-P, R-Net, and R-SVM. **Bold numbers** indicate the best performance.

		PL-P	PL-LB	R-Net	R-SVM		PL-P	PL-LB	R-Net	R-SVM
D-1K	P@1	**66.72**	66.12	64.11	61.95	PSP@1	**33.22**	30.36	32.02	31.03
	P@3	**61.30**	60.13	58.46	57.05	PSP@3	**34.23**	31.69	32.76	31.61
	P@5	**56.63**	54.72	53.60	52.74	PSP@5	**35.02**	31.82	32.85	32.20
	P@10	**47.61**	45.76	45.62	44.57	PSP@10	**35.26**	31.91	33.27	32.71
E-4K	P@1	**78.12**	66.46	77.57	76.46	PSP@1	41.93	34.55	**42.81**	42.71
	P@3	62.81	51.58	**63.05**	61.88	PSP@3	43.46	34.37	45.55	**45.69**
	P@5	51.75	41.35	**52.55**	50.95	PSP@5	45.70	34.44	**48.61**	47.43
	P@10	33.79	26.99	**34.36**	32.56	PSP@10	57.13	42.50	**60.83**	57.50
W-31K	P@1	**85.97**	80.73	82.35	80.88	PSP@1	**13.01**	9.24	12.90	12.55
	P@3	**73.07**	54.14	67.33	60.30	PSP@3	**13.46**	7.61	12.95	12.18
	P@5	**63.01**	44.88	56.96	50.81	PSP@5	**14.03**	7.62	13.18	12.20
	P@10	**47.69**	31.36	42.47	37.50	PSP@10	**15.70**	7.61	14.90	13.04
D-200K	P@1	**47.58**	40.38	41.93	41.41	PSP@1	**8.72**	6.79	7.06	7.13
	P@3	**42.09**	37.40	38.92	38.46	PSP@3	**9.19**	7.74	8.17	8.13
	P@5	**39.23**	35.65	36.94	36.67	PSP@5	**9.82**	8.46	8.93	8.92
	P@10	**35.11**	32.40	33.66	33.44	PSP@10	**11.01**	9.75	10.20	10.21

Table 2: Comparisons between the proposed PL-Paritition and SLEEC and LEML. **Bold numbers** indicate the best performance. The results of SLEEC and LEML are from the XML repository (Bhatia et al., 2016).

		PL-Partition	SLEEC	LEML		PL-Partition	SLEEC	LEML
D-1K	P@1	66.72	**67.59**	65.67	PSP@1	**33.22**	32.11	30.73
	P@3	61.30	**61.38**	60.55	PSP@3	**34.23**	33.21	32.43
	P@5	**56.63**	56.56	56.08	PSP@5	**35.02**	33.83	33.26
E-4K	P@1	78.12	**79.26**	63.40	PSP@1	**41.93**	34.25	24.10
	P@3	62.81	**64.30**	50.35	PSP@3	**43.46**	39.83	27.20
	P@5	51.75	**52.33**	41.28	PSP@5	**45.70**	42.76	29.09
W-31K	P@1	**85.97**	85.88	73.47	PSP@1	**13.01**	11.14	9.41
	P@3	**73.07**	72.98	62.43	PSP@3	**13.46**	11.86	10.07
	P@5	**63.01**	62.70	54.35	PSP@5	**14.03**	12.40	10.55
D-200K	P@1	47.58	**47.85**	40.73	PSP@1	**8.72**	7.17	6.06
	P@3	42.09	**42.21**	37.71	PSP@3	**9.19**	8.16	7.24
	P@5	39.23	**39.43**	35.84	PSP@5	**9.82**	8.96	8.10

a larger running time due to the calculation of multiple partition functions. The proposed PL-Partition method has an overhead cost due to the numerical integration. However, we observe that this overhead cost is amortized as N increases. When $N = 10^5$, the computational cost of PL-Partition becomes close to that of PL-LB. Overall, this benchmark empirically demonstrates that the proposed method is scalable for large-scale applications[5].

