Probabilistic Models over Ordered Partitions with Applications in Document Ranking and Collaborative Filtering

Tran The Truyen[*] [†] Dinh Q. Phung[*] [‡] Svetha Venkatesh[*] [§]

Abstract

Ranking is an important task for handling a large amount of content. Ideally, training data for supervised ranking would include a complete rank of documents (or other objects such as images or videos) for a particular query. However, this is only possible for small sets of documents. In practice, one often resorts to document rating, in that a subset of documents is assigned with a small number indicating the degree of relevance. This poses a general problem of modelling and learning rank data with ties. In this paper, we propose a probabilistic generative model, that models the process as permutations over partitions. This results in super-exponential combinatorial state space with unknown numbers of partitions and unknown ordering among them. We approach the problem from the discrete choice theory, where subsets are chosen in a stagewise manner, reducing the state space per each stage significantly. Further, we show that with suitable parameterisation, we can still learn the models in linear time. We evaluate the proposed models on two application areas: (i) document ranking with the data from the recently held Yahoo! challenge, and (ii) collaborative filtering with movie data. The results demonstrate that the models are competitive against well-known rivals.

1 Introduction.

Ranking is an important data mining task for handling a large amount of content, e.g. we want to sort thousands of documents in the decreasing order of importance with respect to some criteria and select the top 10. In this paper, we are interested in the recent problem of learning to rank (e.g. see [17]), where we want to estimate a ranker that receives a set of objects and a query as the input and returns an ordered list.

This can be formulated as a supervised learning problem. Ideally, training data for supervised ranking would include a complete rank of objects for a particular query, but this is only possible for small sets of objects. In larger sets, it is more natural to rate an object from a rating scale, and the result is that many objects may have the same rating. Such phenomenon is common in large sets such as movies, books or web-pages wherein many objects may have *tied ratings*.

This paper focuses on the modelling and learning rank objects (e.g. documents) with ties. Previous work often involves paired comparisons (e.g. see [7][11][24]), ignoring simultaneous interactions among objects. We take an alternative approach by modelling objects with the same tie as a partition, translating the problem into ranking or ordering these partitions. This problem transformation results in a combinatorial problem which involves simultaneous set partitioning and subset ordering. For a given number of partitions, the order amongst them is a permutation of the partitions being considered, wherein each partition has objects of the same rank. A generative view of the problem can then be as follows: Choose the first partition with elements of rank 1, then choose the next partition from the remaining objects with elements ranked 2 and so on. The number of partitions then does not have to be specified in advance, and can be treated as a random variable. The joint distribution for each ordered partition can then be composed using a variant of the Plackett-Luce model [19][23], substituting *object* potentials by the *partition* potential. We propose two choices for these potential functions: First, we consider the potential of each partition to be the normalised sum of individual object potentials in that partition, leading to a simple normalisation factor in the estimation of the joint distribution. Second, we propose a MCMC based parameter estimation for the general choice of potential functions. We specify this model as the Probabilistic Model over Ordered Partitions.

Demonstrating its application to the learning to rank problem, we use the dataset from the recently held Yahoo! challenge [30]. We show that our results both in terms of predictive performance and training time are competitive with other well-known methods such as RankNet [2], Ranking SVM [15] and ListMLE [29]. With the choice of our proposed simple potential function, we get the added advantage of lower computational cost as it is linear in the query size compared to quadratic complexity for the pairwise methods. Another application is in collaborative filtering, where we use the MovieLens data and evaluate our algorithms against

[*]Department of Computing, Curtin University GPO Box U1987, Perth, Western Australia 6845, Australia.
[†]t.tran2@curtin.edu.au
[‡]d.phung@curtin.edu.au
[§]s.venkatesh@curtin.edu.au

the CoFi^{RANK} algorithm [28].

Our main contributions are the construction of a probabilistic model over ordered partitions and associated inference and learning techniques. We believe that our work is the first to address the problem of learning to rank with ties in its most generic form. The complexity of this problem is super-exponential with respect to number of objects (N) because both the number of partitions and their order are unknown - it grows exponentially as $N!/(2(\ln 2)^{N+1})$ [22, pp. 396–397]. Our contribution is to overcome this computational complexity through the choice of suitable potential functions, yielding learning algorithms with linear complexity, thus making the algorithm deployable in real settings. The novelty lies in the rigorous examination of probabilistic models over ordered partitions, extending earlier work in discrete choice theory [9][19][23]. The significance of the model is its potential for use in many applications. One example is the problem of learning to rank with ties which is studied in this paper. Further, the model opens new potential applications for example, novel types of clustering, in which the clusters are automatically ordered.

2 Background.

In this section, we review some background in rank modelling and learning to rank which are related to our work.

Rank models. Probabilistic models of permutation in general and of rank in particular have been widely analysed in statistical sciences (e.g. [21] for a comprehensive survey). Since the number of all possible permutations over N objects is $N!$, multinomial models are only computationally feasible for small N (e.g. $N \leq 10$). One approach to avoid this state space explosion is to deal directly with the data space, i.e. based on the distance between two ranks. The assumption is that there exists a *modal* ranking over all objects, and what we observe are ranks randomly distributed around the mode. The most well-known model is perhaps the Mallows [20], where the probability of a rank decreases exponentially with the distance from the mode. Depending on the distance measures, the model may differ; and the popular distance measures include those by Kendall and Spearman. The problem with this approach is that it is hard to handle the cases with multiple modes, with ties and with incomplete ranking.

Another line of reasoning is largely associated with the discrete choice theory (e.g. see [19]), which assumes that each object has an intrinsic worth which is the basis for the ordering between them. For example, Bradley and Terry [1] assumed that the probability of object preference is proportional to its worth, resulting in the logistic style distribution for pairwise comparison. Subsequently, Luce [19] and Plackett [23] extended this model to multiple objects. More precisely, for a set of N objects denoted by $\{x_1, x_2, ..., x_N\}$ the probability of ordering $x_1 \succ x_2 \succ ... \succ x_N$ is defined as

$$P(x_1 \succ x_2 \succ ... \succ x_N) = \prod_{i=1}^{N} \frac{\phi(x_i)}{\sum_{j=i}^{N} \phi(x_j)}$$

where $x_i \succ x_j$ denotes the preference of object x_i over x_j, and $\phi(x_i) \in \mathbb{R}^+$ is the worth of the object x_i. The idea is that, we proceed to select objects in a stagewise manner: Choose the first object among N objects with probability of $\phi(x_1)/\sum_{j=1}^{N} \phi(x_j)$, then choose the second object among the remaining $N-1$ objects with probability of $\phi(x_2)/\sum_{j=2}^{N} \phi(x_j)$ and so on until all objects are chosen. It can be verified that the distribution is proper, that is $P(x_1 \succ x_2 \succ ... \succ x_N) > 0$ and the probabilities of all possible orderings will sum to one. This paper will follow this approach as it is easily interpretable and flexible to incorporate ties and incomplete ranks.

Finally, for completeness, we mention in passing the third approach, which treats a permutation as a symmetric group and applying spectral decomposition techniques [8][13]. Their applicability to large-scale practical problems such as learning to rank and collaborative filtering is still unknown at this time of writing.

Learning to rank. Learning-to-rank is an active topic where the basic idea is that we can learn ranking functions which capture the relevance of an object (e.g. document or image) with respect to a query. The setting and goal are inherently different from traditional ranking in statistics. Often, the pool of all possible objects in a typical data management system is very large, and often changes over time. Thus, it is not possible to enumerate objects in rank models. Instead, each object-query pair is associated with a feature vector, which describes how relevant the object is with respect to the query. As a result, the distribution over objects is query-specific, and these distributions share the same parameter set. As discussed in [17], machine learning methods extended to ranking can be divided into:

Pointwise approach which includes methods such as ordinal regression [5][6]. Each query-document pair is assigned an ordinal label, e.g. from the set $\{0, 1, 2, ..., M\}$. This simplifies the problem as we do not need to worry about the exponential number of permutations. The complexity is therefore linear in the number of query-document pairs. The drawback is that the ordering relation between documents is not explicitly modelled.

