# Directly Optimizing Evaluation Measures in Learning to Rank

Jun Xu, Tie-Yan Liu
Microsoft Research Asia
No. 49 Zhichun Road,
Beijing, China 100190
{junxu,tyliu}@microsoft.com

Min Lu
Nankai University
No. 94 Weijin Road,
Tianjin, China 300071
lumin@nankai.edu.cn

Hang Li, Wei-Ying Ma
Microsoft Research Asia
No. 49 Zhichun Road,
Beijing, China 100190
{hangli,wyma}@microsoft.com

## ABSTRACT

One of the central issues in learning to rank for information retrieval is to develop algorithms that construct ranking models by directly optimizing evaluation measures used in information retrieval such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). Several such algorithms including SVM$^{map}$ and AdaRank have been proposed and their effectiveness has been verified. However, the relationships between the algorithms are not clear, and furthermore no comparisons have been conducted between them. In this paper, we conduct a study on the approach of directly optimizing evaluation measures in learning to rank for Information Retrieval (IR). We focus on the methods that minimize loss functions upper bounding the basic loss function defined on the IR measures. We first provide a general framework for the study and analyze the existing algorithms of SVM$^{map}$ and AdaRank within the framework. The framework is based on upper bound analysis and two types of upper bounds are discussed. Moreover, we show that we can derive new algorithms on the basis of this analysis and create one example algorithm called PermuRank. We have also conducted comparisons between SVM$^{map}$, AdaRank, PermuRank, and conventional methods of Ranking SVM and RankBoost, using benchmark datasets. Experimental results show that the methods based on direct optimization of evaluation measures can always outperform conventional methods of Ranking SVM and RankBoost. However, no significant difference exists among the performances of the direct optimization methods themselves.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

## General Terms

Algorithms, Experimentation, Theory

## Keywords

Evaluation measure, Learning to rank, Information retrieval

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*SIGIR'08,* July 20–24, 2008, Singapore.
Copyright 2008 ACM 978-1-60558-164-4/08/07 ...$5.00.

## 1. INTRODUCTION

Learning to rank for Information Retrieval (IR) is a problem as follows. In learning, a ranking model is constructed with training data that consists of queries, their corresponding retrieved documents, and relevance levels provided by human annotators. In ranking, given a new query, the retrieved documents are ranked by using the trained ranking model.

In IR, ranking results are generally evaluated in terms of evaluation measures such as Mean Average Precision (MAP) [1] and Normalized Discounted Cumulative Gain (NDCG) [14]. Ideally, a learning algorithm trains a ranking model by optimizing the performance in terms of a given evaluation measure. In this way, higher accuracy in ranking is expected. However, this is usually difficult due to the non-continuous and non-differentiable nature of IR measures.

Many learning to rank algorithms proposed typically minimize a loss function loosely related to the IR measures. For example, Ranking SVM [13] and RankBoost [10] minimize loss functions based on classification errors in document pairs.

Recently, researchers have developed several new algorithms that manage to directly optimize the performance in terms of IR measures. The effectiveness of these methods have also been verified. From the viewpoint of loss function optimization, these methods fall into three categories. First, one can minimize upper bounds of the basic loss function defined on the IR measures [30, 17, 27]. Second, one can approximate the IR measures with functions that are easy to handle [7, 23]. Third, one can use specially designed technologies for optimizing the non-smooth IR measures [3, 8].

There are open questions regarding the *direct optimization approach*. (1) Is there a general theory that can guide the development of new algorithms? (2) What is the relationship between existing methods? (3) Which direct optimization method empirically performs best?

In this paper, we conduct a study on direct optimization of IR measures in learning to rank and answer the above questions. Specifically, we focus on the first category of methods that minimize loss functions upper bounding the basic loss function defined on the IR measures. This has become one of the hottest research topics in learning to rank.

(1) We conduct a general analysis of the approach. We indicate that direct optimization of IR measures amounts to minimizing different loss functions based on the measures. We first introduce one *basic* loss function, which is directly defined on the basis of IR measures, and indicate that there are two types of upper bounds on the basic loss function. We refer to them as type one bound and type two bound, respectively. Minimizing the two types of upper bounds leads to different learning algorithms. With this analysis, different algorithms can be easily studied and compared. More-

over, new algorithms can be easily derived. As an example, we create a new algorithm called PermuRank.

(2) We show that the existing algorithms of AdaRank and SVM$^{map}$ manage to minimize loss functions which are type one upper bound and type two upper bound, respectively.

(3) We compare the performances of the existing direct optimization methods of AdaRank and SVM$^{map}$ using several benchmark datasets. Experimental results show that the direct optimization methods of SVM$^{map}$, AdaRank, and PermuRank can always improve upon the baseline methods of Ranking SVM and RankBoost. Furthermore, the direct optimization methods themselves can work equally well.

The rest of the paper is organized as follows. After a summary of related work in Section 2, we formally describe the problem of learning to rank for Information Retrieval in Section 3. In section 4, we propose a general framework for directly optimizing evaluation measures. Two existing algorithms of SVM$^{map}$ and AdaRank, and a new algorithm PermuRank are analyzed and discussed within this framework. Section 5 reports our experimental results and Section 6 concludes this paper.

## 2. RELATED WORK

The key problem for document retrieval is ranking, specifically, to create a ranking model that can sort documents based on their relevance to the given query. Traditional ranking models such as BM25 [22] and Language Models for Information Retrieval (LMIR) [20, 16] only have a few parameters to tune. As the ranking models become more sophisticated (with more features) and more labeled data become available, how to tune or train a ranking model becomes a challenging issue.

In recent years, methods of learning to rank have been applied to ranking model construction and promising results have been obtained. Learning to rank is to automatically create a ranking model by using labeled training data and machine learning techniques. Several approaches have been proposed. The pairwise approach transforms the ranking problem into binary classification on document pairs. Typical methods include Ranking SVM [13, 15], RankBoost [10], and RankNet [4]. For other methods belonging to the approach, refer to [12, 29, 6, 25, 21, 30, 24, 31]. The methods of Ranking SVM, RankBoost, and RankNet minimize loss functions that are loosely related to the evaluation measures such as MAP and NDCG.

