Direct Optimization of Evaluation Measures in Learning to Rank

Jun Xu, Hang Li, Tie-Yan Liu, Yisha Peng[1], Min Lu[1], Wei-Ying Ma

Microsoft Research Asia, 4F Sigma Building, No. 49 Zhichun Road, Beijing, 100190, P. R. China

Abstract

One of the central issues in learning to rank for Information Retrieval (IR) 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). In this paper, we aim to conduct a comprehensive study on the approach of directly optimizing evaluation measures in learning to rank for 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, which is based on upper bound analysis and two types of upper bounds are discussed. Moreover, we make theoretical analysis the two types of upper bounds and show that we can derive new algorithms on the basis of this analysis and present two new algorithms called AdaRank and PermuRank. We make comparisons between direct optimization methods of AdaRank, PermuRank, and SVM^{map}, using benchmark datasets. We also compare them with conventional methods of Ranking SVM and RankBoost. Experimental results show that the methods based on direct optimization of ranking measures can always outperform these conventional methods. However, no significant difference exists among the performances of the direct optimization methods themselves.

Key words: Information retrieval, Learning to rank, Evaluation measures

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 consist 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) [16]. Ideally, a learning algorithm would train 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 to minimize a loss function loosely related to the IR measures. For example, Ranking SVM [15] and RankBoost [12] minimize loss functions based on errors in the ordering of document pairs. Recently, researchers have developed several new algorithms that manage to directly optimize the performance in terms of

[1]The work was conducted when Yisha Peng and Min Lu were visiting Microsoft Research Asia

IR measures. The effectiveness of these methods have also been verified. For example, [25] has given a theoretical justification to the approach of directly optimizing IR measures.

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 [33, 19]. Second, one can approximate the IR measures with functions that are easy to handle [9, 27, 25]. Third, one can use specially designed techniques for optimizing the non-smooth IR measures [3, 10].

There are still 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 of existing methods? (3) Which direct optimization method empirically performs best?

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

(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. Relations between the two types of bounds are also analyzed. With this analysis, different algorithms can be studied and compared. Moreover, new algorithms can be easily derived. As examples, we propose new algorithms of AdaRank and PermuRank.

(2) We show that the existing algorithms SVM^{map} and PermuRank manage to minimize type two bound. In contrast, AdaRank tries to minimize type one bound.

(3) We compare the performances of the direct optimization methods using 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.

This paper is an extension of our previous papers of [30] and [31]. Contributions of the paper include the following points. 1) Theoretical analysis on the relationship between type one bound and type two bound is provided; 2) simulation experiments have been conducted to further verify the theoretical results; 3) experiments have been newly performed, so that all of the experiments are based on publicly available benchmark data.

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 new algorithms (AdaRank and PermuRank) and an existing algorithm SVM^{map} are analyzed and discussed respectively in Section 5. Section 6 reports our experimental results with discussions in Section 7, and Section 8 concludes this paper.

2. Related Work

The key problem for document retrieval is ranking, specifically, to create a ranking model (function) that can sort documents based on their relevance to the given query. It is a common practice in IR to tune the parameters of a ranking model using some labeled data and an evaluation measure [1]. For example, the state-of-the-art methods of BM25 [26] and LMIR (Language

Models for Information Retrieval) [18, 24] all have parameters to tune. As the ranking models become more sophisticated (with more features) and more labeled data becomes 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. For example, Joachims [17] applies Ranking SVM to document retrieval. He utilizes click-through data to deduce training data for the model creation. Cao et al [5] adapt Ranking SVM to document retrieval by modifying the Hinge Loss function to better meet the requirements of IR. Specifically, they introduce a Hinge Loss function that heavily penalizes errors on the top of ranking lists and errors from queries with fewer retrieved documents.

In our view, there are two major approaches to learning to rank for IR: the pairwise approach and listwise approach. The pairwise approach transforms the ranking problem into binary classification on document pairs. Typical methods include Ranking SVM [15, 17], RankBoost [12], and RankNet [4]. The pairwise approach minimizes loss functions that are loosely related to the evaluation measures, such as MAP and NDCG. The listwise approach considers the retrieved document list as a unit for learning. One can either define a probability model on the document list, or perform direct optimization of IR measures. ListNet [6] and ListMLE [29] are typical methods for the formmer, and the latter is the major interest of this work.

There are three ways for directly optimizating IR measures. First, one can minimize loss functions upper bounding a loss function defined upon an IR measure. For example, SVM^{map} [33] minimizes a hinge loss function, which upper bounds a loss function based on Average Precision. (See also [19, 7, 8].)

Second, one can approximate the IR measures with easy-to-handle functions. For example, in [27] a smoothed approximation to NDCG [16] is proposed. The work in [25] addresses the task by approximating the IR measures (MAP [1] and NDCG) and optimizing the approximated surrogate functions. (See also [9, 20].)

Third, one can use specially designed techniques for optimizing non-smooth IR measures. For example, LambdaRank [3] implicitly minimizes a loss function related to IR measures. Genetic Programming (GP) is also employed to optimize IR measures [2]. For example, [10] proposes a specifically designed GP to learn a ranking model for IR. (See also [32, 11, 23]).

In this paper, we focus on the first category. For other methods of learning to rank, please refer to the comprehensive survey on learning to rank for IR [21].

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 ℓ 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 π_i on $\mathbf{d}_i$ as a bijection from $\{1, 2, \cdots, n(q_i)\}$ to itself. We use Π_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 $\tau(q_i, \mathbf{d}_i, f)$, or τ_i for short.

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 π_i, denoted as $\Phi(q_i, \mathbf{d}_i, \pi_i)$. We calculate Φ by linearly combining the feature vectors ϕ 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 π_i), and -1 otherwise. We define F as a linear combination of the features in feature vector Φ:

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

where w denotes the weight vector. In ranking, the permutation $\sigma(q_i, \mathbf{d}_i, F)$ (also denoted as σ_i for short) 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 τ_i and σ_i are identical.*

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

A slightly different definition of Φ is given in [33]:

$$\Phi(q_i, \mathbf{d}_i, \pi_i) = \frac{1}{|C^{q_i}| \cdot |C^{\overline{q_i}}|} \sum_{k:d_{ik} \in C^{q_i}} \sum_{l:d_{il} \in C^{\overline{q_i}}} [z_{kl}(\phi(q_i, d_{ik}) - \phi(q_i, d_{il}))],$$

where C^{q_i} and $C^{\overline{q_i}}$ denote the set of relevant and irrelevant documents for query q_i, respectively.

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 π_i w.r.t. $\mathbf{y}_i$ for q_i

It is assumed that the data has only two relevance ranks. Intuitively, Φ is a summation over the vector differences of all relevant/irrelevant document pairings. With this definition of Φ, Equation (4) can also be computed efficiently, still by sorting documents using Equation (1). This can be obtained by following the proof of Theorem 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, 16]. 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 π_i created using the ranking model. The second argument is the list of ranks $\mathbf{y}_i$ given as the ground truth. E measures the agreement between π_i and $\mathbf{y}_i$. Most evaluation measures return real values in [0, +1]. We denote the perfect permutation as π_i^*. Note that there may be more than one perfect permutation for a query, and we use Π_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 the notations described above.