Pairwise approach which spans preference to binary classification [2][10][15] methods, where the goal is to learn a classifier that can separate two documents (per query). This casts the ranking problem into a standard binary classification framework, wherein many algorithms are readily available, for example, SVM [15], neural network and logistic regression [2], and boosting [10]. The complexity is quadratic in number of documents per query and linear in number of queries. Again, this approach ignores the simultaneous interaction among objects within the same query.

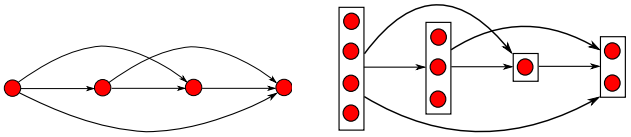

Figure 1: Complete ordering (left) versus subset ordering (right). For the subset ordering, the bounding boxes represents the subsets of elements of the same rank. Subset sizes are $4,3,1,2$, respectively.

Listwise approach which models the distribution of permutations [3][27][29]. The ultimate goal is to model a full distribution of all permutations, and the prediction phase outputs the most probable permutation. This approach appears to be most natural for the ranking problem. In fact, the methods suggested in [3][29] are applications of the Plackett-Luce model.

3 Modelling Sets with Ordered Partitions.

3.1 Problem Description. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a collection of N objects. In a complete ranking setting, each object x_i is further assigned with a ranking index π_i, resulting in the ranked list of $\{x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_N}\}$ where $\pi = (\pi_1, \ldots, \pi_N)$ is a permutation over $\{1, 2, \ldots, N\}$. For example, X might be a set of documents that are related to a query, and π_1 is the index to the first document, π_2 is the index to second document and so on. Ideally π should contain ordering information for all documents in the set; however, this task is not always possible for any non-trivial size N due to the labor cost involved[1]. Instead, in many situations, during training a document is *rated*[2] to indicate the its degree of relevance for the query. This creates a scenario where more than one document will be assigned to the same rating – a situation known as '*ties*' in learning-to-rank. When we enumerate over each object x_i and putting those with the same rating together, the set of N objects X can now be viewed as being divided into K partitions with each partition is assigned with a number to indicate the its unique rank $k \in \{1, 2, .., K\}$. The ranks are obtained by sorting ratings associated with each partition in the decreasing order. Our essential contribution in this section is a probabilistic model over this set of partitions, learning its parameter from data, and performing inference.

Consider a more generic setting in which we know that objects will be rated against an ordinal value from 1 to K but do not know individual ratings. This means that we have to consider all possible ways to split the set X into exactly K partitions, and then each time, *rank* those partitions from 1 to K wherein the kth partition contains all objects rated with the same value k. This is the first rough description of *state space* for our model. Formally, for a given K and the order among the partitions σ, we write the set $X = \{x_1, \ldots, x_N\}$ as a union of K partitions

$$X = \cup_{j=1}^{K} X_{\sigma_j} \tag{3.1}$$

where $\sigma = (\sigma_1, \ldots, \sigma_K)$ is a permutation over $\{1, 2, .., K\}$ and each partition X_k is a non-empty subset[3] of objects with the same rating k. These partitions are pairwise disjoint and having cardinality[4] range from 1 to N. It is easy to see that when $K = N$, each X_k is a singleton, σ is now a complete permutation over $\{1, \ldots, N\}$ and the problem reduces exactly to the complete ranking setting mentioned earlier. To get an idea of the state space, it is not hard to see that there are $\left| \begin{matrix} N \\ K \end{matrix} \right| K!$ ways to partition and order X where $\left| \begin{matrix} N \\ K \end{matrix} \right|$ is the number of possible ways to divide a set of N objects into K partitions, otherwise known as *Stirling numbers of second kind* [26, p. 105]. If we consider all the possible values of K, the size of our state space is

$$\sum_{k=1}^{N} \left| \begin{matrix} N \\ k \end{matrix} \right| k! = \text{Fubini}\,(N) = \sum_{j=1}^{\infty} \frac{j^N}{2^{j+1}} \tag{3.2}$$

which is also known in combinatorics as the Fubini's number [22, pp. 396–397]. This is a super-exponential growth number. For instance, $\text{Fubini}\,(1) = 1$, $\text{Fubini}\,(3) = 13$, $\text{Fubini}\,(5) = 541$ and $\text{Fubini}\,(10) = 102,247,563$. Its asymptotic behaviour can also be shown [22, pp. 396–397] to approach $N!/(2\,(\ln 2)^{N+1})$ as $N \to \infty$ where we note that $\ln(2) < 1$, and thus it grows much faster than $N!$. Clearly, for unknown K this presents a very challenging problem to inference and learning. In this paper, we shall present an efficient parameterisation and a generic MCMC-based approach to tackle this state-space explosion in supervised learning settings.

3.2 Probabilistic Model over Ordered Partitions. Return to our problem, our task is now to model a distribution over the ordered partitioning of set X into K partitions and the ordering $\sigma = (\sigma_1, \ldots, \sigma_K)$ among K partitions given in Eq (3.1):

$$p(X) = p(X_{\sigma_1}, \ldots, X_{\sigma_K}) \tag{3.3}$$

[1] We are aware that clickthrough data can help to obtain a complete ordering, but the data may be noisy.

[2] We caution the confusion between 'rating' and 'ranking' here. Ranking is the process of sorting a set of objects in an increasing or decreasing order, whereas in 'rating' each object is given with a value indicating its preference.

[3] Strictly speaking, a partition can be an empty set but we deliberately left out this case, because empty sets do not contribute to the probability mass of the model, and it does not match the real-world intuition of object's worth.

[4] More precisely, when the number of partitions K is given, the cardinality ranges from 1 to $N - K + 1$ since partitions are non-empty

A two-stage view has been given thus far: first X is partitioned in any arbitrary way so long as it creates K partitions and then these partitions are ranked, result in a ranking index vector σ. This description is generic and one can proceed in different ways to further characterise Eq (3.3). We present here a generative, multistage view to this same problem so that it lends naturally to the specification of the distribution in Eq (3.4): First, we construct a subset X_1 from X by collecting all objects which have the largest ratings. If there are more elements in the the remainder set $\{X \setminus X_1\}$ to be selected, we construct a subset X_2 from $\{X \setminus X_1\}$ whose elements have the second largest ratings. This process continues until there is no more object to be selected.[5] An advantage of this view is that the resulting total number of partitions K_σ is automatically generated, no need to be specified in advance and can be treated as a random variable. If our data truly contains K partitions then K_σ should be equal to K. Using the chain rule, we write the joint distribution over K_σ ranked partitions as

$$\begin{aligned} p(X_1,\ldots,X_{K_\sigma}) &= p(X_1)\prod_{k=2}^{K_\sigma} p(X_k \mid X_1,\ldots,X_{k-1}) \\ &= p_1(X_1)\prod_{k=2}^{K_\sigma} p_k(X_k \mid X_{1:k-1}) \end{aligned} \tag{3.4}$$

where we have used $X_{1:k-1} = \{X_1,\ldots,X_{k-1}\}$ for brevity.

