A General Magnitude-Preserving Boosting Algorithm for Search Ranking

Chenguang Zhu[1,2] *, Weizhu Chen[2], Zeyuan Allen Zhu[2,3], Gang Wang[2], Dong Wang[1,2], Zheng Chen[2]

[1]Institute for Theoretical Computer Science
Tsinghua University
Beijing, China, 100084
{zcg.cs60, wd890415}@gmail.com

[2]Microsoft Research Asia
No. 49 Zhichun Road
Haidian District
Beijing, China, 100080
{v-chezhu, wzchen, v-zezhu, gawa, v-dongmw, zhengc}@microsoft.com

[3]Fundamental Science Class
Department of Physics
Tsinghua University
Beijing, China, 100084
zhuzeyuan@hotmail.com

ABSTRACT

Traditional boosting algorithms for the ranking problems usually employ the pairwise approach and convert the document rating preference into a binary-value label, like RankBoost. However, such a pairwise approach ignores the information about the magnitude of preference in the learning process. In this paper, we present the directed distance function (DDF) as a substitute for binary labels in pairwise approach to preserve the magnitude of preference and propose a new boosting algorithm called MPBoost, which applies GentleBoost optimization and directly incorporates DDF into the exponential loss function. We give the boundedness property of MPBoost through theoretic analysis. Experimental results demonstrate that MPBoost not only leads to better NDCG accuracy as compared to state-of-the-art ranking solutions in both public and commercial datasets, but also has good properties of avoiding the overfitting problem in the task of learning ranking functions.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6[**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Directed distance function (DDF), magnitude-preserving, pairwise-preference

1. INTRODUCTION

Boosting [9, 10] is one of the state-of-the-art algorithms in machine learning. Based on boosting, many algorithms have been proposed to solve supervised learning tasks, including [10] for classification, [24] for facial recognition and [12] for spam filtering. All of these have theoretically and empirically demonstrated that boosting algorithm and its variants have excellent advantages in terms of convergence of loss function, low generalization error, little restriction on the form of weak learners, etc.

As an interesting and significant application, boosting is also successfully employed in the learning to rank problem for Web search, where a set of queries Q, and a set of documents D_q for each query $q \in Q$ are given. For each document of a given query, there is also a rating r_i to encode the relevance level of this document. Current popular strategy to apply boosting for ranking, like RankBoost [7, 8], is to regard the pairwise preference information as the binary classification data, which belongs to the pairwise approach for learning to rank [11]. Specifically, if document x_1 is rated higher than another document x_2 with respect to the same given query q, (x_1, x_2) is labeled as a positive instance. Otherwise, (x_1, x_2) is labeled as a negative instance. However, this kind of pairwise transformation from ranking to classification neglects the scale or magnitude of the difference between rating pairs, like the magnitude value of r_1-r_2. As a matter of fact, far richer representations can be achieved if the magnitude of rating differences can be considered in boosting algorithm. Therefore, designing a new boosting algorithm for ranking with consideration of the magnitude of difference between rating pairs is very desirable.

Some attempts have been made to address similar issues. In the RankBoost algorithm [7, 8], the magnitude issue is mentioned in the introduction of the feedback function Φ, but later the theoretical and empirical analysis focus on the case where

*This work was done when the first author was visiting Microsoft Research Asia.

$\Phi = -1,0,1$, which is under the binary-label model. [21] leveraged multiple hyperplanes to preserve the magnitude of rating differences on the basis of the RankSVM algorithm [13, 17] and demonstrated the importance of preference magnitude. More recently, [4] analyzed the stability bounds of magnitude-preserving loss functions for generalization error and proposed two magnitude-preserving ranking algorithms, MPRank and SVRank, with reports of the improvement on mis-ordering loss.

In this paper, we observe that it is fairly straightforward to apply the preference magnitude into the exponential loss function of boosting to improve the accuracy of ranking. To preserve the magnitude of rating differences, we propose directed distance function (DDF) as the substitute for binary labels in the pairwise approach to ranking. The exact form of DDF can vary with different representations, under only two basic requirements. Thus, we present three kinds of DDFs and list the appropriate scope of usage. Then, on the basis of DDF, we propose a novel ranking algorithm, MPBoost, based on GentleBoost [10]. It directly leverages the exponential loss function with DDF as the substitute for binary labels, which makes this algorithm suitable for magnitude-preserving ranking. We also prove a theorem about its effectiveness on the training set. Experimental results on two public and one commercial datasets all illustrate that MPBoost with DDF can significantly outperform traditional pairwise ranking algorithms as well as the state-of-the-art ranking methods like ListNet. Also, the experiment demonstrates that the application of DDF can lead MPBoost to avoid the overfitting problem.

The rest of the paper is organized as follows: In Section 2, we firstly review some related works. In Section 3, we present the concept of magnitude-preserving labels: directed distance function (DDF), with three exemplary functions. In Section 4, we propose the MPBoost algorithm, which applies exponential loss functions with DDF. We report the experiment results in Section 5 and conclude the paper in Section 6.

2. RELATED WORKS

2.1 Learning to Rank

Learning to rank is a popular topic in both machine learning and information retrieval research. One of the main approaches to ranking problem is referred to as the pairwise approach. In the pairwise approach, the learning to rank task is transformed into a binary classification task based on document pairs (whether the first document or the second should be ranked first given a query). [13, 17] proposed using the SVM techniques to build the classification model, which is referred to as RankSVM. [7, 8] proposed performing the task in a similar way but via the AdaBoost algorithm. [1] also adopted the approach and developed a method called RankNet, which leverages the cross entropy as the loss function and gradient descent as the algorithm to train a neural network model.

Recently, the concept of magnitude-preserving ranking was introduced in [4, 21]. [21] leveraged multiple hyperplanes to preserve the magnitude of rating differences on the basis of the RankSVM and proposed a method called "Multiple Hyperplane Ranker" (MHR). In [4], the authors analyzed the stability bounds of magnitude-preserving loss functions for generalization error. Based on the results, they proposed two algorithms, MPRank and SVRank and reported empirical results which showed improvements on mis-ordering loss. However, the loss functions in [4] are regularization-based, and the σ - admissibility requirement of cost function limits the forms of functions to some extent. For example, exponential cost functions can hardly meet the requirement with a small constant σ .