Recently, the approach of directly optimizing the performance in terms of IR measures has also been proposed. There are three categories:

First, one can minimize loss functions upper bounding the basic loss function defined on the IR measures. For example, SVM$^{map}$ [30] minimizes a hinge loss function, which upper bounds the basic loss function based on Average Precision. AdaRank [27] minimizes an exponential loss function upper bounding the basic loss function. (See also [17].)

Second, one can approximate the IR measures with easy-to-handle functions. For example, the work in [23] proposes a smoothed approximation to NDCG [14]. (See also [7].)

Third, one can use specially designed technologies for optimizing non-smooth IR measures. For example, LambdaRank [3] implicitly minimizes a loss function related to IR measures. Genetic Programming (GP) is also used to optimize IR measures [2]. For example, [8] proposed a specifically designed GP for learn a ranking model for IR. (See also [28, 9, 19]).

In this paper, we focus on the first category and take SVM$^{map}$ and AdaRank as examples of existing methods.

## 3. LEARNING TO RANK

Learning to rank for Information Retrieval is a problem as follows. In retrieval (testing), given a query, the system returns a ranked list of documents in descending order of their relevance scores. In learning (training), a number of queries and their corresponding retrieved documents are given. Furthermore, the labels of the documents with respect to the queries are also provided. The labels represent ranks (i.e., categories in a total order). The objective of learning is to construct a ranking model that achieves the best result on test data in the sense of minimization of a loss function. Ideally the loss function is defined directly on the IR measure used in testing.

Suppose that $Y = \{r_1, r_2, \cdots, r_\ell\}$ is the set of ranks, where $\ell$ denotes the number of ranks. There exists a total order between the ranks $r_\ell \succ r_{\ell-1} \succ \cdots \succ r_1$, where $\succ$ denotes the order. Suppose that $Q = \{q_1, q_2, \cdots, q_m\}$ is the set of queries in training. Each query $q_i$ is associated with a list of retrieved documents $\mathbf{d}_i = \{d_{i1}, d_{i2}, \cdots, d_{i,n(q_i)}\}$ and a list of labels $\mathbf{y}_i = \{y_{i1}, y_{i2}, \cdots, y_{i,n(q_i)}\}$, where $n(q_i)$ denotes the sizes of lists $\mathbf{d}_i$ and $\mathbf{y}_i$, $d_{ij} \in \mathbf{D}$ denotes the $j^{th}$ document in $\mathbf{d}_i$, and $y_{ij} \in Y$ denotes the label of document $d_{ij}$. A feature vector $\phi(q_i, d_{ij})$ is created from each query-document pair $(q_i, d_{ij}), i = 1, 2, \cdots, m; j = 1, 2, \cdots, n(q_i)$. The training set is denoted as $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$.

Let the documents in $\mathbf{d}_i$ be identified by the integers $\{1, 2, \cdots, n(q_i)\}$. We define permutation $\pi_i$ on $\mathbf{d}_i$ as a bijection from $\{1, 2, \cdots, n(q_i)\}$ to itself. We use $\Pi_i$ to denote the set of all possible permutations on $\mathbf{d}_i$, and use $\pi_i(j)$ to denote the position of item $j$ (i.e., $d_{ij}$). Ranking is nothing but to select a permutation $\pi_i \in \Pi_i$ for the given query $q_i$ and the associated list of documents $\mathbf{d}_i$ using the ranking model.

The ranking model is a real valued function of features. There are two types of ranking models. We refer to them as $f$ and $F$ respectively.

Ranking model $f$ is a document level function, which is a linear combination of the features in a feature vector $\phi(q_i, d_{ij})$:

$$f(q_i, d_{ij}) = w^\top \phi(q_i, d_{ij}), \tag{1}$$

where $w$ denotes the weight vector. In ranking for query $q_i$ we assign a score to each of the documents using $f(q_i, d_{ij})$ and sort the documents based on their scores. We obtain a permutation denoted as $\tau_i$.

Ranking model $F$ is a query level function. We first introduce a query level feature vector for each triple of $q_i$, $\mathbf{d}_i$ and $\pi_i$, denoted as $\Phi(q_i, \mathbf{d}_i, \pi_i)$. We calculate $\Phi$ by linearly combining the feature vectors $\phi$ of query-document pairs for $q_i$:

$$\Phi(q_i, \mathbf{d}_i, \pi_i) = \frac{1}{n(q_i) \cdot (n(q_i) - 1)} \sum_{k,l:k<l} [z_{kl}(\phi(q_i, d_{ik}) - \phi(q_i, d_{il}))], \tag{2}$$

where $z_{kl} = +1$ if $\pi_i(k) < \pi_i(l)$ ($d_{ik}$ is ranked ahead of $d_{il}$ in $\pi_i$), and $-1$ otherwise. (A slightly different definition on $\Phi$ is given in [30].) We define $F$ as a linear combination of the features in feature vector $\Phi$:

$$F(q_i, \mathbf{d}_i, \pi_i) = w^\top \Phi(q_i, \mathbf{d}_i, \pi_i), \tag{3}$$

where $w$ denotes the weight vector. In ranking, the permutation with the largest score given by $F$ is selected:

$$\sigma_i = \arg\max_{\sigma \in \Pi_i} F(q_i, \mathbf{d}_i, \sigma). \tag{4}$$

It can be shown that, the two types of ranking models are equivalent, if the parameter vectors $w$'s in the two models are identical.