4. Direct Optimization Framework

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 maximizes the accuracy in terms of an IR measure on training data, or equivalently, minimizes the loss function defined as below:

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

where π_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'.

This paper focuses on the bounding approach. The advantage of taking this approach is that one can leverage existing learning techniques, such as Boosting and SVM. We consider two

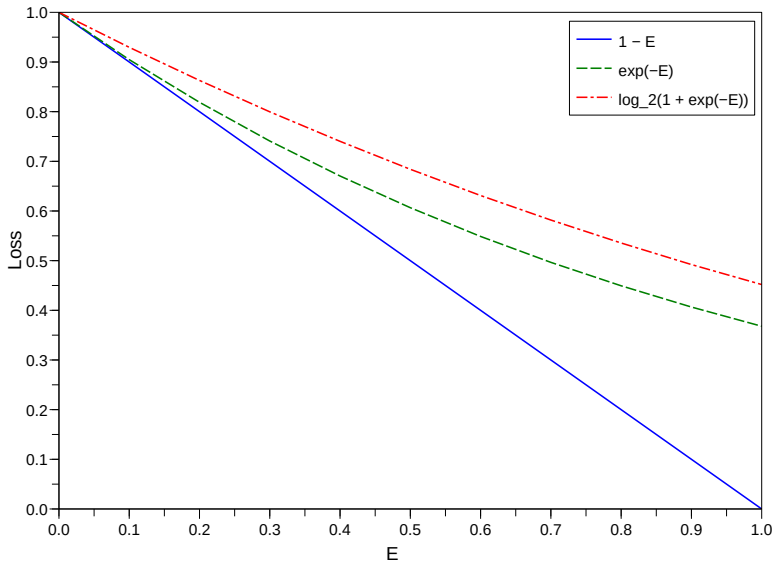

Figure 1: Type one bounds.

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). SVM^{map} turns out to be an algorithm that minimizes a type two bound. AdaRank and PermuRank, which we propose in this paper, are two algorithms that minimize a type one bound and a type two bound, respectively.

4.1. Type One Bound

The basic loss function defined in (5) can be upper bounded directly by the exponential function, logistic function, which is widely used in machine learning. The logistic function and exponential function are defined as

- logistic loss:

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

- exponential loss:

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

Here the π_i is the permutation selected for query q_i and document set $\mathbf{d}_i$, by ranking model f. Note that both functions are continuous, differentiable, and even convex with respect to E. The exponential loss function is tighter than the logistic loss function since $E \in [0, +1]$ (c.f. Fig. 1).

4.2. Type Two Bound

Here, we introduce a new loss function.

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

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 to one) and the minimum performance of an incorrect permutation (it is less than one).

The following theorem holds with respect 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 Appendix B.

The loss function (6) is still not continuous or differentiable because it contains the 0-1 function $[\![\cdot]\!]$, which is neither continuous nor 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, logistic, exponential, and hinge functions, yielding

- Logistic loss:

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

- Exponential loss:

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

- Hinge loss:

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

or

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

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

Fig. 2 shows the relation 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.

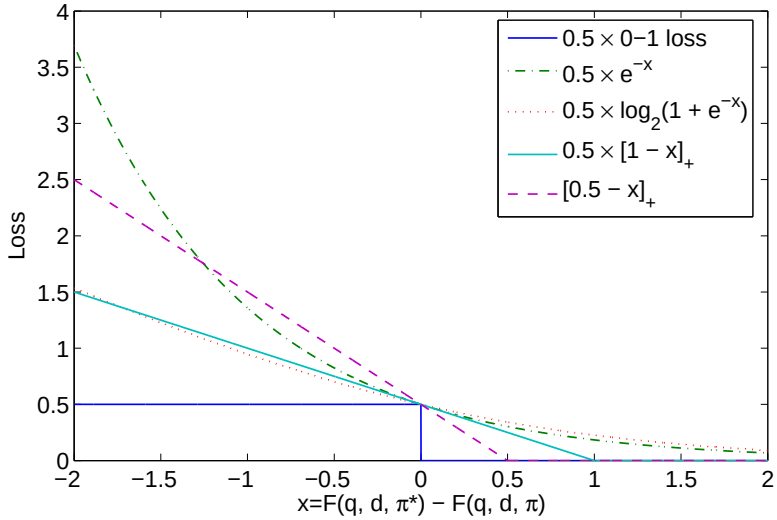

Figure 2: Type two bounds.

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.

4.3. Relation between Type One Bound and Type Two Bound

Relation between the type one bound and type two bound can be summarized as the following Theorem 3:

Theorem 3. *For any monotonically nondecreasing and positive function $\varphi(\cdot)$, the inequality*

$$\sum_{i=1}^{m} \max_{\pi_i^* \in \prod_i^*; \pi_i \in \prod_i \setminus \prod_i^*} (1 - E(\pi_i, y_i)) \cdot \varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right) \leq \sum_{i=1}^{m} \varphi\left(-E(\sigma_i, y_i)\right), \tag{8}$$

if $\forall i = 1, ..., m$,

$$\min_{\pi_i^* \in \prod_i^*; \pi_i \in \prod_i \setminus \prod_i^*} (F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)) \geq E(\sigma_i, y_i) \tag{9}$$

holds, where $\sigma_i = \arg\max_{\sigma \in \prod_i} F(q_i, \mathbf{d}_i, \sigma)$ *is a permutation of integers created for query q_i, corresponding list of documents $\mathbf{d}_i$, and ranking function F.*

Proof of Theorem 3 can be found in Appendix C.

The left side of Equation (8) corresponds to the type two bound and the right side corresponds to the type one bound. φ can be any monotonically nondecreasing positive function such as e^x,

$\log_2(1+e^x)$, and $[1+x]_+$, which corresponds to the exponential loss, logistic loss, and hinge loss, respectively. Theorem 3 indicates that, for a given function φ, the type two bound is tighter than the corresponding type one bound, if condition (9) holds. Otherwise, the relation between the type one bound and type two bound is not clear. One can easily verify that condition (9) means that the trained ranking model is 'close' to the optimal one.

We conduct a simulation to verify the correctness of the result. First, we assume that there are two relevance levels: 'relevant' and 'irrelevant' and instances in the two dimensional Euclidean space are generated according to Gaussian distributions $N(\mathbf{m}_k, \mathbf{\Sigma})$. We set the centers as $\mathbf{m}_1 = (+1, +1)$ and $\mathbf{m}_2 = (-1, -1)$ for 'relevant' and 'irrelevant', respectively. The deviation matrix $\mathbf{\Sigma} = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}$.

Next, according to the distribution, we randomly generate n_1 and n_2 instances for rank levels of 'relevant' and 'irrelevant', respectively (synthetic dataset 1). We set $n_1 = 7$ and $n_2 = 14$. (n_1 and n_2 are relatively small because precisely calculating the type two loss is time consuming for large datasets.)

We assume the ranking model is linear and represented as a 2-dimension vector $\mathbf{w} = (\cos(\theta), \sin(\theta))$. Since only the direction of the ranking model impacts the order of instances, we change the free parameter θ from 0 to 2π to obtain a list of ranking models. Values of the exponential type one bound and exponential type two bound with respect to θ are illustrated in Fig. 3.