4.2 Real-World Datasets

Experiment setups. We also verify the effectiveness of the proposed method on 4 real-world XML datasets (Bhatia et al., 2016): Delicious-1K (D-1K), Eurlex-4K (E-4K), Wiki10-31K (W-31K), and Delicious-200K (D-200K). The trailing number in the name of each dataset indicates the number of classes in the dataset[6]. Following many existing works (Prabhu and Varma, 2014; Bhatia et al., 2015) and the official instructions of XML classification repository (Bhatia et al., 2016), we evaluate different methods with 4 types of ranking metrics, Precision@k, Propensity-Scored Precision@k, nDCG@k, and Propensity-Scored nDCG@k. We observe that nDCG-based metrics show similar trends compared to their Precision-based counterparts. Due to space limit, we leave the results of nDCG metrics in Appendix A.7.

We first compare the proposed PL-Partition method with the 4 baseline methods. Note that PL-LB and AtRank collapse into exactly the same method on XML classification as the number of partitions is 2. So we only report the results of PL-LB. For each method, we train a neural network model with the same architecture, 2-layer fully connected network with ReLU activations and hidden size of 256. We train the neural networks with stochastic gradient descent using the ADAM optimizer. The batch size is fixed to 128. We use the official train-test split of each dataset and further split the training set into training and validation (9:1 for D-200K and 3:1 for other datasets). We tune the learning rate by line-search from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and apply early-stopping in training, based on validation sets.

We also compare PL-Partition with two state-of-the-art embedding-based XML classifiers, **SLEEC** (Bhatia et al., 2015) and **LEML** (Yu et al., 2014), which are listed on the XML repository leaderboard (Bhatia et al., 2016). SLEEC and LEML share similar model architectures with our setup, i.e., 2-layer neural networks, but use different training objectives: SLEEC uses a nearest-neighbor loss; LEML uses a least-square loss.

Results. As can be seen in Table 1, the proposed PL-Partition method significantly outperforms the softmax-based listwise method PL-LB on all datasets, indicating the importance of optimizing the proper utility function for listwise methods. PL-Partition also outperforms the pairwise methods RankSVM and RankNet on D-1K, W-31K, and D-200K, where the number of labels per sample is relatively large. When the number of labels per sample is relatively small, breaking the labels into pairwise comparisons leads to little loss of information, and pairwise methods perform well (E-4K).

Table 2 shows the comparison between PL-Partition and embedding-based XML classifiers SLEEC and LEML. SLEEC is better than LEML on all metrics. PL-Partition achieves similar performance to SLEEC on Precision@k and significantly outperforms the baselines on Propensity-Scored Precision@k. The propensity-scored metrics are believed to be less biased towards the head items. Thus the results indicate PL-Partition has better performance than SLEEC for the torso or tail items.

Discussions. We note for the task of XML classification, tree-based methods (Prabhu and Varma, 2014; Jain et al., 2016) sometimes outperform the embedding-based methods. The focus of this paper is to develop scalable listwise LTR methods for learning neural network ranking models from partitioned preference, instead of methods tailored for XML classification. Therefore we restrain our comparison to embedding methods only, whose model architectures are similar as the 2-layer neural networks in our experiment setup. We also note that SLEEC outperforms tree-based methods for D-200K on the XML repository leaderboard (Bhatia et al., 2016). This indicates PL-Partition achieves state-of-the-art performance on D-200K, where the top partition size is relatively large. Guo et al. (2019) recently showed that, with advanced regularization techniques, embedding-based methods trained by RankSVM or PL-LB can be significantly improved to surpass state-of-the-art tree-based methods on most XML datasets. It seems an interesting future direction to apply such regularization techniques on our proposed LTR objective.

5 Conclusion

In this paper, we study the problem of learning neural network ranking models with a Plackett-Luce-based

[5]In practice, both pairwise and listwise methods can be made more scalable by negative sampling.

[6]The average number of labels per sample (shown in the brackets after the dataset names): D-1K (19.03), E-4K (5.31), W-31K (18.64), D-200K (75.54). See Appendix A.7 for more summary statistics of these datasets.

listwise LTR method from data with partitioned preferences. We overcome the computational challenge of calculating the likelihood of partitioned preferences under the PL model by proposing an efficient numerical integration approach. The key insight of this approach comes from the random utility model formulation of Plackett-Luce with Gumbel distribution. Our experiments on both synthetic data and real-world data show that the proposed method is both more effective and scalable compared to popular existing LTR methods.