2.2 Boosting

AdaBoost [9] is a state-of-the-art classification algorithm which stage-wise combines a number of weak learners and generates a strong hypothesis. [10, 22] analyzed the theoretical advantage of the boosting algorithm from the aspect of margin and VC-dimension. [20] presents the abstraction for different versions of boosting algorithms. In real application, GentleBoost [10] is a variation of boosting algorithm which employs the Newton step to minimize the exponential loss function. It has been shown in [10] that GentleBoost has similar performance to AdaBoost, and often outperforms the latter especially when stability is an issue.

3. MAGNITUDE PRESERVING LABELS

Since our work is under the pairwise ranking approach, we will first walk through the basic concepts in the pairwise learning to rank. Next, we will propose the directed distance function to preserve the magnitude of rating difference and illustrate three examples.

3.1 The Pairwise Approach for Ranking

The pairwise approach for ranking is defined as follows in [3]. A ranking dataset includes a set of queries $q \in Q$, and a set of documents for each query $x_i \in D_q$. The associated relevance rating of document x_i for query q is represented as r_{qi} . The relevance ratings are discrete and ordered, with values such as {*Probably Relevant, Possibly Relevant, Not Relevant*}. Furthermore, there exists a total ordering $\triangleright$ between various relevance levels, e.g. *Probably Relevant* $\triangleright$ *Possibly Relevant* $\triangleright$ *Not Relevant.* In this paper, we assume that numerical ratings values are available because ranking performance measurements often take ratings as a component in calculation. Also, for the learning tasks, documents are represented by a vector of feature weights obtained by some query-document function Φ . For instance, the document x_i for the query q is represented as $\Phi(q,x_i) = x_{qi} = (x_{qi,1},\ldots\ldots,x_{qi,d})$.

The goal of the learning to rank procedure in the pairwise framework is to introduce a score function $s(\bullet)$ over the document space such that:

$$r_{qi} \triangleright r_{qj} \Rightarrow s(x_{qi}) > s(x_{qj}) \tag{1}$$

i.e. if document x_{qi} is preferred over x_{qj}, the score value for x_{qi} should be larger than x_{qj}. This clarifies the relationship between the pairwise-preference learning and the binary classification: a classifier can be introduced to maintain the given preference relations on the left of inequality (1).

Thus, for pairwise approach, we define a preference set containing document pairs for each query q : $S_q = \{((x_{qi},x_{qj}),y_{qij}) \mid r_{qi} \neq r_{qj}\}$, where the binary label y_{qij} satisfies:

$$y_{qij} = \begin{cases} 1, r_{qi} \triangleright r_{qj} \\ -1, r_{qj} \triangleright r_{qi} \end{cases} \tag{2}$$

3.2 Directed Distance Function (DDF)

Although the binary label defined in (2) is suitable for leveraging well-studied classification tools, this transformation from ranking to classification also loses considerable amount of information. For example, mis-ranking two documents with ratings 5 and 1 should receive more "punishment" than mis-ranking two documents with ratings 5 and 4, because the former will cause larger decrease in ranking measurements like NDCG [16]. In this sense, the binary label constructed by (2) only reflects the desired order of two documents, omitting the useful information hidden in the magnitude of rating differences.

On the other hand, far richer representations can be achieved if the magnitude of rating differences is considered. Therefore, we need to modify the traditional definition of labels in (2) to preserve the magnitude information.

Specifically, we define the label to be the *directed distance* from a rating to another: $dist(r_a, r_b)$, which depicts the impetus of placing a document with rating r_a before another document with rating r_b.

As shown in Figure 1, although the concrete values of directed distances can vary, the magnitude of rating differences should be preserved, as well as the advantage of placing high-rated documents in front.

Now, we formally present the requirements (3) and (4) on the concept of *directed distance function (DDF)*: $dist(\bullet,\bullet)$, which will be used as an extension of tradition binary labels (2) in the subsequent analysis:

$$\operatorname{sgn}(dist(r_i, r_j)) = \operatorname{sgn}(r_i - r_j) \tag{3}$$

where $\operatorname{sgn}(x) = \begin{cases} 1 & x>0, \\ 0 & x=0, \\ -1 & x<0 \end{cases}$

$$|r_i - r_j| > |r_i' - r_j'| \Rightarrow |dist(r_i, r_j)| > |dist(r_i', r_j')| \tag{4}$$

Then, DDF is directly applied as the substitute for binary labels and the modified preference set is (5), which will be leveraged into the exponential loss function in our boosting approach for learning to rank.

$$S_q' = \{((x_{qi}, x_{qj}), dist(r_{qi}, r_{qj})) \mid r_{qi} \neq r_{qj}\} \tag{5}$$

Compared to σ-admissibility presented in Definition 2 of [4], the basic requirements on DDF allow many more candidate functions in real application. For instance, we will propose three possible directed distance functions which have performed superiorly in the empirical analysis.

3.2.1 Linear Directed Distance (LDD)

An intuitive integration of preference magnitude into DDF is the linear function based on the difference of the preference difference. We call this function as Linear Directed Distance (LDD) and show its equation in formula (6). The coefficient α is a positive constant for regularizing the scope of $dist(\bullet,\bullet)$ and is used to meet the requirements (3) and (4).

$$dist(r_i, r_j) = \alpha(r_i - r_j) \tag{6}$$

LDD is an easy and simple function to consider the preference magnitude. But when the numerical difference between ratings is large, it's very difficult to tune a good α value to map all rating differences into a proper interval. We will show these findings in

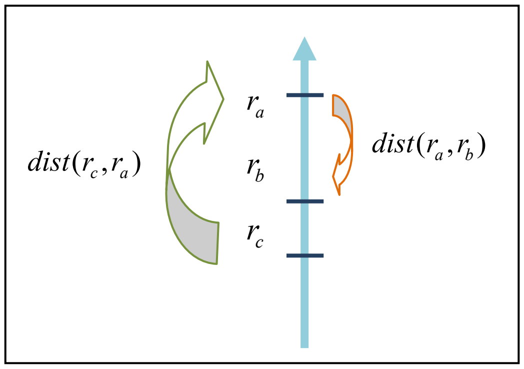

Figure 1. Three ratings $r_a \triangleright r_b \triangleright r_c$, where the directed distances $dist(r_c, r_a)$ and $dist(r_a, r_b)$ are marked. The exact values of the distances can vary, but it should follow that: 1. $|dist(r_c, r_a)| > |dist(r_a, r_b)|$, to preserve magnitude of rating differences.