THEOREM 1. *Given a fixed parameter vector $w$, the two ranking models $f$ and $F$ generate the same ranking result. That is, permutations $\tau_i$ and $\sigma_i$ are identical.*

**Table 1: Summary of notations.**

| Notations | Explanations |
|---|---|
| $q_i \in Q$ | Query |
| $\mathbf{d}_i = \{d_{i1}, d_{i2}, \cdots, d_{i,n(q_i)}\}$ | List of documents for $q_i$ |
| $d_{ij} \in \mathbf{D}$ | $j^{th}$ document in $\mathbf{d}_i$ |
| $\mathbf{y}_i = \{y_{i1}, y_{i2}, \cdots, y_{i,n(q_i)}\}$ | List of ranks for $q_i$ |
| $y_{ij} \in \{r_1, r_2, \cdots, r_\ell\}$ | Rank of $d_{ij}$ w.r.t. $q_i$ |
| $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$ | Training set |
| $\pi_i \in \Pi_i$ | Permutation for $q_i$ |
| $\pi_i^* \in \Pi_i^* \subseteq \Pi_i$ | Perfect permutation for $q_i$ |
| $\phi(q_i, d_{ij})$ | Feature vector w.r.t. $(q_i, d_{ij})$ |
| $\Phi(q_i, \mathbf{d}_i, \pi_i)$ | Feature vector w.r.t. $(q_i, \mathbf{d}_i, \pi_i)$ |
| $f$ and $F$ | Ranking models |
| $E(\pi_i, \mathbf{y}_i) \in [0, +1]$ | Evaluation of $\pi_i$ w.r.t. $\mathbf{y}_i$ for $q_i$ |

Proof of the theorem can be found in the Appendix. Theorem 1 implies that Equation (4) can be computed efficiently by sorting documents using Equation (1).

In IR, evaluation measures are used to evaluate the goodness of a ranking model, which are usually query-based. By query based, we mean that the measure is defined on a ranking list of documents with respect to the query. These include MAP, NDCG, MRR (Mean Reciprocal Rank), WTA (Winners Take ALL), and Precision@n [1, 14]. We utilize a general function $E(\pi_i, \mathbf{y}_i) \in [0, +1]$ to represent the evaluation measures. The first argument of $E$ is the permutation $\pi_i$ created using the ranking model. The second argument is the list of ranks $\mathbf{y}_i$ given as ground truth. $E$ measures the agreement between $\pi_i$ and $\mathbf{y}_i$. Most evaluation measures return real values in [0, +1]. We denote the perfect permutation as $\pi_i^*$. Note that there may be more than one perfect permutation for a query, and we use $\Pi_i^*$ to denote the set of all possible perfect permutations for query $q_i$. For $\pi_i^* \in \Pi_i^*$, we have $E(\pi_i^*, \mathbf{y}_i) = 1$.

Table 1 gives a summary of notations described above.

# 4. DIRECT OPTIMIZATION METHODS

In this section, we give a general framework for analyzing learning to rank algorithms that directly optimize evaluation measures.

Ideally, we would create a ranking model that maximize the accuracy in terms of an IR measure on training data, or equivalently, minimizes the loss function defined as follows:

$$R(F) = \sum_{i=1}^{m} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) = \sum_{i=1}^{m} (1 - E(\pi_i, \mathbf{y}_i)), \quad (5)$$

where $\pi_i$ is the permutation selected for query $q_i$ by ranking model $F$ (or $f$). We refer to the loss function $R(F)$ (or $R(f)$) as the 'basic loss function' and those methods which minimize the basic loss function as the 'direct optimization approach'.

The rest of this paper will focus on the first category of direct optimization methods (defined in Section 1) which minimize loss functions upper bounding the basic loss function. Practically, in order to leverage existing optimization technologies like Boosting and SVM, bound optimization has been widely used. We can consider two types of upper bounds. The first one is defined directly on the IR measures (type one bound). The second one is defined on the pairs between the perfect and imperfect permutations (type two bound). AdaRank and SVM$^{map}$ turn out to be algorithms that minimize one of the two upper bounds, respectively. PermuRank, which we propose in this paper, is an algorithm minimizes a type two bound.

## 4.1 Type One Bound

The basic loss function can be upper bounded directly by the exponential function, logistic function, which is widely used in machine learning. The logistic function is defined as

$$\sum_{i=1}^{m} \log_2 \left(1 + e^{-E(\pi_i, \mathbf{y}_i)}\right).$$

The exponential function is defined as

$$\sum_{i=1}^{m} \exp\{-E(\pi_i, \mathbf{y}_i)\}.$$

We can use the logistic function and exponential function as 'surrogate' loss functions in learning. Note that both functions are continuous, differentiable, and even convex w.r.t. $E$. The exponential loss function is tighter than the logistic loss function since $E \in [0, +1]$.

The AdaRank algorithm proposed in [27] actually minimizes the exponential loss function (type one bound), by taking a boosting approach. Motivated by the famous AdaBoost algorithm [11], AdaRank optimizes the exponential loss function through continuously re-weighting the distribution over the training queries and creating weak rankers. To do so, AdaRank repeats the process of re-weighting the training query, creating a weak ranker, and calculating a weight for the weak ranker, according to the performances (in terms of one IR measure) of the weak rankers on the training queries. Finally, AdaRank linearly combines the weak rankers as the final ranking model.

## 4.2 Type Two Bound

Here, we introduce a new loss function.

$$\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} \left( (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot [\![ (F(q_i, \mathbf{d}_i, \pi_i^*) \leq F(q_i, \mathbf{d}_i, \pi_i)) ]\!] \right), \quad (6)$$

where $[\![\cdot]\!]$ is one if the condition is satisfied, otherwise zero.

The loss function measures the loss when the worst prediction is made, specifically, the difference between the performance of the perfect permutation (it equals one) and the minimum performance of an incorrect permutation (it is less than one).

The following theorem holds with regard to the new loss function.

THEOREM 2. *The basic loss function in (5) is upper bounded by the new loss function in (6).*

Proof of Theorem 2 can be found in the Appendix.

The loss function (6) is still not continuous and differentiable because it contains the 0-1 function $[\![\cdot]\!]$, which is not continuous and differentiable. We can consider using continuous, differentiable, and even convex upper bounds on the loss function (6), which are also upper bounds on the basic loss function (5).