We also generate another synthetic dataset (synthetic dataset 2) with deviation matrix $\mathbf{\Sigma} = \begin{pmatrix} 0.2 & 0 \\ 0 & 1.3 \end{pmatrix}$. All other parameters are identical to that for dataset 1. In synthetic dataset 2, some of the 'relevant' and 'irrelevant' instances are mixed, which reflects the characteristics of the real data. Values of the exponential type one bound and type two bound with respect to θ are shown in Fig. 4.

From these two figures, we can see that both the type one bound and type two bound are optimized when $\theta = \frac{\pi}{4}$. The type two bound is tighter than the type one bound when the ranking model is close to its optimal. The results agree well with Theorem 3.

4.4. Summary on Bounds

Fig. 5 shows the relationship between the bounds. There is a basic loss function (5). On the left hand side is the type one bound. It includes the exponential loss function and logistic loss function as examples. On the right hand side is the type two bound (i.e., Equation (6)). It contains the exponential loss function, logistic loss function, and hinge loss functions.

With this framework, we can: 1) derive new algorithms based on the framework; 2) analyze existing algorithms within the framework (we will analyze the existing direct optimization algorithm of SVM^{map} in detail in Section 5.3)

5. Direct Optimization Algorithms

We can derive different algorithms by using the upper bounds as surrogate loss functions. In principle, any type one and type two bounds can be optimized using optimization techniques such as those in Perceptron, Support Vector Machines, and Boosting. As examples, we will show that the new algorithms of AdaRank and PermuRank can be derived as minimizing the type one bound and type two bound, respectively. We will also show that the existing algorithm of SVM^{map} can be derived as one optimizing the type two bound.

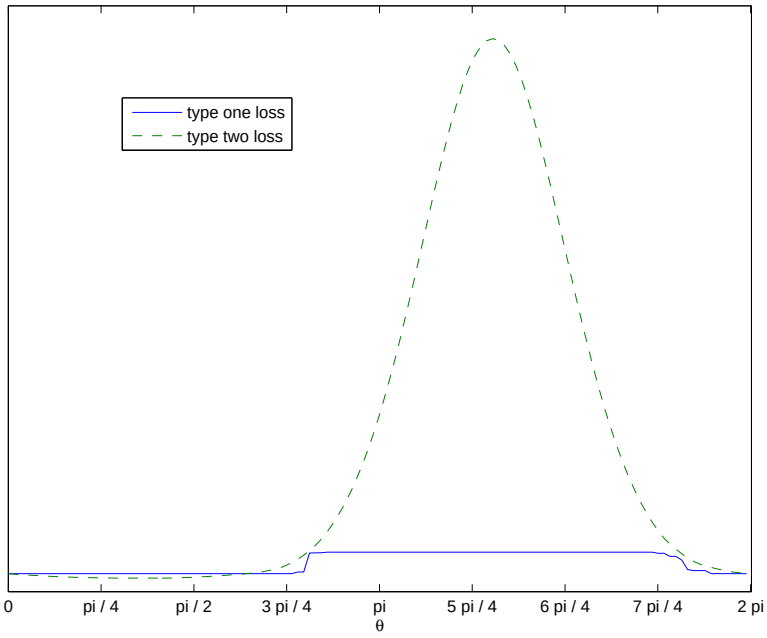

Figure 3: Simulation experiment 1.

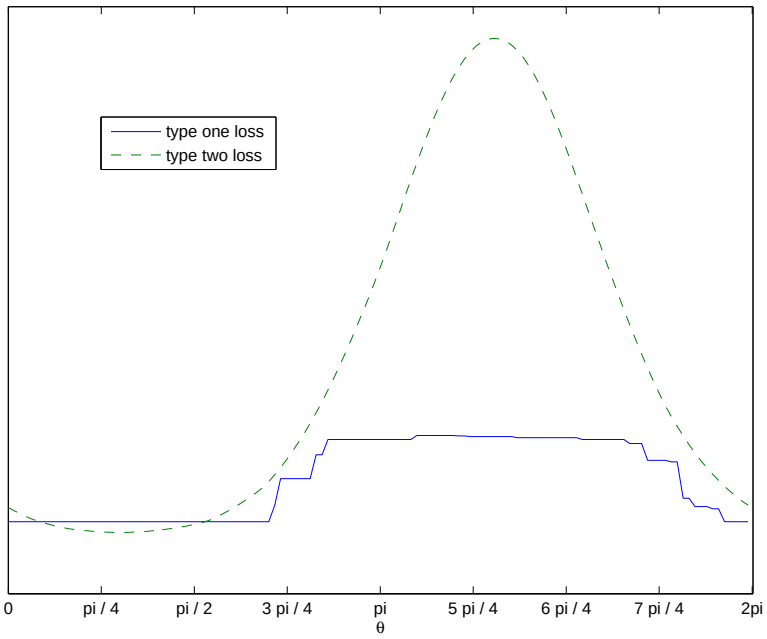

Figure 4: Simulation experiment 2.

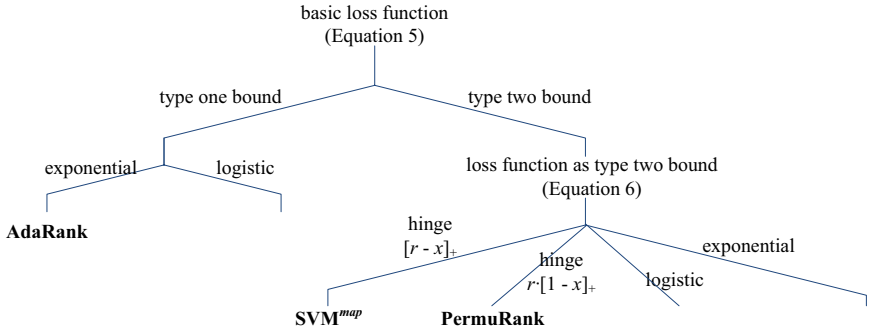

Figure 5: Relation between upper bounds.

5.1. AdaRank

Here, we propose a new direct optimization algorithm which efficiently minimizes one type one bound (exponential type one bound) as a loss function, using the boosting technique [13]. The algorithm is referred to as AdaRank and is shown in Fig. 6.

AdaRank takes a training set $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$ as input and takes the performance measure function E and the number of iterations T as parameters. AdaRank runs T rounds and at each round it creates a weak ranker $h_t (t = 1, \cdots, T)$. Finally, it outputs a ranking model f by linearly combining the weak rankers.

At each round, AdaRank maintains a distribution of weights over the queries in the training data. We denote the distribution of weights at round t as $\boldsymbol{P}_t$ and the weight on the i^{th} training query q_i at round t as $P_t(i)$. Initially, AdaRank sets equal weights to the queries. At each round, it increases the weights of those queries that are not ranked well by f_t, the model created so far. As a result, the learning at the next round will be focused on the creation of a weak ranker that can work on the ranking of those 'hard' queries.

At each round, a weak ranker h_t is constructed based on training data with weight distribution $\boldsymbol{P}_t$. The goodness of a weak ranker is measured by the performance measure E weighted by $\boldsymbol{P}_t$:

$$\sum_{i=1}^{m} P_t(i) E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i),$$

where $\pi(q_i, \mathbf{d}_i, h_t)$ is the permutation created for query q_i, the corresponding list of documents $\mathbf{d}_i$, and the ranking function h_t.