2. $dist(r_c, r_a) < 0$ and $dist(r_a, r_b) > 0$, to present the advantage of placing documents with higher ratings in front.

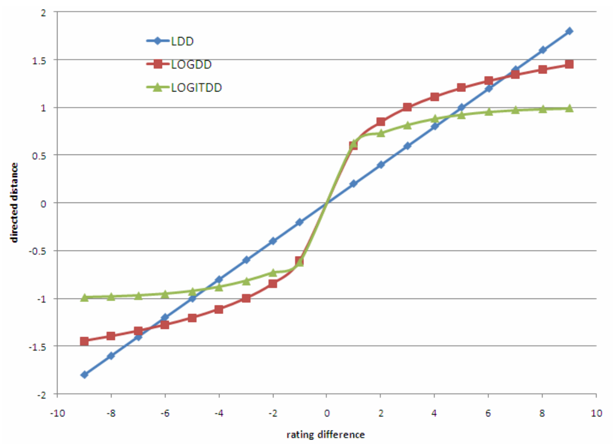

Figure 2. Curves of LDD, LOGDD and LOGITDD under different values of rating differences. The parameters are $\alpha = 0.2$, $\lambda = 3$ and $\beta = 0.5$.

the experiment. In this case, functions that can smooth large rating differences should help, as proposed in Section 3.2.2 and 3.2.3.

3.2.2 Logarithmic Directed Distance (LOGDD)

Logarithmic Directed Distance function (7) takes the form of logarithms, in order to smooth the grading difference between ratings. Like LDD, LOGDD utilizes the positive parameter λ.

$$dist(r_i, r_j) = \operatorname{sgn}(r_i - r_j)\log(1 + \lambda |r_i - r_j|) \tag{7}$$

Compared with LDD, LOGDD can be utilized when the rating values come from a large range or the grades of ratings are non-uniform. It is also obvious to note that due to the logarithmic nature, the output range of LOGDD will eventually be smaller than that of LDD and be more suitable to apply in the exponential loss function. On the other hand, LOGDD is much smoother than LDD in term of the output value, as shown in Figure 2.

3.2.3 *Logistic Directed Distance (LOGITDD)*

Logistic Directed Distance function (8) leverages the well-studied logistic function. Like LDD and LOGDD, LOGITDD applies the positive parameter β.

$$dist(r_i, r_j) = \text{sgn}(r_i - r_j)\frac{1}{1+e^{-\beta|r_i - r_j|}} \quad (8)$$

Compared with LDD and LOGDD, the output range of LOGITDD is always in $(-1,-0.5]\cup[0.5,1)$, which makes LOGITDD be smoother than both the above functions. It follows that it's easier to tune the related parameters when considering the preference magnitude. Also note that when applying LOGITDD, the only parameter β should be carefully chosen to make $|dist(r_i, r_j)|$ non-negligibly different for disparate values of $|r_i - r_j|$. For example, if the grading differences between ratings are usually large, a relatively small β should be selected to avoid saturation and we will use validation dataset to tune this parameter in our experiments.

In summary, the introduction of directed distance function is to substitute conventional binary-value labels and assign magnitude-preserving property to the ranking algorithm like the one we will introduce in Section 4.

4. MPBOOST ALGORITHM

Following the introduction to DDF in Section 3, we will present a novel boosting algorithm, MPBoost. Specifically, we apply the GentleBoost approach [10] to define loss function and design optimization methods.

For convenience, we combine the preference sets over all queries: $S = \bigcup_q S_q'$ and define the index set I over S: $I = \{(i,j)\,|\,((x_i,x_j),\text{dist}(r_i,r_j)) \in \text{S}\}$. Note that for sake of simplicity, we will sometimes omit the query subscript in the following discussion, i.e. whenever (x_i, x_j) is involved, we assume that the documents x_i and x_j belong to the same query q, and $r_{qi} \neq r_{qj}$. Similarly, $dist(r_i, r_j)$ means $dist(r_{qi}, r_{qj})$.

Next, to leverage DDF in the MPBoost algorithm, we require that the condition

$$|dist(r_i, r_j)| \approx 1 \quad (9)$$

is satisfied, which can be attained via carefully setting the parameters within the applied directed distance functions, like α, λ and β in LDD, LOGDD and LOGITDD. This condition will be utilized in the following analysis.

Now, the loss function MPBoost employs is as follows:

$$J(F) = \sum_I e^{-dist(r_i,r_j)(F(x_i)-F(x_j))} \quad (10)$$

where the strong hypothesis $F(x)$ is a score function based on an additive model. In other words, F is initially set as 0. Then, in the ith round, $F(x) \leftarrow F(x) + f_i(x)$, where $f_i(x)$ is called the weak learner in the ith round. Thus, when the MPBoost algorithm ends after *m* rounds, $F(x) = \sum_{i=1}^{m} f_i(x)$.