3.3 Parameterisation, Learning and Inference. It remains to specify the local distribution $P(X_k \mid X_{1:k-1})$. Let us first consider what choices we have after the first $(k-1)$ partitions have been selected. It is clear that we can select any objects from the remainder set $\{X \setminus X_{1:k-1}\}$ for our next partition kth. If we denote this remainder set by $R_k = \{X \setminus X_{1:k-1}\}$ and $N_k = |R_k|$ is the number of remaining objects, then our next partition X_k is a subset of R_k; furthermore, there is precisely $\left(2^{N_k}-1\right)$ such non-empty subsets. Using the notation 2^{R_k} to denote the *power set* of the set R_k, i.e, 2^{R_k} contains all possible non-empty subsets[6] of R, we are ready to specify each local conditional distribution in Eq (3.4) as:

$$p_k(X_k \mid X_{1:k-1}) = \frac{\Phi_k(X_k)}{\sum\limits_{S\in 2^{R_k}} \Phi_k(S)} \tag{3.5}$$

where $\Phi_k(S) > 0$ is an order-invariant[7] set function defined over a set or partition S, and the summation in the denominator clearly makes the definition in Eq (3.5) a proper distribution. The set function $\Phi_k(\cdot)$ can also be interpreted as the potential function in standard probabilistic graphical models literature.

Although the state space 2^{R_k} for this local conditional distribution is significantly smaller than the space of all possible ordered partitions of N objects, it is still exponential as we have shown earlier to be $2^{N_k}-1$. In general, directly computing the normalising term is still not possible, let alone learning the model parameters. In what follows, we will study an efficient special case which has (sub)-quadratic complexity in learning, and a general case with MCMC approximation. *We further term our Probabilistic Model over Ordered Partition as PMOP.*

3.3.1 Full-Decomposition PMOP. Under a full-decomposition setting, we assume the following local *additive* decomposition at each kth step:

$$\Phi_k(X_k) = \frac{1}{|X_k|}\sum_{x\in X_k}\phi_k(x) \tag{3.6}$$

where $\phi(x)$ is some positive function[8] of object x.

The normalising term $|X_k|$ is to ensure that the probability is not monotonically increasing with number of objects in the partition. Given this form, the local normalisation factor represented in the denominator of Eq (3.5) can now efficiently represented as the sum of all weighted sums of objects. Since each object x in the remainder set R_k participates in the *same* additive manner towards the construction of the denominator in Eq (3.5), it must admit the following form[9]:

$$\sum_{S\in 2^{R_k}} \Phi_k(S) = \sum_{S\in 2^{R_k}} \frac{1}{|S|}\sum_{x\in S}\phi_k(x) = C \times \sum_{x\in R_k}\phi_k(x) \tag{3.7}$$

where C is some constant and its exact value is not essential under a maximum likelihood parameter learning treatment (readers are referred to Appendix A for the computation of C). To see this, substitute Eq (3.6) and (3.7) into Eq (3.5):

[5] This process resembles the generative process of Plackett-Luce discrete choice model [19][23], except we apply on partitions rather than single element. It clear from here that Plackett-Luce model is a special case of ours wherein each partition X_k reduces to a singleton.

[6] The usual understanding would also contain the empty set, but we exclude it in this paper.

[7] i.e., the function value does not depend on the order of elements within the partition.

[8] This is application specific, but in practice, it often has the well-known exponential form.

[9] To illustrate this intuition, suppose the remainder set is $R_k = \{a,b\}$, hence its power set, excluding $\emptyset$, contains 3 subsets $\{a\},\{b\},\{a,b\}$. Under the full-decomposition assumption, the denominator in Eq (3.5) becomes $\phi(r_a)+\phi(r_b)+\frac{1}{2}\{\phi(r_a)+\phi(r_b)\} = (1+\frac{1}{2})\sum_{x\in\{a,b\}}\phi(r_x)$. The constant term is $C = \frac{3}{2}$ in this case.

$$\log p(X_k \mid X_{1:k-1}) = \log \frac{\Phi_k(X_k)}{\sum_{S\in 2^{R_k}} \Phi_k(S)} = \log \frac{1}{C|X_k|} \frac{\sum_{x\in X_k} \phi_k(x)}{\sum_{x\in R_k} \phi_k(x)} = \log \frac{\sum_{x\in X_k} \phi_k(x)}{\sum_{x\in R_k} \phi_k(x)} - \log C|X_k| \tag{3.8}$$

Since $\log C|X_k|$ is a constant w.r.t the parameters used to parameterise the potential functions $\phi_k(\cdot)$, it does not affect the gradient of the log-likelihood. It is also clear that maximising the likelihood given in Eq (3.4) is equivalent to maximising each local log-likelihood function given in Eq (3.8) for each k. Discarding the constant term in Eq (3.8), we re-write it in this simpler form:

$$\log p(X_k \mid X_{1:k-1}) = \log \sum_{x\in X_k} g_k(x \mid X_{1:k-1}) \tag{3.9}$$
$$\text{where } g_k(x \mid X_{1:k-1}) = \frac{\phi_k(x)}{\sum_{x\in R_k} \phi_k(x)}$$

Depend on the specific form chosen for $\phi_k(x)$, maximising log-likelihood in the form of Eq (3.9) can be carried on in most cases. Gradient-based learning this type of model is generally takes N^2 time complexity . *However, using the dynamic programming technique, we show that if the function $\phi_k(x)$ does not depend on its position k, then the gradient-based learning complexity can be reduced to linear in N.*

To see how, dropping the explicit dependency of the subscript k in the definition of $\phi_k(\cdot)$, we maintain an auxiliary array $a_k = \sum_{x\in R_k} \phi(x)$ where $a_{K_\sigma} = \sum_{x\in X_{K_\sigma}} \phi(x)$ and $a_k = a_{k+1} + \sum_{x\in X_k} \phi(x)$ for $k < K_\sigma$. Clearly $a_{1:K_\sigma}$ can be computed in N time in a backward fashion. Thus, $g_k(\cdot)$ in Eq (3.9) can also be computed linearly via the relation $g_k(x) = \phi(x)/a_k$. This also implies that the total log-likelihood can also computed linearly in N.

Furthermore, the gradient of log-likelihood function can also be computed linearly in N. Given the likelihood function in Eq (3.4), using Eq (3.9), the log-likelihood function and its gradient, without explicit mention of the parameters, can be shown to be[10]

$$\mathscr{L} = \log p(X_1, \ldots, X_{K_\sigma}) = \sum_{k=1}^{K} \log \sum_{x\in X_k} g_k(x \mid X_{1:k-1}) = \sum_{k=1}^{K} \log \sum_{x\in X_k} \frac{\phi(x)}{a_k} \tag{3.10}$$

$$\partial\mathscr{L} = \sum_k \partial \log \sum_{x\in X_k} \phi(x) - \sum_k \partial \log a_k = \sum_k \frac{\sum_{x\in X_k} \partial\phi(x)}{\sum_{x\in X_k} \phi(x)} - \sum_k \frac{1}{a_k} \sum_{x\in R_k} \partial\phi(x) \tag{3.11}$$

It is clear that the first summation over k in the RHS of the last equation takes exactly N time since $\sum_{k=1}^{K} |X_k| = N$. For the second summation over k, it is more involved because both k and R_k can possibly range from 1 to N, so direct computation will cost at most $N(N-1)/2$ time. Similar to the case of a_k, we now maintain an 2-D auxiliary array[11] $b_k = \sum_{x\in R_k} \partial\phi(x)$, where $b_{K_\sigma} = \sum_{x\in X_{K_\sigma}} \partial\phi(x)$ and $b_k = b_{k+1} + \sum_{x\in X_k} \partial\phi(x)$ for $k < K_\sigma$. Thus, $b_{1:K_\sigma}$, and therefore the gradient $\partial\mathscr{L}$, can be computed in NF time in a backward fashion, where F is the number of parameters.