Several methods for weak ranker construction can be considered. For example, a weak ranker can be created by using a subset of queries (together with their document list and label list) sampled according to the distribution $\boldsymbol{P}_t$. In this paper, we use single features as weak rankers, that is, as a weak ranker we choose the feature that has the optimal weighted performance among all of the features:

$$\max_k \sum_{i=1}^{m} P_t(i) \cdot E(\pi(q_i, \mathbf{d}_i, x_k), \mathbf{y}_i).$$

Creating weak rankers in this way, the learning process turns out to be that of repeatedly selecting features and linearly combining the selected features.

Once a weak ranker h_t is built, AdaRank chooses a weight $\alpha_t > 0$ for the weak ranker. Intuitively, α_t measures the importance of h_t.

Input: $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$, and parameters E and T
Initialize $P_1(i) = 1/m$.
For $t = 1, \cdots, T$

- Create weak ranker h_t with weighted distribution $\boldsymbol{P}_t$ on training data S.
- Choose α_t

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m P_t(i)\{1 + E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}{\sum_{i=1}^m P_t(i)\{1 - E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}.$$

- Create f_t

$$f_t(\vec{x}) = \sum_{k=1}^{t} \alpha_k h_k(\vec{x}).$$

- Update $\boldsymbol{P}_{t+1}$

$$P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\}}{\sum_{j=1}^m \exp\{-E(\pi(q_j, \mathbf{d}_j, f_t), \mathbf{y}_j)\}}.$$

End For
Output ranking model: $f(\vec{x}) = f_T(\vec{x})$.

Figure 6: The AdaRank algorithm.

A ranking model f_t is created at each round by linearly combining the weak rankers constructed so far $h_1, \cdots, h_t$ with weights $\alpha_1, \cdots, \alpha_t$. f_t is then used for updating the distribution $\boldsymbol{P}_{t+1}$.

The loss function in AdaRank is defined on the basis of general IR performance measures. The measures can be MAP, NDCG, WTA, MRR, or any other measures whose range is within $[0, +1]$. We next explain why this is the case.

We attempt to minimize the exponential type one bound in Section 4.1:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)\}.$$

We consider the use of a linear combination of weak rankers as our ranking model:

$$f(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x}).$$

The problem then turns out to be

$$\min_{h_t \in \mathcal{H}, \alpha_t \in \Re^+} L(h_t, \alpha_t) = \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{t-1} + \alpha_t h_t), \mathbf{y}_i)\},$$

where $\mathcal{H}$ is the set of possible weak rankers, α_t is a positive weight, and $(f_{t-1} + \alpha_t h_t)(\vec{x}) = f_{t-1}(\vec{x}) + \alpha_t h_t(\vec{x})$. Several ways of computing coefficients α_t and weak rankers h_t may be considered. Following the idea of AdaBoost, in AdaRank we take the approach of 'forward stage-wise additive modeling' [14] and get the algorithm in Fig. 6.

It can be proved that there exists a lower bound on the ranking accuracy for AdaRank on training data, as presented in Theorem 4.

Theorem 4. *The following bound holds on the ranking accuracy of the AdaRank algorithm on training data:*

$$\frac{1}{m}\sum_{i=1}^{m} E(\pi(q_i, \mathbf{d}_i, f_T), \mathbf{y}_i) \geq 1 - \prod_{t=1}^{T} e^{-\delta^t_{min}} \sqrt{1-\varphi(t)^2},$$

where $\varphi(t) = \sum_{i=1}^{m} P_t(i) E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)$, $\delta^t_{min} = \min_{i=1,\cdots,m} \delta^t_i$, *and*

$$\delta^t_i = E(\pi(q_i, \mathbf{d}_i, f_{t-1} + \alpha_t h_t), \mathbf{y}_i) - E(\pi(q_i, \mathbf{d}_i, f_{t-1}), \mathbf{y}_i) - \alpha_t E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i),$$

for all $i = 1, 2, \cdots, m$ *and* $t = 1, 2, \cdots, T$.

A proof of the theorem can be found in Appendix D. The theorem implies that the ranking accuracy in terms of the performance measure can be continuously improved, as long as $e^{-\delta^t_{min}} \sqrt{1-\varphi(t)^2} < 1$ holds.

Theorem 4, however, cannot guarantee that the total performance will definitely be improved, because it is likely that $e^{-\delta^t_{min}} \sqrt{1-\varphi(t)^2}$ is larger than 1, although in real world applications we usually observe that this is not the case. Fortunately, we can prove that when the evaluation measure is a dot product, the performance of AdaRank with respect to training data is guaranteed to improve when the number of iterations increases. It is easy to verify δ^i_t is always zero in such a case.

5.2. PermuRank

New direct optimization algorithms can also be derived through optimizing different type two bounds. The challenge here is that the sizes of permutation sets Π^*_i and $\Pi_i \setminus \Pi^*_i$ are both of order $O(n!)$, which makes the optimization intractable. Here n denotes the number of documents associated with query q_i.

PermuRank is an algorithm that can efficiently minimize one of the type two bounds as loss function in a greedy way. The algorithm is shown in Fig. 7. The key idea in PermuRank is to maintain a set of perfect permutations and a set of imperfect permutations as working sets, instead of using the entire set of perfect permutations and the entire set of imperfect permutations.

Similarly to AdaRank, 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 $\mathcal{B}^t_i$ and C^t_i, 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:

$$\mathcal{B}^{t+1}_i \leftarrow \mathcal{B}^t_i \cup \{\arg\min_{\pi_i \in \Pi^*_i} F_t(q_i, \mathbf{d}_i, \pi_i)\},$$

$$C^{t+1}_i \leftarrow C^t_i \cup \{\arg\max_{\pi_i \in \Pi_i \setminus \Pi^*_i} F_t(q_i, \mathbf{d}_i, \pi_i)\}.$$

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 $t = 1, \cdots, T$

- $F_t = \arg\min_{F \in \mathcal{F}} L(\mathcal{B}_1^t, C_1^t, \cdots, \mathcal{B}_m^t, C_m^t)$.
- 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 7: The PermuRank algorithm.

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

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

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 Π_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

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

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.

5.3. SVM^{map}

Existing direct optimization algorithms can be analyzed under the framework. Here we make use of SVM^{map} [33] as an example. SVM^{map} is a Support Vector Machine (SVM) algorithm for predicting rankings. The goal of SVM^{map} is to directly optimize ranking measures in terms of Mean Average Precision (MAP). During training, SVM^{map} solves the following quadratic programming problem:

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

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

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

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

$$\sum_{i=1}^{m}\left[\max_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\setminus\Pi_i^*}((E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i))-(F(q_i,\mathbf{d}_i,\pi_i^*)-F(q_i,\mathbf{d}_i,\pi_i)))\right]_+ +\lambda\parallel\vec{w}\parallel^2. \quad (10)$$

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 $c\cdot[\![x\le 0]\!]\le[c-x]_+$ holds for all $c\in\mathfrak{R}^+$ and $x\in\mathfrak{R}$, it is easy to see that the upper bound in (10) also bounds the basic loss function in (5) (See also Fig. 2). In [33], the authors have proved this fact (see also [28]).