Algorithm 1 MPBoost algorithm for generating a ranking function.

```
Input: Query set $Q$ and $\{(x_{qi}, r_{qi})\}_{i=1}^{n_q}$, for each $q \in Q$.
Output: ranking function $F(x)$
1: Generate $S = \bigcup_q \{((x_{qi}, x_{qj}), dist(r_{qi}, r_{qj}))\,|\,r_{qi} \neq r_{qj}\}$
2: Generate index set $I = \{(i,j)\,|\,((x_i,x_j),\text{dist}(r_i,r_j)) \in \text{S}\}$
3: Initialize $w_{ij}^{(1)} = \frac{1}{|I|}$, for $\forall (i,j) \in I$
4: for $t = 1...T$ do
5:     Fit the weak ranker $f_t$, such that:
           $f_t = \arg\min_f J_{wse}(f)$
             $= \arg\min_f \sum_I w_{ij}[dist(r_i,r_j) - (f(x_i) - f(x_j))]^2$
6:     Update: $w_{ij}^{(t+1)} \leftarrow w_{ij}^{(t)} e^{-dist(r_i,r_j)(f_t(x_i)-f_t(x_j))} / Z_t$
       where $Z_t = \sum_I w_{ij}^{(t)} e^{-dist(r_i,r_j)(f_t(x_i)-f_t(x_j))}$
7: end for
8: Output the final ranking function $F(x) = \sum_{t=1}^{T} f_t(x)$
```

In order to minimize loss function, MPBoost needs to find the best weak learner in each round. For example, if the current hypothesis is F, and the next weak learner to be added is f, then the additive loss function should be:

$$\begin{aligned}
J(F+f) &= \sum_I e^{-dist(r_i,r_j)[(F(x_i)+f(x_i))-(F(x_j)+f(x_j))]} \\
&\approx \sum_I e^{-dist(r_i,r_j)(F(x_i)-F(x_j))}(1 - dist(r_i,r_j)(f(x_i)-f(x_j)) \\
&\quad + \frac{dist(r_i,r_j)^2[f(x_i)-f(x_j)]^2}{2}) \\
&\approx \sum_I e^{-dist(r_i,r_j)(F(x_i)-F(x_j))}(\frac{1}{2}[dist(r_i,r_j)-(f(x_i)-f(x_j))]^2 + \frac{1}{2}) \\
&= \frac{1}{2}\sum_I e^{-dist(r_i,r_j)(F(x_i)-F(x_j))}([dist(r_i,r_j)-(f(x_i)-f(x_j))]^2 + 1)) \quad (11)
\end{aligned}$$

where we apply second-order Taylor approximation and the condition (9).

Thus, we can determine the best f to be added to F by minimizing $J(F+f)$, equivalent to minimizing a weighed squared loss:

$$J_{wse}(f) = \sum_I w_{ij}[dist(r_i,r_j) - (f(x_i) - f(x_j))]^2 \quad (12)$$

where $w_{ij} = e^{-dist(r_i,r_j)(F(x_i)-F(x_j))}$, the weight assigned to each document pair.

Now, we formally present our algorithm: MPBoost, which is based on GentleBoost [10], with changes in the loss function calculation and weight modification. In MPBoost, the initial weight of each document pair is uniform. During each iteration, a weak ranker is chosen to minimize (12). Then, the weights are updated with help of the normalizer Z_t:

$$Z_t = \sum_I w_{ij}^{(t)} e^{-dist(r_i, r_j)(f_t(x_i) - f_t(x_j))} \tag{13}$$

The final ranking function is the summation over all weak rankers. The procedure details of MPBoost are presented in Algorithm 1.

Note that MPBoost bears some resemblance to RankBoost, which applies the optimization scheme of AdaBoost. In the RankBoost framework, a bound on the ranking loss is offered in Theorem 1 [8]. Now, with the employment of GentleBoost approach and DDF, $dist(\bullet,\bullet)$, MPBoost still inherits the bounded ranking loss property of RankBoost. We formalize this the following theorem.

Theorem 1. *Assuming the notation of Algorithm 1, the normalized ranking loss (mis-ordering) of F is bounded:*

$$\sum_{\{(i,j)\in I | r_i > r_j\}} w_{ij}^{(1)}[[F(x_i) \le F(x_j)]] + \sum_{\{(i,j)\in I | r_i < r_j\}} w_{ij}^{(1)}[[F(x_i) \ge F(x_j)]] \le \prod_{t=1}^{T} Z_t$$

where $[[\pi]]$ is defined to be 1 if predicate π holds and 0 otherwise.

Proof: The proof is similar to the one for Theorem 1 in [8], but with the introduction of DDF.

Note that $[[x \ge 0]] \le e^{\alpha x}$ and $[[x \le 0]] \le e^{-\alpha x}$ hold for all $\alpha > 0$ and all real x. Furthermore, $dist(r_i, r_j)$ has the same sign as $r_i - r_j$. Thus,

$$\begin{aligned}
&\sum_{\{(i,j)\in I | r_i > r_j\}} w_{ij}^{(1)}[[F(x_i) \le F(x_j)]] + \\
&\sum_{\{(i,j)\in I | r_i < r_j\}} w_{ij}^{(1)}[[F(x_i) \ge F(x_j)]] \\
&\le \sum_{\{(i,j)\in I | r_i > r_j\}} w_{ij}^{(1)} e^{-[dist(r_i, r_j)(F(x_i) - F(x_j))]} + \\
&\quad \sum_{\{(i,j)\in I | r_i < r_j\}} w_{ij}^{(1)} e^{[-dist(r_i, r_j)](F(x_i) - F(x_j))} \\
&= \sum_I w_{ij}^{(1)} e^{-[dist(r_i, r_j)(F(x_i) - F(x_j))]} \\
&= \sum_I w_{ij}^{(T+1)} \prod_{t=1}^{T} Z_t \\
&= \prod_{t=1}^{T} Z_t
\end{aligned} \tag{14}$$

■

In view of the bound established in Theorem 1, we are guaranteed to produce a combined ranking with low ranking loss if on each round t we choose a weak ranker f_t to minimize Z_t.

