

# Ranking with Multiple Hyperplanes

Tao Qin<sup>2\*</sup>, Tie-Yan Liu<sup>1</sup>, Wei Lai<sup>1</sup>, Xu-Dong Zhang<sup>2</sup>, De-Sheng Wang<sup>2</sup>, Hang Li<sup>1</sup>

<sup>1</sup>Microsoft Research Asia, No.49 Zhichun Road, Haidian District, Beijing 100080, P.R. China

<sup>2</sup>Dept. Electronic Engineering, Tsinghua University, Beijing, 100084, P.R. China

<sup>1</sup>{tyliu, weilai, hangli}@microsoft.com

<sup>2</sup>tsintao@gmail.com, {zhangxd, wangdsh\_ee}@mail.thu.edu.cn

## ABSTRACT

The central problem for many applications in Information Retrieval is ranking and learning to rank is considered as a promising approach for addressing the issue. Ranking SVM, for example, is a state-of-the-art method for learning to rank and has been empirically demonstrated to be effective. In this paper, we study the issue of learning to rank, particularly the approach of using SVM techniques to perform the task. We point out that although Ranking SVM is advantageous, it still has shortcomings. Ranking SVM employs a single hyperplane in the feature space as the model for ranking, which is too simple to tackle complex ranking problems. Furthermore, the training of Ranking SVM is also computationally costly. In this paper, we look at an alternative approach to Ranking SVM, which we call “Multiple Hyperplane Ranker” (MHR), and make comparisons between the two approaches. MHR takes the divide-and-conquer strategy. It employs multiple hyperplanes to rank instances and finally aggregates the rankings by the hyperplanes. MHR contains Ranking SVM as a special case, and MHR can overcome the shortcomings which Ranking SVM has. Experimental results on two information retrieval datasets show that MHR can outperform Ranking SVM in ranking.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Retrieval Models

## General Terms

Algorithms, Performance, Theory

## Keywords

Ranking with Multiple Hyperplanes, Ranking SVM

## 1. INTRODUCTION

Ranking is the central problem for many IR applications. These include document retrieval [4], collaborative filtering [15], key term extraction [8], expert finding [1], important email routing [5], sentiment analysis [27], product rating [9], and anti web spam [14]. In the task, given a set of instances, we make use of a ranking model (function) to calculate the score of each object and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

\*This work was done at Microsoft Research Asia.

SIGIR '07, July 23–27, 2007, Amsterdam, Holland.

Copyright 2007 ACM 1-58113-000-0/00/0004...\$5.00.

sort the objects with the scores. The scores may represent the degrees of relevance, preference, or importance, depending on applications.

Learning to rank is aimed at automatically creating the ranking model using some training data and machine learning techniques. A typical setting in learning to rank, which is also the case in this paper, is that feature vectors and ranks (ordered categories) are given as training data.

Because of the importance, learning to rank has been gaining increasing attention in IR and related fields. Many methods of learning to rank have been proposed (e.g., [3][13][18][23][25][26]) and applied to IR applications (e.g., [4][1]). Among them, Ranking SVM (RSVM) [18] is a typical method, which performs the learning task by utilizing the SVM techniques, one of the most powerful tools in machine learning. In this paper, we also focus on learning to rank based on the SVM approach. (The idea, insights, and discussions we have in this paper should be extensible to other approaches.)

In RSVM, ranking is performed by means of classification on instance pairs. More precisely, in training a set of instance pairs is created; each pair consists of two instances from two different ranks. Therefore, for each instance pair, there is an order between its two instances. A classification model is constructed for identifying the order relationship between the two instances in any instance pair. Ranking can be then conducted on the basis of the classification model. (More detailed descriptions on RSVM will be given in Section 2.)

In this paper, we study an alternative approach to RSVM, because the SVM approach to learning to rank is important and needs more investigations, and because RSVM has both pros and cons and needs further improvements.