1) The 0-1 function $[\![\cdot]\!]$ in (6) can be replaced with its upper bounds, for example, hinge, exponential, and logistic functions, yielding

$$\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot e^{-\left(F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \pi_i)\right)};$$

$$\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot \log_2 \left(1 + e^{-\left(F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \pi_i)\right)}\right);$$

$$\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot [1 - (F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \pi_i))]_+;$$

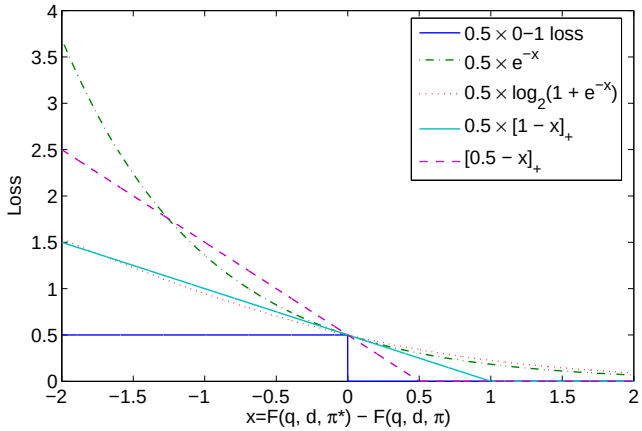


**Figure 1: Type two bounds.**

$$\sum_{i=1}^{m}\left[\max_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\setminus\Pi_i^*}((E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))-(F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i)))\right]_+,$$

where $[\cdot]_+$ denotes the hinge function.

Figure 1 shows the relationship between the loss function (6) and its upper bounds, where $E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i)$ is set to 0.5. From the figure, we can see that it is not possible to say which upper bound is the tightest. Different upper bounds may be suitable for different datasets.

2) The max function can also be replaced with its upper bound, the sum function. This is because $\sum_i x_i \geq \max_i x_i$ if $x_i \geq 0$ holds for all $i$.

3) Relaxations 1 and 2 can be applied simultaneously.

For example, replacing $[\![\cdot]\!]$ with the hinge function and max with sum, we obtain:

$$\sum_{i=1}^{m}\sum_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\setminus\Pi_i^*}(E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))\cdot[1-(F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i))]_+. \tag{7}$$

We can derive different algorithms by using the upper bounds as surrogate loss functions. SVM$^{map}$ and PermuRank are two examples.

SVM$^{map}$ solves the following quadratic programming problem:

$$\min_{\vec{w};\xi\geq 0}\frac{1}{2}\|\vec{w}\|^2+\frac{C}{m}\sum_{i=1}^{m}\xi_i \tag{8}$$

$$s.t.\quad \forall i,\forall\pi_i^*\in\Pi_i^*,\forall\pi_i\in\Pi_i\setminus\Pi_i^*:$$
$$F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i)\geq E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i)-\xi_i,$$

where $C$ is the coefficient for trade-off between total empirical loss and model complexity, and $\xi_i$ represents the empirical loss for $q_i$. One can easily verify that in the constraints the empirical loss $\xi_i$ is the maximum among all the losses of permutations for query $q_i$.

Equivalently, SVM$^{map}$ minimizes the following regularized hinge loss function

$$\sum_{i=1}^{m}\left[\max_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\setminus\Pi_i^*}((E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))\right.$$
$$\left.-(F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i)))\right]_+ +\lambda\parallel\vec{w}\parallel^2. \tag{9}$$

Intuitively, the first term calculates the total maximum empirical loss when selecting the best permutation for each of the queries. Specifically, if the difference between the permutations $F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i)$ is less than the difference between the corresponding evaluation measures $E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))$, then there will be a loss, otherwise not. Next, the maximum loss is selected for each query and they are summed up over all the queries.

Since $c\cdot[\![x\leq 0]\!]\leq[c-x]_+$ holds for all $c\in\Re^+$ and $x\in\Re$, it is easy to see that the upper bound in (9) also bounds the basic loss function in (5) (See also Figure 1). In [30], the authors have proved this fact (see also [26]).

## 4.3 PermuRank

In principle, any type two bound can be optimized using optimization techniques such as those in Perceptron, Support Vector Machines, and Boosting. However, the sizes of permutation sets $\Pi_i^*$ and $\Pi_i\setminus\Pi_i^*$ are both of order $O(n!)$, which makes the optimization infeasible. Here $n$ denotes the number of documents associated with query $q_i$.

In this paper, we propose a new direct optimization algorithm which efficiently minimizes one of the type two bounds as loss function in a greedy way. The algorithm is referred to as PermuRank and is shown in Figure 2. The key idea in PermuRank is to maintain a set of perfect permutations and a set of imperfect permutations as working sets, in stead of using the entire set of perfect permutations and the entire set of imperfect permutations.

PermuRank takes a training set $S=\{(q_i,\mathbf{d}_i,\mathbf{y}_i)\}_{i=1}^m$ as input and takes an evaluation measure $E$ and number of iterations $T$ as parameters. PermuRank runs $T$ rounds and at each round it creates a ranking model $F_t(t=1,\cdots,T)$. Finally, it outputs a ranking model $F$ created during the last round.

At each round $t$, PermuRank maintains a set of perfect permutations and a set of imperfect permutations for each query $q_i$, denoted as $\mathcal{B}_i^t$ and $C_i^t$, respectively. These two sets are initialized with an arbitrary perfect permutation $\pi_i^*\in\Pi_i^*$ and an arbitrary imperfect permutation $\pi_i\in\Pi_i\setminus\Pi_i^*$. At each round, the two sets are updated by adding the most violated perfect and imperfect permutations respectively:

$$\mathcal{B}_i^{t+1}\leftarrow\mathcal{B}_i^t\cup\{\arg\min_{\pi_i\in\Pi_i^*}F_t(q_i,\mathbf{d}_i,\pi_i)\},$$
$$C_i^{t+1}\leftarrow C_i^t\cup\{\arg\max_{\pi_i\in\Pi_i\setminus\Pi_i^*}F_t(q_i,\mathbf{d}_i,\pi_i)\}.$$

At each round $t$, a ranking model $F_t$ is created using the permutation sets $\mathcal{B}_i^t$ and $C_i^t$, $i=1,\cdots,m$ created so far

$$F_t=\arg\max_{F\in\mathcal{F}}L(\mathcal{B}_1^t,C_1^t,\cdots,\mathcal{B}_m^t,C_m^t), \tag{10}$$

where $L(\mathcal{B}_1^t,C_1^t,\cdots,\mathcal{B}_m^t,C_m^t)$ is a type two bound, based on $\mathcal{B}_i^t$ and $C_i^t$ instead of $\Pi_i^*$ and $\Pi_i\setminus\Pi_i^*$.