Actually, minimizing $J_{wse}(f)$ (12) is exactly minimizing a second-order Taylor approximation of Z_t (13), when (9) is satisfied:

$$\begin{aligned}
Z_t &= \sum_I w_{ij}^{(t)} e^{-dist(r_i, r_j)(f_t(x_i) - f_t(x_j))} \\
&\approx \sum_I w_{ij}^{(t)}[1 - dist(r_i, r_j)(f_t(x_i) - f_t(x_j)) \\
&\qquad + \frac{dist(r_i, r_j)^2}{2}(f_t(x_i) - f_t(x_j))^2] \\
&\approx \sum_I w_{ij}^{(t)}\{\frac{1}{2}[dist(r_i, r_j) - (f(x_i) - f(x_j))]^2 + \frac{1}{2}\} \\
&= \frac{1}{2}\sum_I w_{ij}^{(t)}[dist(r_i, r_j) - (f(x_i) - f(x_j))]^2 + \frac{1}{2}\sum_I w_{ij}^{(t)} \\
&= \frac{1}{2} J_{wse}(f) + \frac{1}{2}
\end{aligned} \tag{15}$$

Therefore, in theory, the MPBoost algorithm can achieve low mis-ordering loss (16) via stage-wise gradient descent method. And it has been proved that minimizing the number of mis-orderings is equivalent to maximizing a lower-bound on ranking performance metrics [6]. The empirical analysis in Section 5 also substantiates the result.

$$MisOrder(F) = \sum_{\{(i,j)\in I | r_i > r_j\}} [[F(x_i) \le F(x_j)]] + \sum_{\{(i,j)\in I | r_i < r_j\}} [[F(x_i) \ge F(x_j)]] \tag{16}$$

5. EXPERIMENTAL RESULTS

5.1 Data Collections

We utilized three data sets in the experiments: OHSUMED [14], a benchmark data set for document retrieval downloadable from LETOR 3.0 [18, 19]; Web-1, a Russian web search dataset [15]; Web-2, an English web search dataset obtained from a popular search engine.

OHSUMED [14] is a collection for information retrieval research. It is a subset of MEDLINE, a database on medical publications. OHSUMED contains a total of 348,556 records (out of over 7 million) from 270 medical journals during the period of 1987-1991. The fields of a record embrace title, abstract, MeSH indexing terms, author, source, and publication type. In OHSUMED, there are 106 queries, each with a number of associated documents. Also in the data set are a total of 16,140 query-document pairs, each of which described by 45 features. In OHSUMED, relevance grades are from {0, 1, 2}. We conducted experiments on each of the 5 subfolders in OHSUMED, each containing training/validation/test data.

Web-1 is the public training data from "Internet Mathematics 2009" contest [15]. This dataset contains computed and normalized features of query-document pairs as well as relevance judgments made by Yandex search engine assessors. There are 97,290 query-document pairs within a total of 9,124 queries. Each query-document pair is described by 245 features. All features are either binary value from {0, 1}, or continuous values from [0, 1]. In Web-1, relevance grades are continuous values from range [0, 4], with higher values represents higher relevance. We used five-fold cross validation, with 3+1+1 splits between train/validation/test sets.

Web-2 is from a commercial English search engine. There are 50, 000 query-document pairs within a total of 467 queries. Each query-document pair is described by 1779 features. In Web-2, relevance grades are from {0, 1, 2, 3, 4}. Again, we used five-fold cross validation, with 3+1+1 splits between train/validation/test sets.

5.2 Performance Measures

In the experiment, we apply the *Normalized Discounted Cumulative Gain(NDCG)* [16] as the performance measure.

NDCG can handle multiple levels of relevance and it favors algorithms that give higher ranks to highly relevant documents than marginally relevant ones. Furthermore, lower ranking position is of less value to this metric since it has less chance to be examined by a user. In accordance with these principles, computing NDCG values follow the following four steps:

1) Compute the gain of each document

2) Discount the gain of each document by its ranking position in the list

3) Cumulate these discounted gain of the list

4) Normalize the discounted cumulative gain of the list

Therefore, NDCG of a ranking list at position n is calculated as following:

$$N(n) = Z_n \sum_{j=1}^{n} \begin{cases} 2^{r(j)} - 1, j = 1 \\ \dfrac{2^{r(j)} - 1}{\log_2(j)}, j > 1 \end{cases}$$

where $r(j)$ is the rating of the jth document in the list, and the normalization constant Z_n is chosen so that the perfect ranking list receives a NDCG score of 1. The final NDCG score is the average over all queries.

In addition, in the validation phase of our experiments, we set NDCG@5 as the criteria for selecting the best parameters.

5.3 Experimental results on NDCG

We present the ranking accuracy of MPBoost algorithm on the three datasets along with some state-of-art baseline ranking methods. Specifically, on OHSUMED dataset, we apply RankBoost [7, 8], ListNet [2] and AdaRank-NDCG [23] as the baseline methods. The measurement on these algorithms comes from [19]. Also, we make MPBoost with binary labels as another baseline method, which is represented by MPBoost.BINARY in the figures.

Also in following figures, MPBoost.LDD, MPBoost.LOGDD and MPBoost.LOGITDD respectively represent the MPBoost algorithm with DDF in form (6), (7) and (8). These three versions of algorithms leverage the magnitude-preserving property. The parameters tuned in the experiments include α in (6), λ in (7), β in (8) and the number of boosting rounds T . Note that for MPBoost, the condition (9) should be considered during the tuning. And we use the validation set to tune these parameters independently.

In the experiments, we leverage decision stumps as the weak ranker for the MPBoost algorithm. The definition and optimization process of decision stumps is presented in the appendix.

5.3.1 Experiments on OHSUMED

On OHSUMED, we conducted five-fold cross validation experiments using the data split provided in LETOR. Every ranking measurement was calculated as the mean over five folders. As shown in Figure 3, the three versions of magnitude-preserving MPBoost algorithms outperform nearly all baselines in NDCG@1 to NDCG@10 by 1 point to 6 points gain. Although MPBoost.LDD and MPBoost.LOGDD lag behind ListNet and AdaRank-NDCG by about 0.5% in NDCG@1, MPBoost.LOGITDD consistently achieves the best NDCG accuracy across NDCG@1 to NDCG@10. Furthermore, MPBoost.BINARY, which is the MPBoost algorithm with binary labels, outperforms RankBoost in all metrics except for NDCG@6 and NDCG@7. Thus, we leverage MPBoost.BINARY as the baseline in the following experiments.