3.3.2 General State PMOP and MCMC Inference.

In the general case without any assumption on the form of the potential function $\Phi_k(\cdot)$ using only Eq (3.5) and (3.4), the log-likelihood function and its gradient, again without explicit mention of the model parameter, are:

$$\mathscr{L} = \log p(X_1) + \sum_{k=2}^{K_\sigma} \log p_k(X_k \mid X_{1:k-1}) \tag{3.12}$$

$$\partial\mathscr{L} = \sum_{k=1}^{K_\sigma} \partial \log \Phi_k(X_k) - \sum_{k=1}^{K_\sigma} \left\{ \sum_{S\in 2^{R_k}} p_k(S \mid X_{1:k-1}) \partial \log \Phi_k(S) \right\} \tag{3.13}$$

Clearly, both the distribution $p_k(X_k \mid X_{1:k-1})$ and the expectation $\sum_{S\in 2^{R_k}} p_k(S \mid X_{1:k-1}) \partial \log \Phi_k(S)$ are generally intractable to evaluate. In this paper, we make use of MCMC methods to approximate $p_k(X_k \mid X_{1:k-1})$. There are two natural choices: the Gibbs sampling and Metropolis-Hastings sampling. For Gibbs sampling we note that this problem can be viewed as sampling from a random field with binary variables. Each object is attached with binary variable whose states are either '*selected*' or '*not selected*' at kth stage. Thus, there will be $2^{N_k} - 1$ joint states in the random field, where we recall that N_k is the total number of remaining objects after $(k-1)$-th stage. The pseudo code for Gibbs and Metropolis-Hastings routines performed at kth stage is illustrated in Figure (2).

Finally, we note that in practical implementation of learning, we follow the proposal in [12] wherein for each local distribution at kth round we run the MCMC for *only* a few steps starting from the observed subset X_k. This technique is known to produce a biased estimate, but empirical evidences have so far indicated that the bias is small and the

[10] To be more precise, for $k = 1$ we define $X_{1:0}$ to be $\emptyset$.

[11] This is 2-D because we also need to index the parameters as well as the subsets.

Gibbs sampling

1. Randomly choose an initial subset X_k.
2. Repeat until stopping criteria met:
   - For each remaining object x at stage k, randomly select the object with the probability
   $$\frac{\Phi_k(X_k^{+x})}{\Phi_k(X_k^{+x})+\Phi_k(X_k^{-x})}$$
   where $\Phi_k(X_k^{+x})$ is the potential of the currently selected subset X_k if x is included and $\Phi_k(X_k^{-x})$ is when x is not.

Metropolis-Hastings sampling

1. Randomly choose an initial subset X_k
2. Repeat until stopping criteria met:
   - Randomly choose number of objects m, subject to $1 \leq m \leq N_k$.
   - Randomly choose m distinct objects from remaining set $R_k = \{X \setminus X_{1:k-1}\}$ to construct a new partition denoted by S.
   - Set $X_k \leftarrow S$ with the probability of
   $$\min\left\{1, \frac{\Phi_k(S)}{\Phi_k(X_k)}\right\}.$$

Figure 2: MCMC sampling approaches for PMOP in general case.

estimate is effective. Importantly, it is very fast compared to full sampling.

4 Applications with PMOP

4.1 Document Ranking. We now present a specific application of PMOP for the problem of document ranking. The ultimate goal after training is that, for each query the system needs to return a list of related objects and their *ranking*.[12] Slightly different from the standard rank setting in statistics, the objects in learning-to-rank problem are often not indexed (e.g. the identity of the object is not captured in any parameter). Instead, we will assume that for each query-object pair (q,x) we can extract a feature vector x^q. Model distribution specified in this way is thus *query-specific*. As a result, we are not interested in finding the single mode for the rank distribution over all queries[13], but in finding the rank mode for each query.

At the ranking phase, suppose for an unseen query q a list of $X^q = \left\{x_1^q, \ldots, x_{N_q}^q\right\}$ objects related to q is given[14]. The task is then to rank these objects in decreasing order of relevance w.r.t q. Enumerating over all possible rankings take an order of $N_q!$ time. Instead we would like to establish a *scoring function* $f(x^q, w) \in \mathbb{R}$ for the query q and each object x returned where w is now introduced as the parameter. Sorting can then be carried out much more efficiently in the complexity order of $N_q \log N_q$ instead of $N_q!$. The function specification can be a simple linear combination of features or more complicated form, such as a multilayer neural network.

In the practice of learning-to-rank, the dimensionality of feature vector x^q often remains the same across all queries, and since it is observed, we use PMOP described before to specify conditional model specific to q over the set of returned objects X^q as follows.

$$\begin{aligned} p(X^q|w) &= p(X_1^q, X_2^q, ..., X_{K_\sigma}^q \mid w) \\ &= P(X_1^q \mid w) \prod_{k=2}^{K_\sigma} p(X_k^q \mid X_{1:k-1}^q, w) \end{aligned} \tag{4.14}$$

We can see that Eq (4.14) has exactly the same form of Eq (3.4) specified for PMOP, but applied instead on the query-specific set of objects X^q and additional parameter w. During training, each query-object pair is labelled by a relevance score, which is typically an integer from the set $\{0,..,M\}$ where 0 means the object is irrelevant w.r.t the query q, and M means the object is highly relevant[15]. The value of M is typically much smaller than N_q, thus, the issue of *ties,* described at the beginning of this section, occur frequently. In a nutshell, for each training query q and its rated associated list of objects a PMOP is created. *The important parameterisation to note here is that the parameter w is shared across all queries*; and thus, enabling ranking for unseen queries in the future.

Using the scoring function $f(x,w)$ we specify the individual potential function $\phi(\cdot)$ in the exponential form:

$$\phi_k(x,w) = \exp\{f(x,w)\}$$

[12] We note a confusion that may arise here is that, although during training each training query q is supplied with a list of related objects and their *ratings,* during the ranking phase the system still needs to return a ranking over the list of related objects for an unseen query.

[13] This would lead to something like the *static* rank over all possible objects in the database - like those in Google's PageRank

[14] In document querying, for example, the list may consist of all documents which contain one or more query words

[15] Note that generally $K \leq M+1$ because there may be gaps in rating scales for a specific query.

The local potential function defined over for partition $\Phi_k\left(X_k^q\right)$ can now be explicitly constructed under full-decomposition (Subsection 3.3.1) and general case (Subsection 3.3.2) as respectively follows.

Full-decomposition. The partition potential is simply the mean of local potentials

$$\Phi_k\left(X_k^q\right) = \frac{1}{|X_k^q|}\sum_{x\in X_k^q}\exp\{f(x,w)\} \tag{4.15}$$

General case. While we have an entire freedom to define any form of partition potentials to meet our needs (e.g. including prior knowledge of partition sizes or the clustering properties), for this paper, we will use a simple version:

$$\Phi_k\left(X_k^q\right) = \exp\left\{\frac{1}{|X_k^q|}\sum_{x\in X_k^q} f(x,w)\right\} \tag{4.16}$$

Basically, this can be considered as a geometric mean of local potentials, as opposed to the arithmetic mean in the full-decomposition case.