6. Experiments

We conducted experiments to test the performances of the learning to rank methods of SVM^{map}, AdaRank, PermuRank, Ranking SVM, and RankBoost, on LETOR 2.0 benchmark datasets and benchmark datasets of WSJ and AP. We choose these datasets because they are all public available benchmark datasets and easy to reproduce the experiments.

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.MAP, we utilized features as weak rankers, as described in Section 5.1. As measures for evaluation, we actually used MAP and NDCG at the positions of 1, 3, 5, and 10.

6.1. Experiment with LETOR Data

In the first experiment, we used the LETOR 2.0 benchmark datasets of OHSUMED, TD2003, and TD2004[22].

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, We index the fields of .T (title) and .W (abstract) for documents and the field .W for queries. For both documents and queries, the field .I is used as id. 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.

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 'title' field is considered as the query strings. 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. Details of the datasets please refers to [22].

Following the data partition [22], we conducted 5-fold cross validation experiments. Fig. 8 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}). Fig. 9 and Fig. 10 show the results on the LETOR TD2003 and TD2004 datasets.

We also conducted experiments to observe the training curve of AdaRank.MAP and 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. Fig. 11) shows the learning curves during the training phase in one trial of the cross validation. From the figure, we can see that the ranking accuracy of AdaRank.MAP steadily improves, as the training goes on. The result agrees well with Theorem 4. That is, the ranking accuracy in terms of the performance measure can be continuously improved, as long as $e^{-\delta^t_{min}}\sqrt{1-\varphi(t)^2} < 1$ holds. For PermuRank, 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.

6.2. Experiment with WSJ and AP Data

In the second experiment, we made use of the WSJ and AP datasets.

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 by the 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]. 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].

The WSJ and AP data were split into four even subsets and 4-fold cross-validation experiments were conducted. Fig. 12 and Fig. 13 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

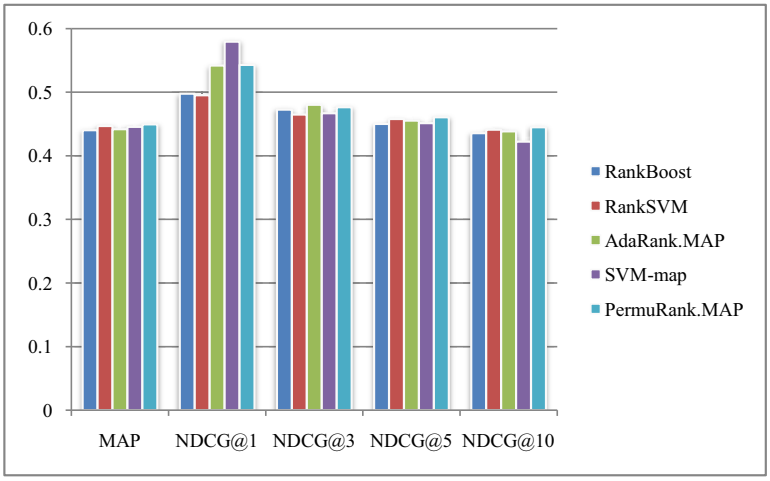

Figure 8: Ranking accuracies on LETOR OHSUMED data.

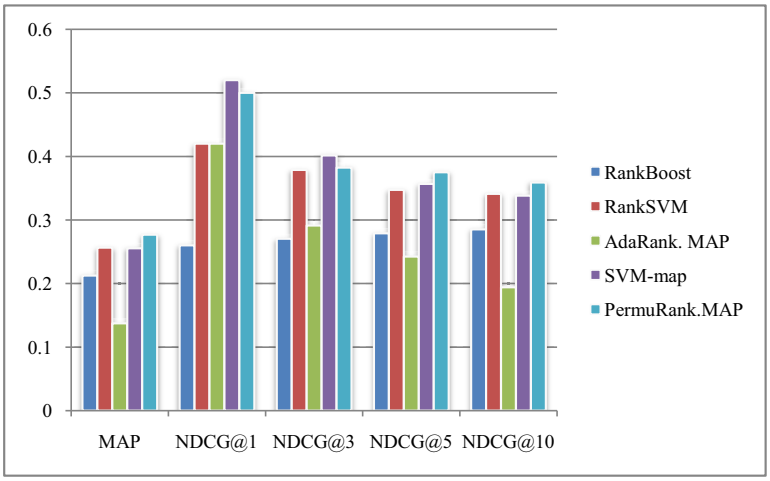

Figure 9: Ranking accuracies on Letor TD2003 data.

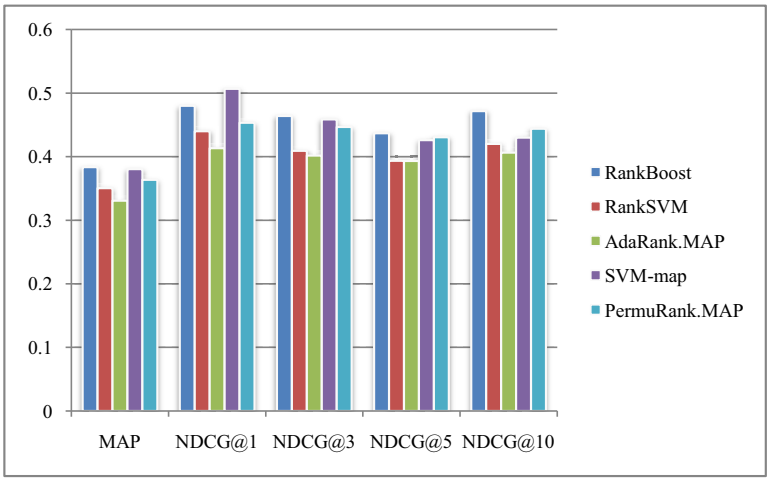

Figure 10: Ranking accuracies on Letor TD2004 data.

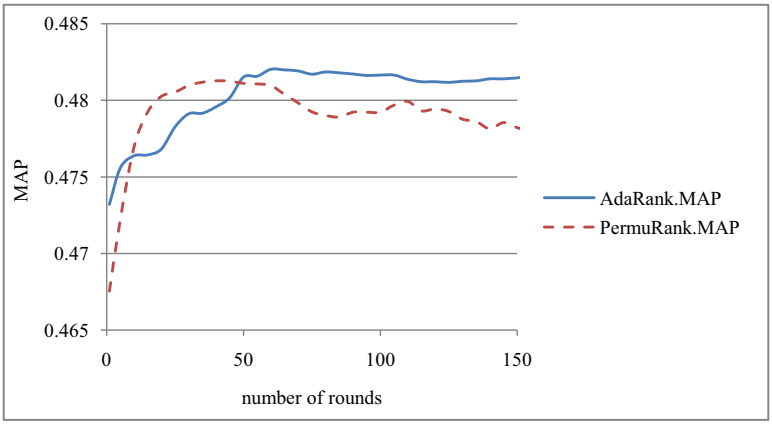

Figure 11: Learning curves of AdaRank and PermuRank.