In this paper, without loss of generality, we use the hinge loss function of Equation (7). The total empirical loss $L$ becomes

$$L(\mathcal{B}_1,C_1,\cdots,\mathcal{B}_m,C_m)=\sum_{i=1}^{m}l(\mathcal{B}_i,C_i), \tag{11}$$

where

$$l(\mathcal{B}_i,C_i)=\frac{1}{|\mathcal{B}_i|}\sum_{\pi_i^*\in\mathcal{B}_i}\sum_{\pi_i\in C_i}(E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))$$
$$\cdot[1-(F(q_i,\mathbf{d}_i,\pi_i^*;w)-F(q_i,\mathbf{d}_i,\pi_i;w))]_+.$$

In this paper, we employ the SVM technique to minimize the regularized hinge loss function.

The learned ranking model $F_t$ is then used to update $\mathcal{B}_i^{t+1}$ and $C_i^{t+1}$ for training the next ranking model $F_{t+1}$.

At each round, PermuRank checks whether the permutation sets $\mathcal{B}_i^t$ and $C_i^t$ are changed. If there is no change, the algorithm will stop and return $F_t$ as the final ranking model.

Input: $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$, parameters $E$ and $T$
Initialize $\mathcal{B}_i^1$ and $C_i^1$, for all $i = 1, \cdots, m$.
**For** $t = 1, \cdots, T$

- $F_t = \arg\max_{F \in \mathcal{F}} L(\mathcal{B}_1^t, C_1^t, \cdots, \mathcal{B}_m^t, C_m^t)$.
- Update $\mathcal{B}_i^{t+1}$ and $C_i^{t+1}$, for all $i = 1, \cdots, m$.
- **break if** $\mathcal{B}_i^{t+1} = \mathcal{B}_i^t$ and $C_i^{t+1} = C_i^t$, for all $i = 1, \cdots, m$.

**End For**
**return** $F_t$.

**Figure 2: PermuRank algorithm.**

## 4.4 Summary on Bounds

We give a summary of the upper bounds on the basic loss function. Figure 3 shows the relationship. There is a basic loss function (5). On the left hand side is the type one bound. The upper bounds of the exponential loss function, logistic loss function can be used. On the right hand side is the type two bound. Equation (6) is the loss function for type two bound, which also upper bounds the basic loss function. Furthermore, the upper bounds of exponential loss function, logistic loss function, hinge loss functions, etc can be considered.

# 5. EXPERIMENTS

We conducted experiments to test the performances of the learning to rank methods of SVM$^{map}$, AdaRank, PermuRank, Ranking SVM, and RankBoost.

AdaRank and PermuRank can optimize any evaluation measure in $[0, +1]$. In our experiments, we chose MAP as the evaluation measure for them, denoted as AdaRank.MAP and PermuRank.MAP respectively. For AdaRank, we utilized features as weak rankers.

As measures for evaluation, we actually used MAP and NDCG at the positions of 1, 3, 5, and 10.

## 5.1 Experiment with Letor Data

In the first experiment, we used the Letor benchmark datasets [18]: OHSUMED, TD2003, and TD2004.

Letor OHSUMED dataset consists of articles from medical journals. There are 106 queries in the collection. For each query, there are a number of associated documents. The relevance degrees of documents with respect to the queries are given by humans, on three levels: definitely, possibly, or not relevant. There are 16,140 query-document pairs with relevance labels. In Letor, the data is represented as feature vectors and their corresponding relevance labels. Features in Letor OHSUMED dataset consists of 'low-level' features and 'high-level' features. Low-level features include term frequency (tf), inverse document frequency (idf), document length (dl), and their combinations. High-level features include BM25 and LMIR scores. In total, there are 25 features.

Letor TD2003 and TD2004 datasets are from the topic distillation task of TREC 2003 and TREC 2004. TD2003 has 50 queries and TD2004 has 75 queries. The document collection is a crawl of the .gov domain. For each query, there are about 1,000 associated documents. Each query document pair is given a binary judgment: relevant or irrelevant. The features of Letor TD2003 and TD2004 datasets include low-level features such as term frequency (tf), inverse document frequency (idf), and document length (dl), as well as high-level features such as BM25, LMIR, PageRank, and HITS. In total, there are 44 features.

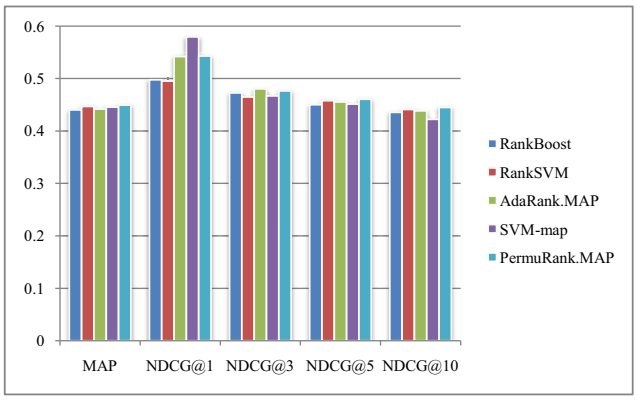


**Figure 4: Ranking accuracies on Letor OHSUMED data.**

We conducted 5-fold cross validation experiments, following the guideline of Letor. Figure 4 shows the results on Letor OHSUMED dataset in terms of MAP and NDCG, averaged over five trials. In calculation of MAP, we viewed 'definitely' and 'partially relevant' as relevant. (We tried treating 'partially relevant' as 'irrelevant', it did not work well for SVM$^{map}$). Figure 5 and Figure 6 show the results on the Letor TD2003 and TD2004 datasets.