RSVM is unique in that it constructs one single hyperplane classifier on instance pairs and makes use of it for ranking. One advantage of it is the simplicity of the model. However, this also causes a problem. The single classifier is built and utilized for ranking instances from all ranks. In reality, the ranking of order relationships between instances from different ranks are hard to be handled with a single model. Another problem with RSVM is that the training of it is in general costly. This is because it uses instance pairs to train the model, which is of quadratic order of the training instance size.

In this paper, we try to look at the other side of the spectrum. Specifically we take the divide-and-conquer strategy and work out a new method for learning to rank using SVM techniques. We try to make thorough comparisons between the two methods. The

relation between RSVM and our method is similar to that between Multi-class SVM [19][30] and ECOC [10][24] for multi-class classification.

Our method, referred to as “Multiple Hyperplane Ranker” (MHR), employs several hyperplane classifiers and aggregates the results of them for final ranking. MHR contains two major components: base ranker and rank aggregation. Each base ranker is a hyperplane for identifying ordering relationships between instances from two ranks. In learning, a base ranker is created for any two ranks to ensure high accuracies on the local ranking. Rank aggregation is conducted by using an ensemble model of the base rankers. In learning, the ensemble model is created, to ensure high accuracy in the global ranking.

It is easy to see that MHR contains RSVM as its special case, and is complementary to RSVM. The advantages of MHR include higher accuracy in ranking, efficiency in training, and ease of incorporating prior knowledge. The disadvantage of it is a slight increase in model complexity. Experimental results on two information retrieval data sets show that MHR works better than RSVM.

The contribution of this paper is three folds: (1) a proposal of a new learning to rank method MHR, (2) presentation of a general picture on the SVM approach to learning to rank, and (3) thorough comparisons between RSVM and MHR, and empirical verification of the effectiveness of MHR.

The rest of this paper is organized as follow. Section 2 introduces Ranking SVM. Section 3 describes MHR. Experimental results are reported in Section 4. Section 5 introduces related work. Conclusion and future work are given in Section 6.

## 2. RANKING SVM

### 2.1 Model

Suppose that  $X \in R^d$  is the feature space, in which  $d$  is the number of features, and  $Y = \{r_1, r_2, \dots, r_K\}$  is the set of labels representing ranks. Further assume that there exists a total order between ranks  $r_1 > r_2 > \dots > r_K$ , where  $>$  denotes the order relationship.

In training a set of labeled instances is given  $Z = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ , where  $x_i \in X$  is the feature vector of instance  $i$ , and  $y_i \in Y$  is the label of instance  $i$ . If  $y_i > y_j$ , we say that  $x_i$  is ranked ahead of  $x_j$ , denoted as  $x_i \succ x_j$ . Assume that  $F$  is the set of ranking functions, such that each member of it  $f \in F$  can rank instances:

$$x_i \succ x_j \Leftrightarrow f(x_i) > f(x_j) \quad (1)$$

In Ranking SVM (RSVM),  $f$  is assumed to be a linear function [18],

$$f(x) = \langle \omega, x \rangle \quad (2)$$

where  $\omega$  is weight vector and  $\langle \cdot, \cdot \rangle$  denotes inner product.  $f(x) = 0$  corresponds to a hyperplane in the feature space. Thus we have

$$x_i \succ x_j \Leftrightarrow \langle \omega, x_i - x_j \rangle > 0 \quad (3)$$

Given an instance pair  $x_i$  and  $x_j$ , we create a new instance  $x_i - x_j$ . If  $x_i$  is ranked ahead of  $x_j$ , we assign a label +1, otherwise -1 to the new instance. In this way, we produce a new data set and we can build a binary classification model, for example, a linear SVM

(hyperplane) using the data set. This is exactly the RSVM model. Since (1) and (3) hold, the RSVM model (2) can be directly used for ranking instances.