Acknowledgements

The authors would like to thank Ao Liu, Tyler Lu, Lirong Xia, and Zhibing Zhao for helpful discussions. Jiaqi Ma and Qiaozhu Mei were in part supported by the National Science Foundation under grant numbers 1633370 and 1620319.

References

Agrawal, R., Gupta, A., Prabhu, Y., and Varma, M. (2013). Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24.

Ai, Q., Bi, K., Guo, J., and Croft, W. B. (2018). Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 135–144.

Bhatia, K., Dahiya, K., Jain, H., Mittal, A., Prabhu, Y., and Varma, M. (2016). The extreme classification repository: Multi-label datasets and code.

Bhatia, K., Jain, H., Kar, P., Varma, M., and Jain, P. (2015). Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738.

Bruch, S., Han, S., Bendersky, M., and Najork, M. (2020). A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 61–69.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Christakopoulou, K. and Banerjee, A. (2015). Collaborative ranking with a push at the top. In *Proceedings of the 24th International Conference on World Wide Web*, pages 205–215.

Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. In *SIGIR'94*, pages 222–231. Springer.

Guo, C., Mousavi, A., Wu, X., Holtmann-Rice, D. N., Kale, S., Reddi, S., and Kumar, S. (2019). Breaking the glass ceiling for embedding-based classifiers for large output spaces. In *Advances in Neural Information Processing Systems*, pages 4944–4954.

Hino, H., Fujimoto, Y., and Murata, N. (2010). A grouped ranking model for item preference parameter. *Neural computation*, 22(9):2417–2451.

Hunter, D. R. et al. (2004). Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406.

Jain, H., Prabhu, Y., and Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

Khetan, A. and Oh, S. (2018). Generalized rank-breaking: computational and statistical tradeoffs. *The Journal of Machine Learning Research*, 19(1):983–1024.

Kool, W., van Hoof, H., and Welling, M. (2020). Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*.

Lebanon, G. and Mao, Y. (2008). Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9(Oct):2401–2429.

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, pages 4328–4335.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and trends in information retrieval*, 3(3):225–331.

Lu, T. and Boutilier, C. (2014). Effective sampling and learning for mallows models with pairwise-preference data. *The Journal of Machine Learning Research*, 15(1):3783–3829.

Luce, R. D. (1959). *Individual choice behavior: A theoretical analysis*. Wiley.

Mallows, C. L. (1957). Non-null ranking models. i. *Biometrika*, 44(1/2):114–130.

Maystre, L. and Grossglauser, M. (2015). Fast and accurate inference of plackett–luce models. In *Advances in neural information processing systems*, pages 172–180.

McFadden, D. (1978). Modeling the choice of residential location. *Transportation Research Record*, (673).

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

Prabhu, Y. and Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272.

Schäfer, D. (2018). *Dyad Ranking with Generalized Plackett-Luce Models*. PhD thesis, Paderborn, Universität Paderborn.

Seshadri, A. and Ugander, J. (2019). Fundamental limits of testing the independence of irrelevant alternatives in discrete choice. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 65–66.

Taylor, M., Guiver, J., Robertson, S., and Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86.

Wang, X., Li, C., Golbandi, N., Bendersky, M., and Najork, M. (2018). The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1313–1322.

Wilhelm, M., Ramanathan, A., Bonomo, A., Jain, S., Chi, E. H., and Gillenwater, J. (2018). Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2165–2173.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199.

Yellott Jr, J. I. (1977). The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2):109–144.

Yıldız, İ., Dy, J., Erdoğmuş, D., Kalpathy-Cramer, J., Ostmo, S., Campbell, J. P., Chiang, M. F., and Ioannidis, S. (2020). Fast and accurate ranking regression. In *International Conference on Artificial Intelligence and Statistics*.

Yu, H.-F., Jain, P., Kar, P., and Dhillon, I. (2014). Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601.

Zhao, Z. and Xia, L. (2020). Learning mixtures of plackett-luce models with features from top-l orders. *arXiv preprint arXiv:2006.03869*.