The gradient of the log-likelihood function can also be computed efficiently. For simplicity, assume that the scoring function has the linear form $f(x^q,w) = w^\top x^q$. For full-decomposition, it can be shown to be:

$$\frac{\partial \log p\left(X_k^q \mid X_{1:k-1}^q\right)}{\partial w} = \sum_{x\in X_k^q}\frac{\phi_k(x,w)x}{\sum_{x\in X_k^q}\phi_k(x,w)} - \sum_{x\in R_k^q}\frac{\phi_k(x,w)x}{\sum_{x\in R_k^q}\phi_k(x,w)}$$

For the general case, the gradient of the log-likelihood function can be shown to be:

$$\frac{\partial \log p\left(X_k^q \mid X_{1:k-1}^q\right)}{\partial w} = \bar{x}_k^q - \sum_{S_k\in 2^{R_k^q}} p\left(S_k \mid X_{1:k-1}^q\right)\bar{s}_k^q$$

where

$$\bar{x}_k^q = \frac{1}{|X_k^q|}\sum_{x\in X_k} x^q, \qquad \bar{s}_k^q = \frac{1}{|S_k|}\sum_{x\in S_k} x^q$$

The quantity $p\left(X_k^q \mid X_{1:k-1}^q\right)$ can be interpreted as the probability that the subset X_k^q is chosen out of all possible subsets at stage k, and $\bar{x}_k$ is the centre of the chosen subset.

The expectation $\sum_{S_k} P(S_k \mid X_{1:k-1}^q)\bar{s}_k$ is expensive to evaluate, since there are $2^{N_k}-1$ possible subsets. Thus, we resort to MCMC techniques. We follow the suggestion in [12] to start the Markov chain from the observed subset X_k and run for a few iterations. The parameter update is stochastic

$$w \leftarrow w + \eta\sum_k\left(\bar{x}_k^q - \frac{1}{n}\sum_{l=1}^n \bar{s}_k^{(l)}\right)$$

where $\bar{s}_k^{(l)}$is the centre of the subset sampled at iteration l, and $\eta > 0$ is the learning rate, and n is number of samples. Typically we choose n to be small, e.g. $n = 1,2,3$.

4.2 Collaborative Filtering. We now present an application of our PMOP in collaborative filtering. Recall that in collaborative filtering, we are given a set of users, each of whom has expressed preferences over a set of items. The preferences can be in the form of a (partial) ranked list or a set of numerical ratings. The goal is to predict the preferences over unseen items for each user. Since the popular representation of preference is rating, most research in collaborative filtering so far has focused on predicting ratings instead of the more direct goal of predicting the ranked list of new items. There is, however, a refocus recently - we are now interested in modelling the ranking directly without going through the intermediate step of modelling the rating [18, 25, 28].

Let N be the number of users and M the number of items. To facilitate the interaction between an user u and an item i, the local potential function can be chosen as follows

$$\phi_k(x = i,u) = \exp\left\{\sum_{d=1}^D W_{ud}H_{di}\right\}$$

where $W \in \mathbb{R}^{N\times D}$ and $H \in \mathbb{R}^{D\times M}$, typically with $D \ll \min\{N,M\}$. This potential function can then be used for ranking items with respect to user u.

Different from the case of document ranking, the feature vector for each item $(H_{1i}H_{2i},..,H_{Di})$ is not given and must be discovered from the data. Second, the parameters are user-specific. As a result, the log-likelihood function is no longer concave in both W and H, although it is still concave in either W or H. Denote by $L_k^u = \log p\left(X_k^u \mid X_{1:k-1}^u\right)$. For full-decomposition, the gradient of the log-likelihood reads:

$$\frac{\partial L_k^u}{\partial W_{ud}} = \sum_{i\in X_k^u}\frac{\phi_k(i,u)H_{di}}{\sum_{j\in X_k^u}\phi_k(j,u)} - \sum_{i\in R_k^u}\frac{\phi_k(i,u)H_{di}}{\sum_{j\in R_k^u}\phi_k(j,u)}$$

$$\frac{\partial L_k^u}{\partial H_{di}} = W_{ud}\left[\frac{\phi_k(i,u)}{\sum_{j\in X_k^u}\phi_k(j,u)} - \frac{\phi_k(i,u)}{\sum_{j\in R_k^u}\phi_k(j,u)}\right]$$

For the general case, the situation is more involved, depending on the choice of the partition potentials which we omit here due to space constraints.

5 Discussion.

In our specific choice of the local distribution in Eq (3.5), we share the same idea with that of Plackett-Luce, in which

the probability of choosing the subset is proportional to the subset's worth, which is realised by the subset potential. In fact, when we limit the subset size to 1, i.e. there are no ties, the proposed model reduces to the well-known Plackett-Luce models.

The distribution of the full-decomposition case has an interesting interpretation. From Eq 3.8 the local partition distribution can be rewritten as

$$p(X_k \mid X_{1:k-1}) = \frac{1}{C|X_k|} \sum_{x \in X_k} \frac{\phi_k(x)}{\sum_{x' \in R_k} \phi_k(x')}$$

Since $\phi_k(x)/\sum_{x' \in R_k} \phi_k(x')$ is the probability of choosing x as the top object at stage k, $p(X_k \mid X_{1:k-1})$ can be interpreted as the probability of choosing any member in the subset X_k as the top object, up to a multiplicative constant. Thus, the full-decomposition offers a simple way to model the inherent uncertainty in the choices when ties occur.

It is worth mentioning that the factorisation in Eq (3.4) and the choice of local distribution in Eq (3.5) are not unique. In fact, the chain-rule can be applied to any sequence of choices. For example, we can factorise in a backward manner

$$p(X_1, \ldots, X_{K_\sigma}) = p_1(X_{K_\sigma}) \prod_{k=1}^{K_\sigma - 1} p_k(X_k \mid X_{k+1:K_\sigma}) \quad (5.17)$$

where $X_{k+1:K_\sigma}$ is a shorthand for $\{X_{k+1}, X_{k+2}, ..., X_{K_\sigma}\}$. Interestingly, we can interpret this reverse process as *subset elimination*: First we choose to eliminate the worst subset, then the second worst, and so on. This line of reasoning has been discussed in [9] but it is limited to 1-element subsets. However, if we are free to choose the parameterisation of $p_k(X_k \mid X_{k+1:K_\sigma})$ as we have done for $p_k(X_k \mid X_{1:k-1})$ in Eq (3.5), there is no guarantee that the forward and backward factorisation admits the same distribution.

Our model can be placed into the framework of probabilistic graphical models (e.g. see [16]). Recall that in standard probabilistic graphical models, we have a set of variables, each of which receives values from a fixed set of states. Generally, variables and states are orthogonal concepts, and the *state space* of a variable do not explicitly depends on the states of other variables[16]. In our setting, the objects play the role of the variables, and their memberships in the subsets are their states. However, since there are exponentially many subsets, enumerating the state spaces as in standard graphical models is not possible. Instead, we can consider the ranks of the subsets in the list as the states, since the ranks only range from 1 to N. Different from the standard graphical models, the variables and the states are not always independent, e.g. when the subset sizes are limited to 1, then the state assignments of variables are mutually exclusive, since for each position, there is only one object. Probabilistic graphical models are generally directed (such as Bayesian networks) or undirected (such as Markov random fields), and our PMOP can be thought as a directed model. The undirected setting is also of great interest, but it is beyond the scope of this paper.

With respect to tie handling, most previous work focuses on pairwise models. The basic idea is to assign some probability mass for the event of ties [7][11][24]. For instance, denote by $x_i \succ x_j$ the preference of x_i over x_j, and by $x_i \approx x_j$ the tie between the two objects, Rao and Kupper [24] proposed the following models

$$\begin{aligned} P(x_i \succ x_j) &= \frac{\phi(x_i)}{\phi(x_i) + \theta\phi(x_j)} \\ P(x_i \approx x_j) &= \frac{(\theta^2 - 1)\phi(x_i)\phi(x_j)}{[\phi(x_i) + \theta\phi(x_j)]\,[\theta\phi(x_i) + \phi(x_j)]} \end{aligned}$$

where $\theta \geq 1$ is the parameter to control the contribution of ties. When $\theta = 1$, the model reduces to the standard Bradley-Terry model [1] . This method of ties handling is further studied in [31] in the context of learning to rank. Another method is introduced in [7], where the probability masses are defined as

$$\begin{aligned} P(x_i \succ x_j) &= \frac{\phi(x_i)}{\phi(x_i) + \phi(x_j) + \nu\sqrt{\phi(x_i)\phi(x_j)}} \\ P(x_i \approx x_j) &= \frac{\nu\sqrt{\phi(x_i)\phi(x_j)}}{\phi(x_i) + \phi(x_j) + \nu\sqrt{\phi(x_i)\phi(x_j)}} \end{aligned}$$

where $\nu \geq 0$. The applications of these two tie-handling models to learning to rank are detailed in Appendix C.