5.3.2 Experiments on Web-1

We conducted experiments on Web-1 via five-fold cross-validation, and the reported result is the average over five folders. As shown in Figure 4, the advantage of MPBoost with magnitude-preserving loss functions is not as clear as that in OHSUMED. Still, an average of 0.16% and 0.13% NDCG advantage is gained by MPBoost.LOGDD and MPBoost.LOGITDD over MPBoost.BINARY. However, MPBoost.LDD lags behind MPBoost.BINARY by an average of 0.24% NDCG.

5.3.3 Experiments on Web-2

The experiment on Web-2 was also conducted via five-fold cross-validation, and the reported result is an average over five folders. From Figure 5, we can see that MPBoost with magnitude-preserving loss functions significantly outperform the version with binary labels by an average of 1.5% (MPBoost.LDD), 2.2% (MPBoost.LOGDD) and 1.8%(MPBoost.LOGITDD) and MPBoost.LOGITDD performs the best in NDCG@1 and NDCG@2, while MPBoost.LOGDD achieves the best in the rest positions.

5.4 Discussion

In this section, we will try to present some analysis in light of the consistent and excellent experimental results generated from our MPBoost algorithm as compared to the state-of-the-art ranking algorithms.

Firstly, the loss function (10) applied in MPBoost utilizes the concept of directed distance function over ratings, thus preserving the magnitude of rating differences even in classification. In the experiment on three datasets, all three versions of MPBoost with DDF outperform MPBoost with binary labels. Hence, it substantiates that magnitude-preserving labels can lead the algorithm to grasp the inherent pattern in document vectors and in general yield higher performance.

Secondly, MPBoost pursued the ranking problem with quasi-GentleBoost approach. In [10], GentleBoost has been empirically proved to perform better than AdaBoost, while RankBoost is constructed on the basis of AdaBoost. Thus, MPBoost.BINARY and the other three versions of MPBoost with magnitude-preserving property consistently outperform RankBoost in OHSUMED dataset.

It is interesting to observe that among the three versions of MPBoost applying LDD, LOGDD and LOGITDD, the logarithm-based one achieved the best performance in two datasets; while the linear-based one fall behind the other two in all three datasets. We attribute the result to the ability to depict rating differences. The linear-based DDF, LDD, tends to output too large values for large rating grades, thus forcing the parameter α to take pretty small values (e.g. the best α in the

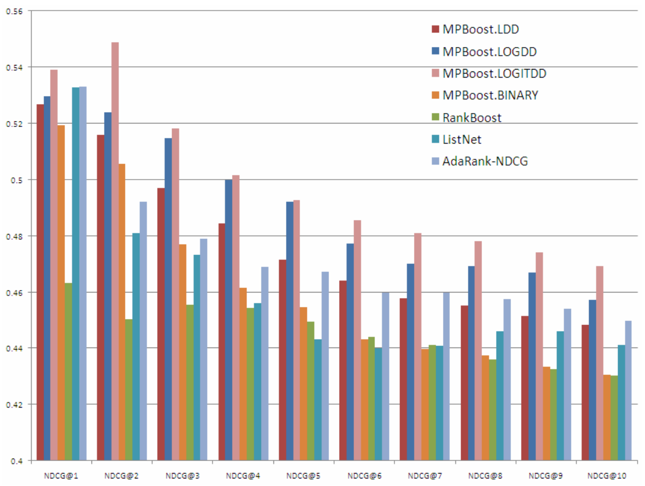

Figure 3. Ranking accuracies on OHSUMED dataset

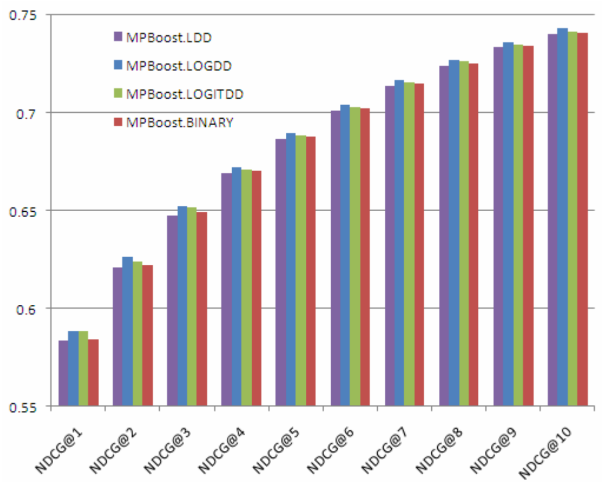

Figure 4. Ranking accuracies on Web-1 dataset

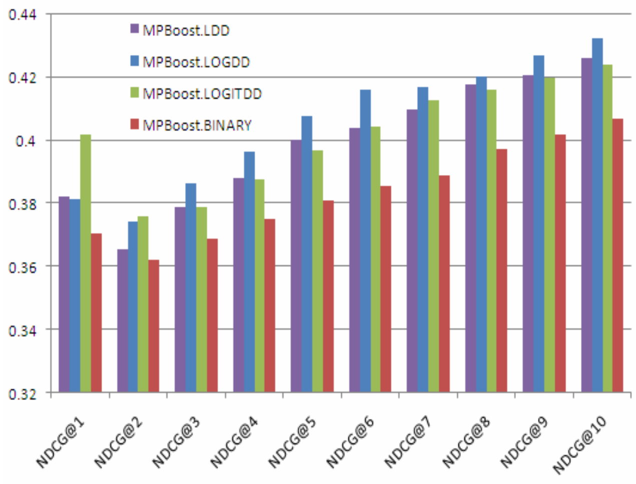

Figure 5. Ranking accuracies on Web-2 dataset.