The actual task of learning is formalized as a Quadratic Programming problem as shown below:

$$\begin{aligned} \min_{\omega, \xi_{ij}} \quad & \frac{1}{2} \|\omega\|^2 + C \sum \xi_{ij} \\ \text{s. t.} \quad & \langle \omega, x_i - x_j \rangle > 1 - \xi_{ij}, \forall x_i \succ x_j, \xi_{ij} \geq 0 \end{aligned} \quad (4)$$

where  $\|\omega\|^2$  denotes  $\ell_2$  norm measuring the margin of the hyperplane and  $\xi_{ij}$  denotes a slack variable. Suppose that the solution to (4) is  $\omega^*$ , then we can have the ranking function as,

$$f(x) = \langle \omega^*, x \rangle.$$

### 2.2 Problems

The advantages of RSVM include the simplicity of its model and the effectiveness of its uses. Theoretically, the ranking accuracy of RSVM in terms of Average Precision is known to be approximately bounded from below by the inverse of classification errors on instance pairs [23]. Empirically, the effectiveness of it has been demonstrated.

However, RSVM also has certain disadvantages. First, the model might be too simple for tackling complex ranking problems in practice, as will be seen below. Second, the training process of RSVM is time consuming. The problem becomes severe, when the size of training data gets large and when the number of ranks increases. This is because RSVM makes use of instance pairs as training data (which is of quadratic order of training data size). Third, it is not easy to incorporate prior knowledge into the model. In many IR applications, it is important to focus on the training on the tops of rankings, as indicated in [25]. One solution to the problem is to add different weights on the ordering (classification) decisions between instances from different ranks. The original RSVM cannot cope with the problem, however.

Next, we use a real example to show why employing a single hyperplane for ranking can be problematic. We have made analyses on the OHSUMED document retrieval data, by means of Principal Component Analysis (PCA). There are three ranks in the data “definitely relevant”, “partially relevant”, and “irrelevant”. For each query, there are a certain number of associated documents. An instance (feature vector) can be created by using one query and one associated document. We have plotted the instances of different queries in the space described by the two principal coordinates. It seems to be always true that single hyperplanes cannot separate the instance pairs in different ranks very well.

As example, Figure 1 shows the distribution of instances for query 5 in the OHSUMED data set, in the space described by the two principal coordinates. Red crosses denote “definitely relevant” documents, green rectangles “partially relevant” documents, and blue triangles “irrelevant” documents. We can see that exploiting a single hyperplane cannot rank the documents properly. When we create hyperplanes for identifying order relationships between instances from any two of the ranks, we see that the normal directions of the hyperplanes are very different: direction 1 is that between “definitely relevant” and “irrelevant”; direction 2 that between “partially relevant” and “irrelevant”; and direction 3 that between “definitely relevant” and “partially relevant”.

One may argue that the examples shown are those after the processing of PCA. In the original space, the data might be more separable. It seems that this is not the case, as will be seen in our experimental results in Section 4.

One may also point out that the distributions of data also depend on the features used and when more features become available the problem may disappear. It is difficult to anticipate what will happen when more features are available. In our current data, we use most of the conventional features in document retrieval. Therefore, at least with the standard feature set, we observe the phenomena explained above.

A question may also be raised on whether the tendency still exists when kernels are used. We think, however, that the overall trend should not change in the case. This is because as a general principle, dividing the whole problem into sub-problems will always lead to better solutions to the sub-problems, and further effectively combining the base solutions will lead to a better total solution.

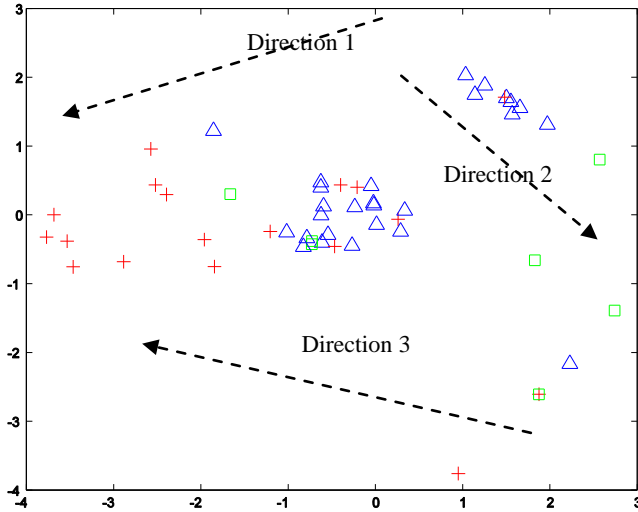


Figure 1. Distribution of instances in OSHUMED data (query 5)