For ties of multiple objects, we can create a group of objects, and work directly on groups. For example, let X_i and X_j be two sport teams, the pairwise team ordering can be defined using the Bradley-Terry model as

$$P(X_i \succ X_j) = \frac{\sum_{x \in X_i} \phi(x)}{\sum_{x \in X_i} \phi(x) + \sum_{s \in X_j} \phi(s)}$$

The extension of the Plackett-Luce model to multiple groups has been discussed in [14]. However, we should emphasize that this setting is not the same as ours, because the partitioning is known in advance, and the groups behave just like standard super-objects. Our setting, on the other hand, assumes no fixed partitioning, and the membership of the objects in a group is arbitrary.

Another way to deal with ties is to create an 'equivalent permutation set' (e.g. see [17]) from ties and then train with the full-rank algorithms. The idea is to minimise the min loss over the set in a fashion similar to multiple-instance learning. This is different from our work, however, since we are focusing on modelling probability of the set out of all possible set partitionings.

[16] Note that, this is different from saying the states of variables are independent.

6 Evaluation.

In this section we present evaluation results of our proposed PMOP on two tasks: document ranking on Web data and collaborative filtering on movie data.

Two performance metrics are reported: the Normalised Discounted Cumulative Gain at position T (NDCG@T), and the Expected Reciprocal Rank (ERR) [4]. NDCG@T metric is defined as

$$\text{NDCG@}T \quad = \frac{1}{\kappa(T)} \quad \sum_{i=1}^{T} \frac{2^{r_i}-1}{\log_2(1+i)}$$

where r_i is the relevance judgment of the document at position i, $\kappa(T)$ is a normalisation constant to make sure that the gain is 1 if the rank is correct. The ERR is defined as

$$\text{ERR} \quad = \quad \sum_i \frac{1}{i} V(r_i) \prod_{j=1}^{i-1} (1 - V(r_j)) \quad \text{where } V(r) = \frac{2^r - 1}{16}.$$

	ERR	NDCG@1	NDCG@5
Rank Regress	0.4882	0.683	0.6672
RankNet	0.4919	0.6903	0.6698
Ranking SVM	0.4868	0.6797	0.6662
ListMLE	0.4955	0.6993	0.6705
PairTies-D	0.4941	0.6944	0.6725
PairTies-RK	0.4946	0.6970	0.6716
PMOP-FD	0.5038	0.7137	0.6762
PMOP-Gibbs	0.5037	0.7105	**0.6792**
PMOP-MH	**0.5045**	**0.7139**	0.6790

Table 1: Performance measured in ERR and NDCG@T. PairTies-D and PairTies-RK are the Davidson method and Rao-Kupper method for ties handling, respectively. PMOP-FD is the PMOP with full-decomposition, and PMOP-Gibbs/MH is the PMOP with Gibbs/Metropolis-Hasting sampling (see Section 4.1 for a description).

6.1 Document Ranking. The data is from Yahoo! learning to rank challenge [30]. This is a subset of the real dataset used to train Yahoo! search engines, and is currently one of the largest datasets available for research[17]. The data contains the groundtruth labels of 473,134 documents returned from 19,944 queries. The label is the relevance judgment from 0 (irrelevant) to 4 (perfectly relevant). Features for each document-query pairs are also supplied by Yahoo!, and there are 519 unique features. We first normalised the features across the whole training set to have mean 0 and standard deviation 1.

We split the data into two sets: the training set contains roughly 90% queries, and the test set is the remaining 10%. For comparison, we implement several well-known methods, including RankNet [2], Ranking SVM [15] and ListMLE [29]. The RankNet and Ranking SVM are pairwise methods, and they differ on the choice of loss functions, i.e. logistic loss for the RankNet and hinge loss for the Ranking SVM[18]. Similarly, choosing quadratic loss gives us a rank regression method, which we will call Rank Regress (see Appendix B for more details). From rank modelling point of view, the RankNet is essentially the Bradley-Terry model [1] applied to learning to rank. Likewise, the ListMLE is essentially the Plackett-Luce model, which has been argued to be one of the best performing methods [17].

We also implement two variants of the Bradley-Terry model with ties handling, one by Rao-Kupper [24] (denoted by PairTies-RK; this also appears to be implemented in [31] under the functional gradient setting) and another by Davidson [7] (denoted by PairTies-D; and this is the first time the Davidson method is applied to learning to rank). See Appendix C for implementation details.

There are three methods resulted from our framework (see description in Section 4.1). The first is the PMOP with full-decomposition (denoted by PMOP-FD), the second is with Gibbs sampling (denoted by PMOP-Gibbs), and the third is with Metropolis-Hastings sampling (denoted by PMOP-MH).

For those pairwise methods without ties handling, we simply ignore the tied document pairs. For the ListMLE, we simply sort the documents within a query by relevance scores, and those with ties are ordered according to the sorting algorithm. All methods, except for PMOP-Gibbs/MH, are trained using the Limited Memory Newton Method known as L-BFGS. The L-BFGS is stopped if the relative improvement over the loss is less than 10^{-5} or after 100 iterations. As the PMOP-Gibbs/MH are stochastic, we run the MCMC for a few steps per query, then update the parameter using the Stochastic Gradient Ascent. The learning rate is fixed to 0.1, and the learning is stopped after 1,000 iterations.

The results are reported in Table 1. It can be seen that modelling ties are beneficial, as PairTies-D and PairTies-RK perform better than the RankNet (without ties handling), and our **PMOP** variants improve over ListMLE, despite of the simplicity in the potential function choices in Equations 4.15 and 4.16. The PMOP-MH wins over the best performing baseline, ListMLE, by 1.82% according to the ERR metric. In our view, this is a significant improvement given the scope of the dataset. We note that the difference in the top 20 in the

[17] This is much larger than the commonly used LETOR 3.0 and 4.0 datasets. In the preparation of this manuscript, we learnt that Microsoft had released two large sets of comparable size with that of Yahoo! but due to time constraint, we do not report the results here.

[18] Strictly speaking, RankNet makes use of neural networks as the scoring function, but the overall loss is still logistic, and for simplicity, we use simple perceptron.

Pairwise models	PMOP/ListMLE
$\max\{\mathcal{O}(N^2), \mathcal{O}(NF)\}$	$\mathcal{O}(NF)$

Table 2: Learning complexity of models, where F is the number of unique features. For pairwise models, see Appendix B for the details.

leaderboard[19] of the Yahoo! challenge is just 1.56%.

As for training time, the PMOP-FD is numerically the fastest method. Theoretically, it has the linear complexity similar to ListMLE. All other pairwise methods are quadratic in query size, and thus numerically slower. The PMOP-Gibbs/MH is also linear in the query size, by a constant factor that is determined by the number of iterations. See Table 2 for a summary.