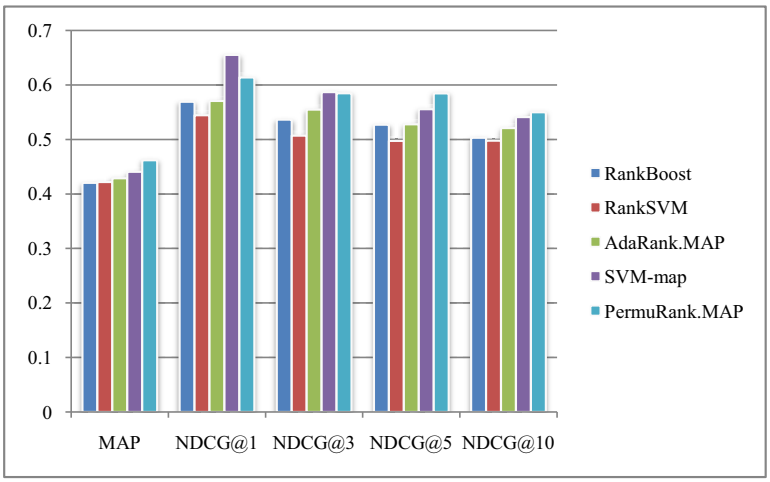

Figure 12: Ranking accuracies on WSJ data.

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)**

baselines are *statistically significant* (p-value < 0.05). Furthermore, SVM^{map} and PermuRank work equally well, without a 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.

6.3. 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 three 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 4 compares the direct optimization algorithms of AdaRank, PermuRank, and SVM^{map}. We can see that all of the three algorithms perform well empirically. However, AdaRank has several advantages over PermuRank and SVM^{map}.

First, AdaRank can incorporate any performance measure, provided that the measure is query based and in the range of $[0, +1]$. Notice that the major IR measures meet this requirement. In contrast the SVM^{map} only minimizes the IR measure of MAP.

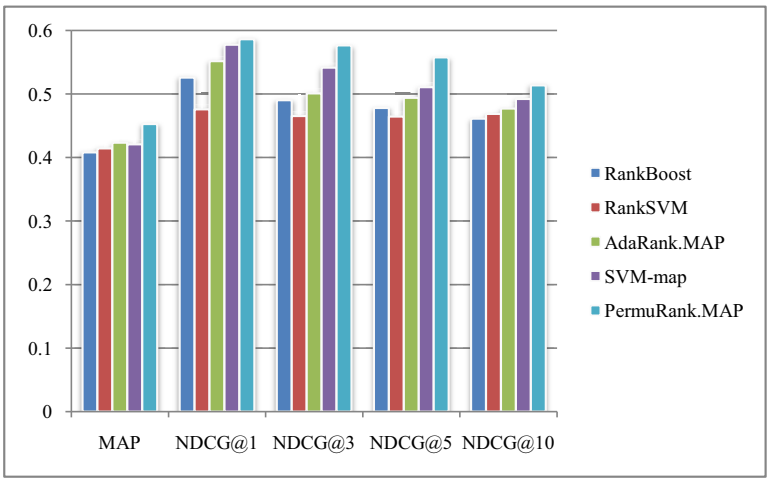

Figure 13: Ranking accuracies on AP data.

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)**

Table 4: Comparison of the direct optimization algorithms.

	AdaRank	PermuRank	SVM^{map}
Measure	measures in [0, +1]	measures in [0, +1]	MAP
Time complexity	low	high	high
Implementation	easy	difficult	difficult
Empirical performance	good	good	good

Second, the learning process of AdaRank is more efficient than PermuRank and SVM^{map}. The time complexity of AdaRank is of order $O((k+T)\cdot m\cdot n\log n)$, where k denotes the number of features, T the number of rounds, m the number of queries in training data, and n is the maximum number of documents for queries in training data. The time complexity of SVM^{map} and PermuRank is much higher since they try to optimize with the loss functions defined on permutations, which is of the order $n!$.

Third, AdaRank is easier to implement. AdaRank follows the idea of AdaBoost and takes the approach of 'forward stage-wise additive modeling' [14]. SVM^{map} and PermuRank, however, employ the technique of quadratic programming and are more difficult to be implemented.

7. Discussions

We investigated the reasons that direct optimization algorithms outperform the baseline methods, using the results of AdaRank on the OHSUMED dataset as examples.

We examined the reason that AdaRank performs better than pairwise learning to rank algorithms, such as RankBoost. Specifically we compared the error rates between different rank pairs made by RankBoost and AdaRank on the test data. The results over all of the five trials in the 5-fold cross validation are shown in Fig. 14. We use 'd-n' to stand for the pairs between 'definitely relevant' and 'not relevant', 'd-p' the pairs between 'definitely relevant' and 'partially relevant', and 'p-n' the pairs between 'partially relevant' and 'not relevant'. From Fig. 14, we can see that AdaRank makes fewer errors for 'd-n' and 'd-p', which are related to the tops of rankings and are important. This is because direct optimization algorithms can naturally focus upon the training at the top by optimizing the performance measures.

We also gathered statistics on the number of document pairs per query in the entire OHSUMED dataset. The queries are clustered into different groups based on the the number of their associated document pairs. Fig. 15 shows the distribution of the query groups. In the figure, for example, '0-1k' is the group of queries whose number of document pairs are between 0 and 999. We can see that the numbers of document pairs really vary from query to query. Next we evaluated the accuracies of AdaRank and RankBoost in terms of MAP for each of the query groups. The results are reported in Fig. 16. We found that the average MAP of AdaRank over the groups is higher than RankBoost. Furthermore, it is interesting to see that AdaRank performs particularly better than RankBoost for queries with small numbers of document pairs (e.g., '0-1k', '1k-2k', '2k-3k', '3k-4k', '4k-5k', and '5k-6k'). The results indicate that AdaRank can effectively avoid creating a model biased towards queries with more document pairs.

8. 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.

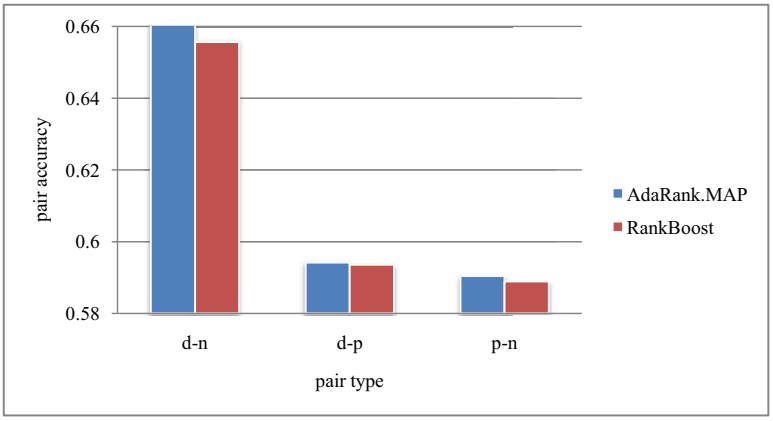

Figure 14: Accuracy on ranking document pairs with OHSUMED dataset.

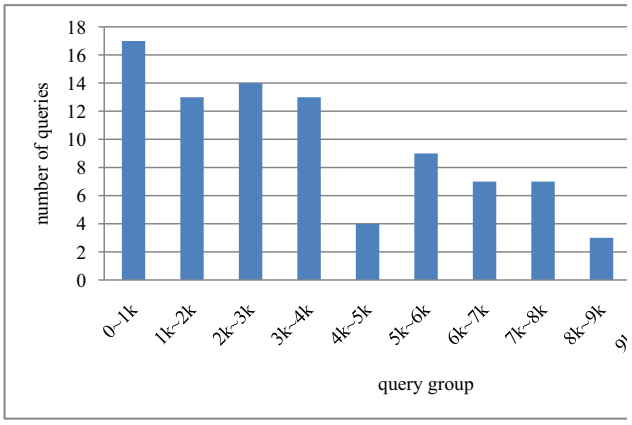

Figure 15: Distribution of queries with different number of document pairs in all of the test data.