We also conducted experiments to observe the training curve of PermuRank.MAP in terms of MAP on OHSUMED. We found that, in each fold of the cross validation, the training accuracy in terms of MAP would converge after 40 $\sim$ 100 iterations. Equivalently, the sizes of the working sets are also 40 $\sim$ 100, which is significantly smaller than $n!$, where $n$ denotes the number of documents associated with the query. Similar results were also observed in the experiments on TD2003 and TD2004.

On OHSUMED, the direct optimization methods of SVM$^{map}$, AdaRank, and PermuRank almost always outperform the baselines of Ranking SVM and RankBoost. We conducted t-tests on the improvements between the methods in terms of NDCG@1. The results show that on OHSUMED, the improvements of the direct optimization methods over the baselines are *statistically significant* (p-value $< 0.05$). The t-test results also show that no statistically significant difference exists among the performances of the direct optimization methods.

However, on TD2003 and TD2004 all the t-tests show that there is no statistically significant difference among the performances of all the methods. This is because the numbers of queries in TD2003 and TD2004 are too small, which is a common problem for the major publicly available datasets.

## 5.2 Experiment with WSJ and AP Data

In the second experiment, we made use of the WSJ and AP datasets used in [27].

The WSJ and AP datasets are from the TREC ad-hoc retrieval track. WSJ contains news articles by the Wall Street Journal, and AP contains 158,240 news articles of Associated Press. 200 queries are selected from the TREC topics (No.101 $\sim$ No.300). Each query has a number of documents associated and they are labeled as 'relevant' or 'irrelevant'. As features, we adopted those used in document retrieval [5] and [27]. They are *tf* (term frequency), *idf* (inverse document frequency), *dl* (document length), and BM25. Details of the datasets and features can be found in [5, 27].

WSJ and AP data were split into four even subsets and 4-fold cross-validation experiments were conducted. Figure 7 and Fig-

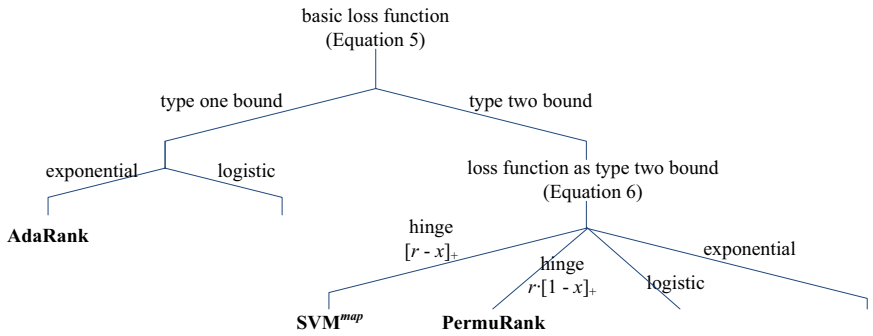


**Figure 3: Relation between upper bounds.**

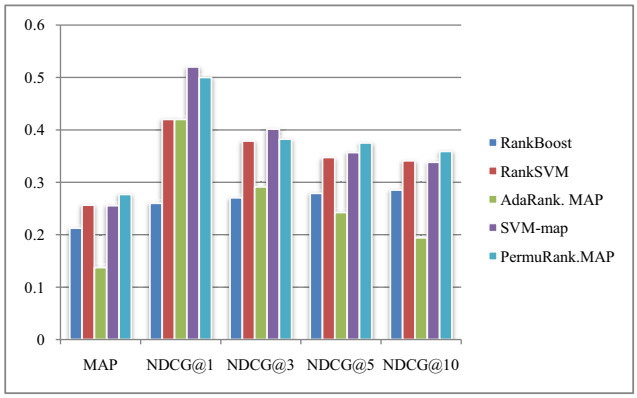


**Figure 5: Ranking accuracies on Letor TD2003 data.**

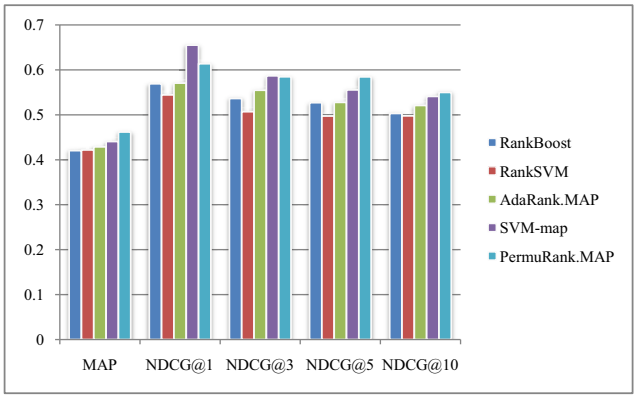


**Figure 7: Ranking accuracies on WSJ data.**

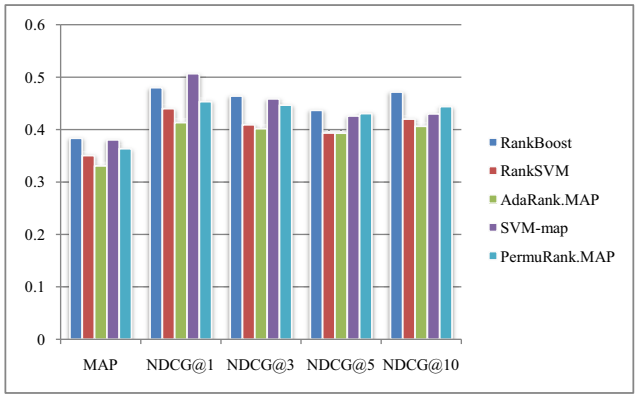


**Figure 6: Ranking accuracies on Letor TD2004 data.**

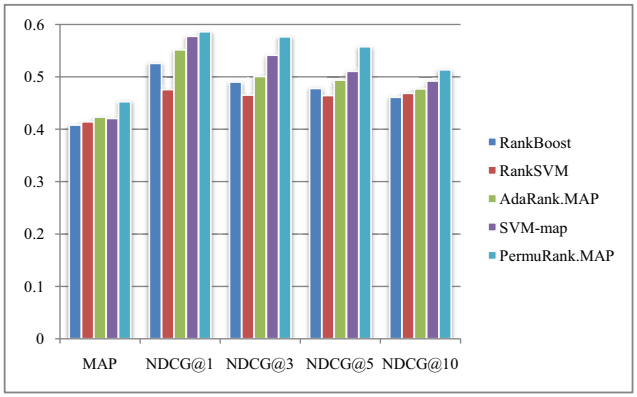


**Figure 8: Ranking accuracies on AP data.**

ure 8 respectively show the results in terms of MAP and NDCG, averaged over four trials.