6.2 Collaborative Filtering. In this experiment, we study how the handling of ties improves the fitness of the Plackett-Luce model. We use the MovieLens data[20], which has $100,000$ ratings assigned by 943 users to a set of 1682 movies. The ratings are integer in the 5-star scale. In this setting, the user plays the role of the query, and the movies the role of documents. For exact evaluation of likelihood, we use the full-decomposition setting. For each user, we randomly select 10, 20 and 50 movies for training, and the rest for testing. To ensure that there are at least 10 test movies for each user, we remove those users with less than 20, 30 and 60 ratings, respectively. Once the training is completed, the training data is thrown away, and we measure the results on the test data only. For the Plackett-Luce model, we first sort the ratings by each user in the training data, and then reorder them according to a sort algorithm.

For comparison we run the CoFi^{RANK}-NDCG algorithm of [28] on the data with the code provided by the authors[21]. For all algorithms, we set the feature dimensionality to 5. The matrices W and H are initialised randomly in the range $[0, 0.5]$. The experimental results are reported in Table 3. Note that we do not impose any regularisation on the Plackett-Luce and our PMOP since the goal is to estimate the likelihood of the test data. It can be seen that the handling of ties invariably improve over the Plackett-Luce model which incorrectly assumes complete ranks in the data. The predictive performance over test data completes well with the CoFi^{RANK}-NDCG, which is perhaps the best-known algorithm in this class of problems.

[19] Our result on a more deliberate design of features (which is not the primary concern of this paper) was submitted to the Yahoo! challenge and obtained a position in the top 4% over 1055 teams, given that our main purpose was to propose a new theoretical and useful model.

[20] http://grouplens.org/node/73

[21] The code is available at: http://www.cofirank.org/downloads. We implement a simple wrapper to compute the ERR and NDCG scores (at various positions), which are not available in the code.

		PL	PMOP-FD	CoFi^{RANK}
N=10	LL	-430.042	**-159.534**	-
	ERR	0.683	0.718	**0.721**
	NDCG@1	0.556	0.604	0.627
	NDCG@5	0.583	**0.617**	0.593
N=20	LL	-463.212	**-179.918**	-
	ERR	0.714	**0.753**	0.715
	NDCG@1	0.594	**0.660**	0.598
	NDCG@5	0.602	**0.648**	0.608
N=50	LL	-505.365	**-202.991**	-
	ERR	0.739	**0.764**	0.703
	NDCG@1	0.633	**0.670**	0.576
	NDCG@5	0.630	**0.655**	0.591

Table 3: Results on movie ranking, where LL=log-likelihood of the test data.

7 Conclusions.

Addressing the general problem of ranking with ties, we have proposed a generative probabilistic model, with suitable parameterisation to address the problem complexity. We present efficient algorithms for learning and inference. We evaluate the proposed models on two problems: the first is document ranking with the data from the recently held Yahoo! challenge and the second is collaborative filtering with the well-studied MovieLens dataset. Our experimental results demonstrate that the models are competitive against well-known rivals designed specifically for the problems.

There are several promising directions to further this work, including the introduction of Bayesian modelling, extending to latent aspects of the rank data, evaluating on a variety of approximate inference and learning methods.

References

[1] R.A. Bradley and M.E. Terry, *Rank analysis of incomplete block designs*, Biometrika, 39 (1952), pp. 324–345.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, *Learning to rank using gradient descent*, In Proc. of ICML, (2005).

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

[4] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, *Expected reciprocal rank for graded relevance*, In Proceeding of the 18th ACM conference on Information and knowledge management, (2009) pp. 621–630.

[5] W. Chu and Z. Ghahramani, *Gaussian processes for ordinal regression*, Journal of Machine Learning Research, 6 (2006).

[6] D. Cossock and T. Zhang, *Statistical analysis of Bayes optimal subset ranking*, IEEE Transactions on Information Theory, 54 (2008) pp. 5140–5154.

[7] R.R. Davidson, *On extending the Bradley-Terry model to accommodate ties in paired comparison experiments*, Journal of the American Statistical Association, 65 (1970) pp. 317–328.

[8] P. Diaconis, *A generalization of spectral analysis with application to ranked data*, The Annals of Statistics, (1989) pp. 949–979.

[9] M.A. Fligner and J.S. Verducci, *Multistage ranking models*, Journal of the American Statistical Association, 83 (1988) pp. 892–901.

[10] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer, *An efficient boosting algorithm for combining preferences*, Journal of Machine Learning Research, 4 (2004), pp. 933–969.

[11] W.A. Glenn and H.A. David, *Ties in paired-comparison experiments using a modified Thurstone-Mosteller model*, Biometrics, 16 (1960) pp. 86–109.

[12] G.E. Hinton, *Training products of experts by minimizing contrastive divergence*, Neural Computation, 14 (2002) pp. 1771–1800.

[13] J. Huang, C. Guestrin, and L. Guibas, *Fourier theoretic probabilistic inference over permutations*, The Journal of Machine Learning Research, 10 (2009) pp. 997–1070.

[14] T.K. Huang, R.C. Weng, and C.J. Lin, *Generalized Bradley-Terry models and multi-class probability estimates*, The Journal of Machine Learning Research, 7 (2006).

[15] T. Joachims, *Optimizing search engines using clickthrough data*. In Proc. of SIGKDD, (2002) pp. 133–142.

[16] S.L. Lauritzen, *Graphical Models*, Oxford Science Publications, 1996.

[17] T.Y. Liu, *Learning to rank for information retrieval*, Foundations and Trends in Information Retrieval, 3 (2009) pp. 225–331.

[18] N.N. Liu, M. Zhao, and Q. Yang, *Probabilistic latent preference analysis for collaborative filtering*, In Proceeding of the 18th ACM conference on Information and knowledge management (2009), pp. 759–766.

[19] R.D. Luce, *Individual choice behavior*, Wiley New York, 1959.

[20] C.L. Mallows, *Non-null ranking models I*, Biometrika, 44 (1957) pp. 114–130.

[21] J.I. Marden, *Analyzing and modeling rank data*, Chapman & Hall/CRC, 1995.

[22] M. Muresan, *A concrete approach to classical analysis*, Springer Verlag, 2008.

[23] R.L. Plackett, *The analysis of permutations*, Applied Statistics, (1975) pp. 193–202.

[24] P.V. Rao and L.L. Kupper, *Ties in paired-comparison experiments: A generalization of the Bradley-Terry model*, Journal of the American Statistical Association, (1967) pp. 194–204.

[25] Y. Shi, M. Larson, and A. Hanjalic, *List-wise learning to rank with matrix factorization for collaborative filtering*, In Proceedings of the fourth ACM conference on Recommender systems, (2010), pp. 269–272.

[26] J.H. van Lint and R.M. Wilson, *A course in combinatorics*, Cambridge University Press, 1992.

[27] M.N. Volkovs and R.S. Zemel, *BoltzRank: learning to maximize expected ranking gain*, In Proceedings of the 26th Annual International Conference on Machine Learning, (2006) ACM New York, NY, USA.

[28] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola, *CoFi^{RANK}-maximum margin matrix factorization for collaborative ranking*, Advances in neural information processing systems, 20 (2008), pp. 1593–1600.

[29] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li, *Listwise approach to learning to rank: theory and algorithm*, In Proc. of ICML, (2008) pp. 1192–1199.

[30] Yahoo! *Yahoo! learning to rank challenge*, http://learningtorankchallenge.yahoo.com, 2010.

[31] K. Zhou, G.R. Xue, H. Zha, and Y. Yu, *Learning to rank with ties*. In Proc. of SIGIR, (2008) pp. 275–282.

A Computing C.