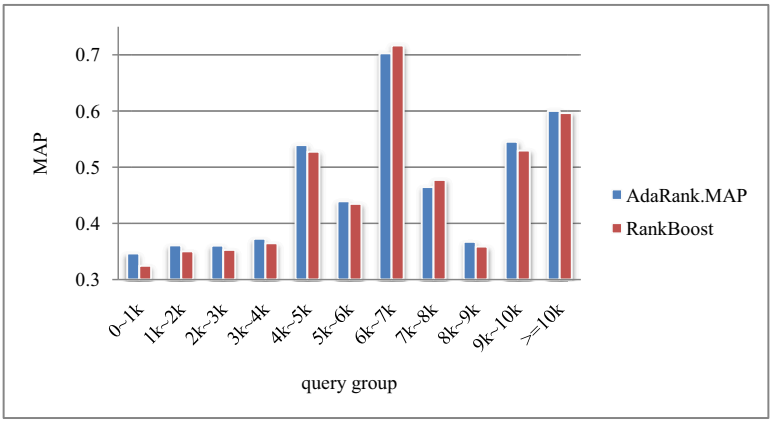

Figure 16: Differences in MAP for different query groups.

We have built a theoretical framework of the direct optimization approach. According to our analysis, the direct optimization approach is one that minimizes the basic loss function defined on the IR measures. It turns out that the algorithms AdaRank and PermuRank, which we propose, actually try to minimize two types of upper bounds of the basic loss function, respectively called type one bound and type two bound in the paper. Existing algorithm SVM^{map} also minimizes a type two bound upon the basic loss function. We have also analyzed the relation between the two types of bounds both theoretically and empirically. With this framework, we are able to analyze existing methods, such as SVM^{map}, and derive new algorithms.

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.

There are several interesting future research directions. First, one can further explore the generalization ability of direct optimization algorithms for learning to rank. Second, it is interesting to develop other ranking algorithms that can utilize both type one and type two bounds. For example, we know that the type two bound is tighter than the type one bound but harder to optimize. One possibility is to learn the ranking model first according to the type one bound and then switch to the type two bound. Finally, it is better to further extend the framework, specifically, to study direct optimization achieved by approximating the IR measures functions [27] or by other techniques [3, 10, 23].

References

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

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

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

[4] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G., 2005. Learning to rank using gradient descent. In: ICML '05: Proceedings of the 22nd international conference on Machine learning. ACM Press, New York, NY, USA, pp. 89–96.

[5] Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., Hon, H.-W., 2006. Adapting ranking SVM to document retrieval. In: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM Press, New York, NY, USA, pp. 186–193.

[6] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H., 2007. Learning to rank: from pairwise approach to listwise approach. In: ICML '07: Proceedings of the 24th international conference on Machine learning. ACM, New York, NY, USA, pp. 129–136.

[7] Chakrabarti, S., Khanna, R., Sawant, U., Bhattacharyya, C., 2008. Structured learning for non-smooth ranking losses. In: KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, pp. 88–96.

[8] Chapelle, O., Le, Q., Smola, A., 2007. Large margin optimization of ranking measures. In: NIPS workshop on Machine Learning for Web Search.

[9] Cossock, D., Zhang, T., 2006. Subset ranking using regression. In: COLT '06: Proceedings of the 19th Annual Conference on Learning Theory. pp. 605–619.

[10] de Almeida, H. M., Gonçalves, M. A., Cristo, M., Calado, P., 2007. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, New York, NY, USA, pp. 399–406.

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

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

[13] Freund, Y., Schapire, R. E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.
[14] Hastie, T., Tibshirani, R., Friedman, J. H., August 2001. The Elements of Statistical Learning. Springer.
[15] Herbrich, R., Graepel, T., Obermayer, K., 2000. Large margin rank boundaries for ordinal regression. MIT Press, Cambridge, MA.
[16] Jarvelin, K., Kekalainen, J., 2000. IR evaluation methods for retrieving highly relevant documents. In: SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. ACM Press, New York, NY, USA, pp. 41–48.
[17] Joachims, T., 2002. Optimizing search engines using clickthrough data. In: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, New York, NY, USA, pp. 133–142.
[18] Lafferty, J., Zhai, C., 2001. Document language models, query models, and risk minimization for information retrieval. In: SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM Press, New York, NY, USA, pp. 111–119.
[19] Le, Q., Smola, A., 2007. Direct optimization of ranking measures. Tech. rep.
[20] Li, P., Burges, C., Wu, Q., 2007. Learning to rank using classification and gradient boosting. In: NIPS.
[21] Liu, T.-Y., 2009. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3 (3), 225–331.
[22] Liu, T.-Y., Xu, J., Qin, T., Xiong, W., Li, H., 2007. 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).
[23] Metzler, D., 2005. Direct maximization of rank-based metrics. Tech. rep., CIIR Technical Report.
[24] Ponte, J. M., Croft, W. B., 1998. A language modeling approach to information retrieval. In: SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM Press, New York, NY, USA, pp. 275–281.
[25] Qin, T., Liu, T.-Y., Li, H., Nov. 2008. A general approximation framework for direct optimization of information retrieval measures. Tech. rep., Microsoft Research Asia, MSR-TR-2008-164.
[26] Robertson, S. E., Hull, D. A., 2000. The trec-9 filtering track final report. In: TREC '00: Proceedings of the 9th Text REtrieval Conference. pp. 25–40.
[27] Taylor, M., Guiver, J., Robertson, S., Minka, T., 2007. Softrank: Optimising non-smooth rank metrics. In: Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007).
[28] Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., 2005. Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. 6, 1453–1484.
[29] Xia, F., Liu, T.-Y., Wang, J., Zhang, W., Li, H., 2008. Listwise approach to learning to rank - theorem and algorithm. In: ICML '08: Proceedings of the 25th international conference on Machine learning. pp. 1192–1199.
[30] Xu, J., Li, H., 2007. Adarank: a boosting algorithm for information retrieval. In: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, New York, NY, USA, pp. 391–398.
[31] Xu, J., Liu, T.-Y., Lu, M., Li, H., Ma, W.-Y., 2008. Directly optimizing evaluation measures in learning to rank. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, New York, NY, USA, pp. 107–114.
[32] Yeh, J.-Y., Lin, J.-Y., Ke, H.-R., Yang, W.-P., 2007. Learning to rank for information retrieval using genetic programming. In: Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007).
[33] Yue, Y., Finley, T., Radlinski, F., Joachims, T., 2007. A support vector method for optimizing average precision. In: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, New York, NY, USA, pp. 271–278.