The results show that the direct optimization methods of SVM$^{map}$ and PermuRank almost always outperform the baselines of Ranking SVM and RankBoost on WSJ and AP. With the help of t-test, we confirmed that the improvements of the SVM$^{map}$ and PermuRank over the baselines are *statistically significant* (p-value $<$ 0.05). Furthermore, SVM$^{map}$ and PermuRank work equally well, without statistically significant difference between their performances. In addition, this time AdaRank does not perform as well as expected: its performance is similar to the baselines, and significantly worse than SVM$^{map}$ and PermuRank.

## 5.3 Summary of Results

Table 2 and Table 3 show the ranking accuracies of the five methods on the datasets in terms of MAP and NDCG@3, respectively. Ranks of the five methods based on their performances on the datasets are also shown. The top ranked methods on the five datasets are highlighted. Note that the results on TD2003 and TD2004 are not statistically reliable; we list them here only for reference. From the results, we can conclude that the direct optimization methods of SVM$^{map}$, AdaRank, and PermuRank perform better than the baselines. Also, we conclude that these direct optimization methods themselves perform equally well.

**Table 2: Ranking accuracies in terms of MAP**

| | OHSUMED | WSJ | AP | TD2003 | TD2004 |
|---|---|---|---|---|---|
| SVM$^{map}$ | 0.4456(2) | 0.4406(2) | 0.4208(3) | 0.2554(3) | 0.3804(2) |
| AdaRank | 0.4419(4) | 0.4287(3) | 0.4233(2) | 0.1373(5) | 0.3308(5) |
| PermuRank | **0.4495(1)** | **0.4617(1)** | **0.4527(1)** | **0.2768(1)** | 0.3636(3) |
| RankSVM | 0.4469(3) | 0.4218(4) | 0.4144(4) | 0.2564(2) | 0.3505(4) |
| RankBoost | 0.4403(5) | 0.4203(5) | 0.4081(5) | 0.2125(4) | **0.3835(1)** |

**Table 3: Ranking accuracies in terms of NDCG@3**

| | OHSUMED | WSJ | AP | TD2003 | TD2004 |
|---|---|---|---|---|---|
| SVM$^{map}$ | 0.4669(4) | **0.5867(1)** | 0.5415(2) | **0.4014(1)** | 0.4586(2) |
| AdaRank | **0.4803(1)** | 0.5547(3) | 0.5010(3) | 0.2912(4) | 0.4017(5) |
| PermuRank | 0.4764(2) | 0.5846(2) | **0.5765(1)** | 0.3823(2) | 0.4467(3) |
| RankSVM | 0.4649(5) | 0.5069(5) | 0.4653(5) | 0.3787(3) | 0.4092(4) |
| RankBoost | 0.4726(3) | 0.5362(4) | 0.4902(4) | 0.2704(5) | **0.4640(1)** |

# 6. CONCLUSION AND FUTURE WORK

In this paper, we have studied the direct optimization approach to learning to rank, in which one trains a ranking model that can directly optimize the evaluation measures used in IR.

Previously several methods of direct optimization such as SVM$^{map}$ and AdaRank have been proposed. However, further theoretical and empirical investigations on this approach are still needed. In this paper we have tried to clarify many of the problems with regard to the direct optimization approach. We believe that this is extremely important for enhancing the state-of-the-art in learning to rank.

We conducted a theoretical analysis on the direct optimization approach. According to our study, the direct optimization approach is one that minimizes the basic loss function defined on the IR measures. It turns out that existing methods of AdaRank and SVM$^{map}$ try to minimize two types of upper bounds upon the basic loss function, respectively called type one bound and type two bound in the paper. With this analysis we are also able to derive a new direct optimization algorithm, called PermuRank.

We have also conducted empirical studies on AdaRank, SVM$^{map}$, and PermuRank using a number of benchmark datasets. Experimental results show that the direct optimization methods of SVM$^{map}$, AdaRank, and PermuRank can always perform better than the conventional methods of Ranking SVM and RankBoost. Furthermore, these direct optimization methods themselves can work equally well.

Future directions for this research include studying the tightness of different upper bounds upon the basic loss function and improving the framework so that more algorithms can be analyzed and compared.

# 7. ACKNOWLEDGMENTS

We thank Andrew Arnold, Lex Stein, Tao Qin, and Dwight Daniels for their valuable comments and suggestions to this paper.

# 8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[2] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[3] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 18*, pages 395–402. MIT Press, Cambridge, MA, 2006.

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

[5] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2006.

[6] 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*, pages 129–136, New York, NY, USA, 2007.

[7] D. Cossock and T. Zhang. Subset ranking using regression. In *Proceedings of the 19th Annual Conference on Learning Theory*, pages 605–619, 2006.

[8] H. M. de Almeida, M. A. Gonçalves, M. Cristo, and P. Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 399–406, New York, NY, USA, 2007.

[9] W. Fan, P. Pathak, and L. Wallace. Nonlinear ranking function representations in genetic programming-based ranking discovery for personalized search. *Decis. Support Syst.*, 42(3):1338–1349, 2006.

[10] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[11] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[12] G. Fung, R. Rosales, and B. Krishnapuram. Learning rankings via convex hull separation. In *Advances in Neural Information Processing Systems 18*, pages 395–402. MIT Press, Cambridge, MA, 2006.

[13] R. Herbrich, T. Graepel, and K. Obermayer. *Large Margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[14] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, New York, NY, USA, 2000.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002.