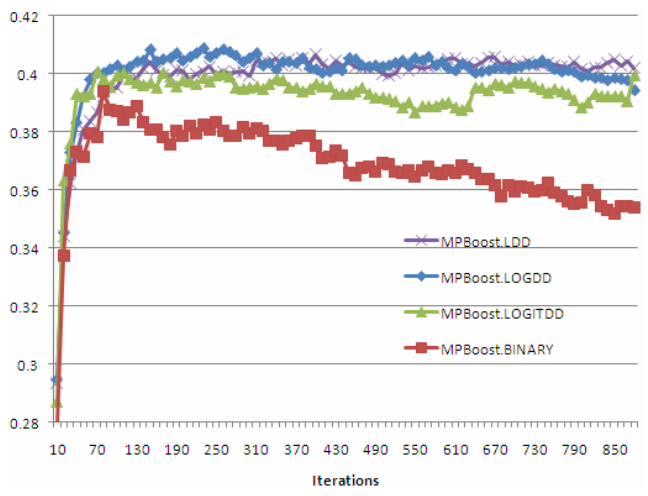

Figure 6. Average NDCG@5 on test set over 5 folds in Web-2 Dataset. For MPBoost.LDD, MPBoost.LOGDD and MPBoost.LOGITDD, the parameters α , λ and β are set as the one achieving the best performance on validation set.

Web-2 dataset was in average 0.062). The consequence is that the ability to differentiate smaller rating differences degrades. The logistic-based DDF, LOGITDD, has similar problems in that the overall output interval is $(-1,-0.5]\cup[0.5,1)$ and the saturation problem appears quite often. On the contrary, LOGDD generates moderate range of output while preserving the original magnitude properly.

5.5 Overfitting Issues

In the boosting method, overfitting is a thorny issue that needs to be handled [5]. Specifically, while the empirical error will keep decreasing in the training phase, the performance on test set may degrade as the number of training rounds increases. In our experiments, we observe that MPBoost with binary label suffers from overfitting, while MPBoost with magnitude-preserving properties perform well as training goes on.

To demonstrate, in the previous experiment on Web-2, we record the performance of the trained model on test set every 10 iterations. Figure 6 shows the average NDCG@5 on test set over five folders in the Web-2 dataset. As shown in the figure, MPBoost.BINARY suffers from serious overfitting problem after peaking at around the 90th iteration. However, MPBoost.LDD, MPBoost.LOGDD and MPBoost.LOGITDD can achieve comparatively stable ranking accuracies as the number of training rounds increase and avoid the overfitting issue to some extent.

We attribute this phenomenon to the incompleteness of traditional binary-label pairwise approach to ranking, combined with the distribution of different magnitude of ratings. On one hand, the calculation of metrics such as NDCG deals with the *magnitude of ratings*, not directly with the pairwise order. On the other hand, in the Web-2 dataset, the relevance ratings come from {0, 1, 2, 3, 4}, which give considerable impetus to place document with ratings like 3 and 4 in front. Thus, ignoring the magnitude of ratings and only retaining the relative order, binary labels lose a large amount of information during the transformation from ranking to classification. Thus, the trained model deviates from correctly capturing the way to improve metrics like NDCG, and test data substantiates the incompleteness.

On the contrary, the other three versions of magnitude-preserving MPBoost all apply loss functions with DDF. Thus, in the training phase, these three versions can learn a more suitable model to match the goal of ranking in the sense of improving NDCG. Hence, the performance on test set does not degrade after sufficient long time of training.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a new approach to magnitude-preserving ranking: directed distance function (DDF). Compared to previous schemes [4], DDF imposes less restriction on the form of functions while retaining the magnitude of rating differences. We also present three kinds of directed distance functions: LDD, LOGDD and LOGITDD, which can be applied under different circumstances due to the output range. The parameters in these DDFs can be easily adapted to meet requirements of different ranking algorithms.

Based on DDF, we propose a new boosting method for ranking problem: MPBoost. MPBoost incorporates directed distance function with the exponential loss function and applies GentleBoost-like optimization. The ranking loss, or misordering, of MPBoost is still bounded, like RankBoost, which is based on AdaBoost.

Experimental results with three datasets indicate that the MPBoost method, when combined with magnitude-preserving DDF, outperforms binary-label-based MPBoost and existing state-of-art approaches like RankBoost, ListNet and AdaRank-NDCG. Furthermore, MPBoost with DDF tend to avoid overfitting in training.

For future work, we plan to study the theoretical advantage in our method. We also intend to apply mixed directed distance for different pairs of ratings to more accurately depict the magnitude issue.

7. ACKNOWLEDGEMENT

The work of Chenguang Zhu was supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant 60604033, 60553001, and the Hi-Tech research and Development Program of China Grant 2006AA10Z216.We would also express our sincere acknowledgement to Xiaohui Wu and Teng Gao for their great help for our work.

8. REFERENCES

[1] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds,M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. Proceedings of ICML 2005, 89–96.

[2] Cao, Z., Qin, T., Liu, T., Y., Tsai, M.-F., and Li, H. (2007). Learning to Rank: From Pairwise Approach to Listwise Approach. Proceedings of ICML 2007, 129-136.

[3] Carvalho, V. R., Elsas, J. L., Cohen, W. W., and Carbonell, J. G. (2008). A Meta-Learning Approach for Robust Rank Learning. SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR 2008), 15-23.

[4] Cortes, C., Mohri, M., and Rastogi A. (2007). Magnitude-Preserving Ranking Algorithms. Proceedings of ICML 2007, 169-176.

[5] Ditterich, T. G. (1999). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning, 40 (2) (1999).

[6] Elsas, J., Carvalho, V. R., and Carbonell, J. G. (2008). Fast learning of document ranking functions with the committee perceptron. Proceedings of ACM International Conference on Web Search and Data Mining 2008.

[7] Freund, Y., Iyer, R., Schapire, R.E., and Singer Y.(1998). An Efficient Boosting Algorithm for Combining Preferences. Proceedings of ICML 1998.

[8] Freund, Y., Iyer, R., Schapire, R. E., and Singer Y.(2003). An Efficient Boosting Algorithm for Combining Preferences. Journal of Machine Learning Research 4 (2003) 933-969

[9] Freund Y. and Schapire, R. E. (1995). A decision-theoretic generalization of online learning and an application to boosting. In Computational Learning Theory: Eurocolt ’95, 23–37.

[10] Friedman, J., Hastie, T. and Tibshirani, R.(2000) Additive Logistic Regression: a Statistical View of Boosting. Annals of Statistics 2000, Vol. 28.