A. 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

$$\begin{aligned} F(q, \mathbf{d}, \pi) &= w^\top \frac{1}{n(q) \cdot (n(q)-1)} \sum_{k,l:k<l} [z_{kl}(\phi(q, d_k) - \phi(q, d_l))] \\ &= \frac{1}{n(q) \cdot (n(q)-1)} \sum_{k,l:k<l} \left[z_{kl}(w^\top \phi(q, d_k) - w^\top \phi(q, d_l)) \right] \\ &= \frac{1}{n(q) \cdot (n(q)-1)} \sum_{k,l:k<l} [z_{kl}(s_k - s_l)], \end{aligned} \tag{11}$$

where $z_{kl} = +1$ if $\pi(k) < \pi(l)$, and $z_{kl} = -1$ otherwise. Since π is only related to the variables z_{kl}'s in the equation, the equation is maximized with respect to π, 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 τ 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 τ. Since all the terms in the summation of Equation (11) are not negative, F is maximized: $\tau = \arg\max_{\tau \in \Pi} F(q, \mathbf{d}, \tau) = \sigma$.

2 σ 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 σ 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.

B. 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 σ_i is the permutation selected for query q_i by model F. There are two cases:

Case 1 $\sigma_i \in \Pi_i^*$ If $\sigma_i \in \Pi_i^*$, $E(\sigma_i, \mathbf{y}_i) = 1$, 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 $\sigma_i = \arg\max_{\sigma\in\Pi_i} F(q_i, \mathbf{d}_i, \sigma)$, we have $F(q_i, \mathbf{d}_i, \pi_i^*) - F(q_i, \mathbf{d}_i, \sigma_i) \leq 0$. Thus

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

Summarizing case 1 and case 2, we obtain

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

C. Proof of Theorem 3

Proof For $\forall i = 1, ..., m$, since

$$\min_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\backslash\Pi_i^*} \left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right) \geq E(\sigma_i, y_i),$$

we have

$$F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i) \geq E(\sigma_i, y_i), \text{ for } \forall \pi_i^* \in \Pi_i^*, \forall \pi_i \in \Pi_i\backslash\Pi_i^*.$$

Since φ is monotonically nondecreasing, then

$$\varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right) \leq \varphi\left(-E(\sigma_i, y_i)\right), \text{ for } \forall \pi_i^* \in \Pi_i^*, \forall \pi_i \in \Pi_i\backslash\Pi_i^*.$$

Since φ is positive, for $\forall \pi_i^* \in \Pi_i^*$, $\forall \pi_i \in \Pi_i\backslash\Pi_i^*$, we have

$$\begin{aligned} \left(E(\pi_i^*, \mathrm{y}_i) - E(\pi_i, \mathrm{y}_i)\right) \cdot \varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right) &\leq \varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right) \\ &\leq \varphi\left(-E(\sigma_i, y_i)\right), \end{aligned}$$

Thus,

$$\varphi\left(-E(\sigma_i, y_i)\right) \geq \max_{\pi_i^*\in\Pi_i^*;\pi_i\in\Pi_i\backslash\Pi_i^*} \left(E(\pi_i^*, \mathrm{y}_i) - E(\pi_i, \mathrm{y}_i)\right) \cdot \varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right).$$

Therefore,

$$\sum_{i=1}^{m} \varphi\left(-E(\sigma_i, y_i)\right) \geq \sum_{i=1}^{m} \max_{\pi_i^*\in\prod_i^*;\pi_i\in\prod_i\backslash\prod_i^*} \left(E(\pi_i^*, \mathrm{y}_i) - E(\pi_i, \mathrm{y}_i)\right) \cdot \varphi\left(-\left(F(q_i, d_i, \pi_i^*) - F(q_i, d_i, \pi_i)\right)\right).$$

D. Proof of Theorem 4

Proof Set $Z_T = \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_T), \mathbf{y}_i)\}$ and $\phi(t) = \frac{1}{2}(1+\varphi(t))$. According to the definition of α_t, we know that $e^{\alpha_t} = \sqrt{\frac{\phi(t)}{1-\phi(t)}}$.

$$
\begin{aligned}
Z_T &= \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{T-1} + \alpha_T h_T), \mathbf{y}_i)\} \\
&= \sum_{i=1}^{m} \exp\left\{-E(\pi(q_i, \mathbf{d}_i, f_{T-1}), \mathbf{y}_i) - \alpha_T E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i) - \delta_i^T\right\} \\
&\leq \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{T-1}), \mathbf{y}_i)\} \exp\{-\alpha_T E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i)\}\, e^{-\delta_{min}^T} \\
&= e^{-\delta_{min}^T} Z_{T-1} \sum_{i=1}^{m} \frac{\exp\{-E(\pi(q_i, \mathbf{d}_i, f_{T-1}), \mathbf{y}_i)\}}{Z_{T-1}} \exp\{-\alpha_T E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i)\} \\
&= e^{-\delta_{min}^T} Z_{T-1} \sum_{i=1}^{m} P_T(i) \exp\{-\alpha_T E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i)\}.
\end{aligned}
$$

Moreover, if $E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i) \in [0, +1]$ then,

$$
\begin{aligned}
Z_T &\leq e^{-\delta_{min}^T} Z_{T-1} \sum_{i=1}^{m} P_T(i) \left(\frac{1+E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i)}{2} e^{-\alpha_T} + \frac{1-E(\pi(q_i, \mathbf{d}_i, h_T), \mathbf{y}_i)}{2} e^{\alpha_T} \right) \\
&= e^{-\delta_{min}^T} Z_{T-1} \left(\phi(T) \sqrt{\frac{1-\phi(T)}{\phi(T)}} + (1-\phi(T)) \sqrt{\frac{\phi(T)}{1-\phi(T)}} \right) \\
&= Z_{T-1} e^{-\delta_{min}^T} \sqrt{4\phi(T)(1-\phi(T))} \\
&\leq Z_{T-2} \prod_{t=T-1}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&\leq Z_1 \prod_{t=2}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&= m \sum_{i=1}^{m} \frac{1}{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, \alpha_1 h_1), \mathbf{y}_i)\} \prod_{t=2}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&= m \sum_{i=1}^{m} \frac{1}{m} \exp\{-\alpha_1 E(\pi(q_i, \mathbf{d}_i, h_1), \mathbf{y}_i) - \delta_i^1\} \prod_{t=2}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&\leq m e^{-\delta_{min}^1} \sum_{i=1}^{m} \frac{1}{m} \exp\{-\alpha_1 E(\pi(q_i, \mathbf{d}_i, h_1), \mathbf{y}_i)\} \prod_{t=2}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&\leq m \left\{ e^{-\delta_{min}^1} \sqrt{4\phi(1)(1-\phi(1))} \right\} \prod_{t=2}^{T} e^{-\delta_{min}^t} \sqrt{4\phi(t)(1-\phi(t))} \\
&= m \prod_{t=1}^{T} e^{-\delta_{min}^t} \sqrt{1-\varphi(t)^2}.
\end{aligned}
$$

$$\begin{aligned}\therefore \frac{1}{m}\sum_{i=1}^{m} E(\pi(q_i, \mathbf{d}_i, f_T), \mathbf{y}_i) &\geq \frac{1}{m}\sum_{i=1}^{m}\{1 - \exp(-E(\pi(q_i, \mathbf{d}_i, f_T), \mathbf{y}_i))\} \\ &= 1 - \frac{Z_T}{m} \\ &\geq 1 - \prod_{t=1}^{T} e^{-\delta^t_{min}} \sqrt{1 - \varphi(t)^2}.\end{aligned}$$