[16] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001.

[17] Q. Le and A. Smola. Direct optimization of ranking measures. Technical report, 2007.

[18] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007)*, 2007.

[19] D. Metzler. Direct maximization of rank-based metrics. Technical report, CIIR Technical Report, 2005.

[20] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998.

[21] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 279–286, New York, NY, USA, 2007.

[22] S. E. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *Proceedings of the 9th Text REtrieval Conference*, 2000.

[23] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimising non-smooth rank metrics. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007)*, 2007.

[24] A. Trotman. Learning to rank. *Inf. Retr.*, 8(3):359–381, 2005.

[25] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, New York, NY, USA, 2007.

[26] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

[27] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, USA, 2007.

[28] J.-Y. Yeh, J.-Y. Lin, H.-R. Ke, and W.-P. Yang. Learning to rank for information retrieval using genetic programming. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007)*, 2007.

[29] H. Yu. SVM selective sampling for ranking with application to data retrieval. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 354–363, New York, NY, USA, 2005.

[30] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, New York, NY, USA, 2007.

[31] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, Cambridge, MA, 2008.

# Appendix

Proof of Theorem 1.

Proof. Without loss of generality, assume that we have a query $q$ with $n$ associated documents $d_1, d_2, \cdots, d_n$.

With the use of model $f$, the relevance scores of the $n$ documents become $s_1 = f(q, d_1) = w^\top \phi(q, d_1)$, $s_2 = f(q, d_2) = w^\top \phi(q, d_2), \cdots, s_n = f(q, d_n) = w^\top \phi(q, d_n)$.

With the use of $F$ and the features defined in Equation (2), $F(q, \mathbf{d}, \pi)$ can be written as

$$\begin{aligned} F(q, \mathbf{d}, \pi) &= w^\top \frac{1}{n(q)\cdot(n(q)-1)} \sum_{k,l:k<l} [z_{kl}(\phi(q, d_k) - \phi(q, d_l))] \\ &= \frac{1}{n(q)\cdot(n(q)-1)} \sum_{k,l:k<l} [z_{kl}(w^\top \phi(q, d_k) - w^\top \phi(q, d_l))] \\ &= \frac{1}{n(q)\cdot(n(q)-1)} \sum_{k,l:k<l} [z_{kl}(s_k - s_l)], \end{aligned} \tag{12}$$

where $z_{kl} = +1$ if $\pi(k) < \pi(l)$, and $z_{kl} = -1$ otherwise. Since $\pi$ is only related to the variables $z_{kl}$'s in the equation, the equation is maximized w.r.t. $\pi$, if and only if all the terms in the summation are not negative.

Next, we prove that the permutation given by model $f$ is equivalent to the permutation given by model $F$, and vice versa.

**1.** $\tau$ is obtained by sorting documents in descending order with $f(q, d_i)(i = 1, \cdots, n)$. We have $\tau(k) < \tau(l) \Rightarrow s_k \geq s_l$, for $k, l = 1, \cdots, n$. According to the definition of $z_{kl}$, we have $z_{kl}(s_k - s_l) = |s_k - s_l| \geq 0$ for all $k, l = 1, \cdots, n$, given $\tau$. Since all the terms in the summation of Equation (12) are not negative, $F$ is maximized: $\tau = \arg\max_{\tau\in\Pi} F(q, \mathbf{d}, \tau) = \sigma$.

**2.** $\sigma$ is obtained by maximizing $F$: $\sigma = \arg\max_{\sigma\in\Pi} F(q, \mathbf{d}, \sigma)$. Based on the analysis above, we know the maximum is achieved when all of the terms in the summation are not negative: $z_{kl}(s_k - s_l) = |s_k - s_l|$ for all $k, l = 1, \cdots, n$. According to the definition of $z_{kl}$, for all $k, l = 1, \cdots, n$, we have: (a) $s_k > s_l \Rightarrow \sigma(k) < \sigma(l)$; and (b) $s_k = s_l \Rightarrow \sigma(k) < \sigma(l)$ or $\sigma(k) > \sigma(l)$. (a) and (b) mean $\sigma$ can also be obtained by ranking the documents according to their relevance scores, i.e., $\tau = \sigma$.

Summarizing 1 and 2, we conclude that with the same parameter vector $w$, the ranking models $f$ and $F$ generate the same ranking result.

□

Proof of Theorem 2.

Proof. Let

$$l(q_i) = \max_{\pi_i^*\in\Pi_i^*, \pi_i\in\Pi_i\setminus\Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot [\![(F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \pi_i)) \leq 0]\!],$$

and $r(q_i) = 1 - E(\sigma_i, \mathbf{y}_i)$, where $\sigma_i$ is the permutation selected for query $q_i$ by model $F$. There are two cases:

**Case 1** $\sigma_i \in \Pi_i^*$**:** If $\sigma_i \in \Pi_i^*$, $E(\sigma_i, \mathbf{y}_i) = E(\pi_i^*, \mathbf{y}_i) = 1$, it is obvious that $r(q_i) = 1 - E(\sigma_i, \mathbf{y}_i) = 0$ and $l(q_i) \geq 0$. Thus we have $l(q_i) \geq 0 = r(q_i)$.

**Case 2** $\sigma_i \notin \Pi_i^*$**:** Since $\sigma_i = \arg\max_{\sigma\in\Pi_i} F(q, \mathbf{d}, \sigma)$, we have $F(q, \mathbf{d}, \pi_i^*) - F(q, \mathbf{d}, \sigma_i) \leq 0$. Thus

$$\begin{aligned} l(q_i) &\geq \max_{\pi_i^*\in\Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\sigma_i, \mathbf{y}_i)) \cdot [\![(F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \sigma_i)) \leq 0]\!] \\ &= \max_{\pi_i^*\in\Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\sigma_i, \mathbf{y}_i)) \\ &= r(q_i). \end{aligned}$$

Summarizing case 1 and case 2, we obtain

$$\sum_{i=1}^{m} l(q_i) \geq \sum_{i=1}^{m} r(q_i).$$

□