[11] Fürnkranz, J., and Hüllermeier, E. (2003). Pairwise Preference Learning and Ranking. Proceedings of ECML 2003 (pp. 145-156).

[12] He J. and Bo T. (2007). Asymmetric gradient boosting with application to spam filtering. Proceedings of Fourth Conference on Email and Anti-Spam CEAS, 2007.

[13] Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support vector learning for ordinal regression. Proceedings of ICANN 1999, 97–102.

[14] Hersh, W. R., Buckley, C., Leone, T. J., and Hickam, D. H. (1994). OHSUMED: An interactive retrieval evaluation and new large test collection for research. Proceedings of SIGIR 1994, 192–201.

[15] Internet Mathematics Contest 2009 training data (Learning to Rank) http://download.yandex.ru/imat2009/imat2009.tar.bz2 Accessed 20 May 2009

[16] Jarvelin, K., and Kekanainen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. Proceedings of SIGIR 2000, 41–48.

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

[18] Liu, T. Y., Qin, T., Xu, J., Xiong, W. Y., and Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. Proceedings of SIGIR 2007.

[19] Liu, T. Y., Zhang, R. C. (2008) Learning to Rank (LETOR) http://research.microsoft.com/en-us/um/beijing/projects/letor/index.html. Accessed 17 April 2009.

[20] Mason, L., Baxter, J., Bartlett, P. and Frean, M. (2000). Boosting algorithms as Gradient Descent. Proceedings of NIPS 12, 512–518.

[21] Qin, T., Liu, T. Y., Lai, W., Zhang, X. D., Wang, D. S., and Li, H. Ranking with Multiple Hyperplanes. Proceedings of

the 30th Annual International ACM SIGIR Conference, (2007), 279-286.

[22] Schapire, R. E., Freund Y., Bartlett, P. L., and Lee, W. S. (1998) Boosting the margin: A new explanation for the effectiveness of voting methods, The Annals of Statistics, J. Machine Learning Research, vol. 26(5), 1651-1686.

[23] Xu, J. and Li, H. AdaRank: A Boosting Algorithm for Information Retrieval. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, (2007), 391-398.

[24] Yang, P., Shan, S., Gao, W., Li, S. and Zhang, D. (2004) Face Recognition Using Ada-Boosted Gabor Features. IEEE Int. conf. On Automatic Face and Gesture Recognition (FG2004), 356-361.

9. APPENDIX - THE WEAK RANKER: DECISION STUMPS

The weak ranker based on decision stumps, $r_d(x)$, is defined as the following:

$$r_d(x) = \begin{cases} a & \mathrm{x}_k > \theta \\ 0 & \mathrm{x}_k \le \theta \end{cases} \tag{17}$$

k is the serial number of the feature $r_d(x)$ selects. θ is the threshold for this feature. a and 0 are the only two possible values $r_d(x)$ can take. We choose 0 here because in pairwise approach, only the difference of the two values $r_d(x)$ can take matters. In other words, we only need $r_d(x_i) - r_d(x_j)$ in calculation.

To find the best $r_d(x)$, we can iterate k. When k is determined, we only have to iterate $(n+1)$ values for θ : $-\infty, \mathrm{x}_{1,k}, \mathrm{x}_{2,k}, \ldots, \mathrm{x}_{n,k}$. Thus, the only problem is to find the best a when k and θ are determined. Notice that both $r_d(x_i)$ and $r_d(x_j)$ can possibly take two values, yielding 4 combinations. Therefore, we should split the sum into 4 cases.

Define:

$$\begin{aligned} A_1 &= \{(i,j) \in I \mid x_{i,k} > \theta, x_{j,k} \le \theta\} \\ A_2 &= \{(i,j) \in I \mid x_{i,k} > \theta, x_{j,k} > \theta\} \\ B_1 &= \{(i,j) \in I \mid x_{i,k} \le \theta, x_{j,k} \le \theta\} \\ B_2 &= \{(i,j) \in I \mid x_{i,k} \le \theta, x_{j,k} > \theta\} \end{aligned} \tag{18}$$

Suppose the best a for specific k and θ is $a_{k,\theta}$. We have:

$$\begin{aligned} J_{wse}(f_t) = &\sum_{A_1} w_{ij}^{(t)} (dist(r_i, r_j) - a_{k,\theta} + 0)^2 + \sum_{A_2} w_{ij}^{(t)} (dist(r_i, r_j) \\ &- a_{k,\theta} + a_{k,\theta})^2 \\ &+ \sum_{B_1} w_{ij}^{(t)} (dist(r_i, r_j) - 0 + 0)^2 + \sum_{B_2} w_{ij}^{(t)} (dist(r_i, r_j) \\ &- 0 + a_{k,\theta})^2 \\ &= \sum_{A_1} w_{ij}^{(t)} (dist(r_i, r_j) - a_{k,\theta})^2 + \sum_{A_2} w_{ij}^{(t)} dist(r_i, r_j)^2 \\ &+ \sum_{B_1} w_{ij}^{(t)} dist(r_i, r_j)^2 + \sum_{B_2} w_{ij}^{(t)} (dist(r_i, r_j) + a_{k,\theta})^2 \end{aligned} \tag{19}$$

Consequently, in order to minimize $J_{wse}(f_t)$, we take the partial derivative and obtain the best $a_{k,\theta}$:

$$a_{k,\theta} = \frac{\sum_{A_1} w_{ij}^{(t)} dist(r_i, r_j) - \sum_{B_2} w_{ij}^{(t)} dist(r_i, r_j)}{\sum_{A_1} w_{ij}^{(t)} + \sum_{B_2} w_{ij}^{(t)}}$$

Thus, after iterating through all possible k 's and θ 's, we can get the minimum $J_{wse}(f_t)$ and the corresponding best parameters. Then, $f_t(x) = \begin{cases} a_{k_t,\theta_t} & \mathrm{x}_{i,k_t} > \theta_t \\ 0 & \mathrm{x}_{i,k_t} \le \theta_t \end{cases}$.