Let us calculate the constant C in Eq (3.7). Let us rewrite the equation for ease of comprehension

$$\sum_{S\in 2^{R_k}} \frac{1}{|S|} \sum_{x\in S} \phi_k(x) = C \times \sum_{x\in R_k} \phi_k(x)$$

where 2^{R_k} is the power set with respect to the set R_k, or the set of all non-empty subsets of R_k. Equivalently

$$C = \sum_{S\in 2^{R_k}} \frac{1}{|S|} \sum_{x\in S} \frac{\phi_k(x)}{\sum_{x\in R_k} \phi_k(x)}$$

If all objects are the same, then this can be simplified to

$$\begin{aligned} C &= \sum_{S\in 2^{R_k}} \frac{1}{|S|} \sum_{x\in S} \frac{1}{N_k} = \frac{1}{N_k} \sum_{S\in 2^{R_k}} 1 \\ &= \frac{2^{N_k}-1}{N_k} \end{aligned}$$

where $N_k = |R_k|$. In the last equation, we have made use of the fact that $\sum_{S\in 2^{R_k}} 1$ is the number of all possible non-empty subsets, or equivalently, the size of the power set, which is known to be $2^{N_k}-1$. One way to derive this result is the imagine a collection of N_k variables, each has two states: '*selected*' and '*not selected*', where 'selected' means the object belongs to a subset. Since there are 2^{N_k} such configurations over all states, the number of non-empty subsets must be $2^{N_k}-1$.

For arbitrary objects, let us examine the the probability that the object x belong to a subset of size m, which is $\frac{m}{N_k}$. Recall from standard combinatorics that the number of m-element subsets is the binomial coefficient $\binom{N_k}{m}$, where $1 \le m \le N_k$, and . Thus the number of times an object appears in any m-subset is $\binom{N_k}{m}\frac{m}{N_k}$. Taking into account that this number is weighted down by m (i.e. $|S|$ in Eq (3.7)), the the contribution towards C is then $\binom{N_k}{m}\frac{1}{N_k}$. Finally, we can compute the constant C, which is the weighted number of

times an object belongs to any subset of any size, as follows

$$\begin{aligned} C &= \sum_{m=1}^{N_k} \binom{N_k}{m} \frac{1}{N_k} = \frac{1}{N_k} \sum_{m=1}^{N_k} \binom{N_k}{m} \\ &= \frac{2^{N_k}-1}{N_k} \end{aligned}$$

We have made use of the known identity $\sum_{m=1}^{N_k} \binom{N_k}{m} = 2^{N_k}-1$.

B Pairwise Losses.

Let $\delta_{ij}(w) = \phi(x_i,w) - \phi(x_j,w)$, the pairwise losses are

$$\ell(x_i \succ x_j; w) = \begin{cases} \log(1+\exp\{-\delta_{ij}(w)\}) & \text{(RankNet)} \\ \max\{0, 1-\delta_{ij}(w)\} & \text{(Ranking SVM)} \\ (1-\delta_{ij}(w))^2 & \text{(Rank Regress)} \end{cases}$$

C Learning the Paired Ties Models.

This section describes the details of learning the paired ties models discussed in Section 5.

Rao-Kupper method. Recall that the Rao-Kupper model defines the following probability masses

$$\begin{aligned} P(x_i \succ x_j; w) &= \frac{\phi(x_i)}{\phi(x_i) + \theta\phi(x_j)} \\ P(x_i \approx x_j; w) &= \frac{(\theta^2-1)\phi(x_i)\phi(x_j)}{[\phi(x_i) + \theta\phi(x_j)]\,[\theta\phi(x_i) + \phi(x_j)]} \end{aligned}$$

where $\theta \geq 1$ is the ties factor and w is the model parameter. Note that $\phi(.)$ is also a function of w, which we omit here for clarity. For ease of unconstrained optimisation, let $\theta = 1+e^{\alpha}$ for $\alpha \in \mathbb{R}$. In learning, we want to estimate both α and w. Let

$$\begin{aligned} P_i &= \frac{\phi(x_i)}{\phi(x_i) + (1+e^{\alpha})\phi(x_j)} \\ P_j^* &= \frac{\phi(x_j)}{\phi(x_i) + (1+e^{\alpha})\phi(x_j)} \\ P_i^* &= \frac{\phi(x_i)}{(1+e^{\alpha})\phi(x_i) + \phi(x_j)} \\ P_j &= \frac{\phi(x_j)}{(1+e^{\alpha})\phi(x_i) + \phi(x_j)} \end{aligned}$$

Taking partial derivatives of the log-likelihood gives

$$\begin{aligned} \frac{\partial \log P(x_i \succ x_j; w)}{\partial w} &= (1-P_i)\frac{\partial \log\phi(x_i,w)}{\partial w} \\ &\quad -(1+e^{\alpha})P_j\frac{\partial \log\phi(x_j,w)}{\partial w} \\ \frac{\partial \log P(x_i \succ x_j; w)}{\partial \alpha} &= -P_j e^{\alpha} \\ \frac{\partial \log P(x_i \approx x_j; w)}{\partial w} &= (1-P_i-(1+e^{\alpha})P_i^*)\frac{\partial \log\phi(x_i,w)}{\partial w} \\ &+ (1-P_j-(1+e^{\alpha})P_j^*)\frac{\partial \log\phi(x_j,w)}{\partial w} \\ \frac{\partial \log P(x_i \approx x_j; w)}{\partial \alpha} &= \left(\frac{2(1+e^{\alpha})}{(1+e^{\alpha})^2-1} - P_i^* - P_j^*\right)e^{\alpha} \end{aligned}$$

Davidson method. Recall that in the Davidson method the probability masses are defined as

$$\begin{aligned} P(x_i \succ x_j; w) &= \frac{\phi(x_i)}{\phi(x_i)+\phi(x_j)+\nu\sqrt{\phi(x_i)\phi(x_j)}} \\ P(x_i \approx x_j; w) &= \frac{\nu\sqrt{\phi(x_i)\phi(x_j)}}{\phi(x_i)+\phi(x_j)+\nu\sqrt{\phi(x_i)\phi(x_j)}} \end{aligned}$$

where $\nu \geq 0$. Again, for simplicity of unconstrained optimisation, let $\nu = e^{\beta}$ for $\beta \in \mathbb{R}$. Let

$$\begin{aligned} P_i &= \frac{\phi(x_i)}{\phi(x_i)+\phi(x_j)+e^{\beta}\sqrt{\phi(x_i)\phi(x_j)}} \\ P_j &= \frac{\phi(x_j)}{\phi(x_i)+\phi(x_j)+e^{\beta}\sqrt{\phi(x_i)\phi(x_j)}} \\ P_{ij} &= \frac{e^{\beta}\sqrt{\phi(x_i)\phi(x_j)}}{\phi(x_i)+\phi(x_j)+e^{\beta}\sqrt{\phi(x_i)\phi(x_j)}} \end{aligned}$$

Taking derivatives of the log-likelihood gives

$$\begin{aligned} \frac{\partial \log P(x_i \succ x_j; w)}{\partial w} &= (1-P_i-0.5P_{ij})\frac{\partial \log\phi(x_i,w)}{\partial w} \\ &\quad -(P_i+0.5P_{ij})\frac{\partial \log\phi(x_j,w)}{\partial w} \\ \frac{\partial \log P(x_i \succ x_j; w)}{\partial \beta} &= -P_{ij} \\ \frac{\partial \log P(x_i \approx x_j; w)}{\partial w} &= (0.5-P_i-0.5P_{ij})\frac{\partial \log\phi(x_i,w)}{\partial w} \\ &\quad +(0.5-P_j-0.5P_{ij})\frac{\partial \log\phi(x_j,w)}{\partial w} \\ \frac{\partial \log P(x_i \approx x_j; w)}{\partial \beta} &= 1-P_{ij}. \end{aligned}$$