### 3. MULTIPLE HYPERPLANE RANKER

#### 3.1 Model

We propose an alternative to Ranking SVM for learning to rank, referred to as Multiple Hyperplane Ranker (MHR). MHR exploits multiple hyperplanes as base rankers and aggregates the rankings of the base rankers. Base rankers are prepared for all the rank pairs. Each base ranker is simply a hyperplane trained with ranking SVM for ranking instances from one rank pair. Rank aggregation is performed by using an ensemble of the base rankers. It is easy to verify that MHR will be equivalent to RSVM, if we set the normal directions of all the hyperplanes to be the same. That is, MHR contains RSVM as a special case.

In learning, we create base rankers by using the training data. We first partition the training set into several subsets by putting all the instance pairs from the same rank pairs into the same subsets. For clarity, we use rank pair  $(s,t)$  to denote the subset of instance pairs consisting of a document with rank  $s$  and a document with rank  $t$ . Using the instance pairs in each subset, we

can construct one base ranker for the corresponding rank pair. The base rankers can be trained separately or in parallel. Obviously, the number of instance pairs for each base ranker is smaller than that of the entire problem. Therefore, the space complexity and time complexity of the training in base ranker construction are lower than those of RSVM training on the entire set of rank pairs. Furthermore, each base ranker focuses on the rankings regarding to one rank pair, and thus the accuracy should be higher than that of RSVM.

In ranking, an ensemble of base rankers is utilized. We can employ an existing method to create the ensemble, in either a supervised or an unsupervised fashion. If each of the base rankers can conduct accurate ranking on a subset of the data and the base rankers do not have strong correlation, then an appropriate aggregation method will lead to a higher final ranking accuracy. We can also incorporate prior knowledge into the rank aggregation process. For example, we can give higher weights to the base rankers which we think are important.

#### 3.2 Base Ranker

Let  $\omega_{s,t}$  denote that the parameter vector of the base ranker (RSVM) for the rank pair  $s$  and  $t$ . Then, we can build up a base ranker:

$$\begin{aligned} \min_{\omega_{s,t}, \xi_{i,j,s,t}} \frac{1}{2} \|\omega_{s,t}\|^2 + C \sum_{i,j} \xi_{i,j,s,t} \\ s. t. \quad \langle \omega_{s,t}, x_i^{(s)} - x_j^{(t)} \rangle \geq 1 - \xi_{i,j,s,t} \\ \xi_{i,j,s,t} \geq 0 \end{aligned}$$

where  $x_i^{(s)}$  indicate an instance  $x_i$  with rank  $s$ .

If there are  $K$  ranks, then there are  $K(K-1)/2$  base rankers for  $K(K-1)/2$  rank pairs. For example if  $K$  is 4, then the base rankers are hyperplanes represented by  $\{\omega_{1,2}, \omega_{1,3}, \omega_{1,4}, \omega_{2,3}, \omega_{2,4}, \omega_{3,4}\}$ , corresponding to rank pairs  $\{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$ .

We note that there are other ways for creating base rankers. For example, instead of creating base rankers for all the rank pairs, we can create base rankers only for the adjacent rank pairs. We can also conduct sampling on the data and use the sampled data in base ranker creation.

#### 3.3 Rank aggregation

We employ two rank aggregation methods for creating an ensemble of base rankers: BordaCount [2][12] and Weighted BordaCount.

Let  $D$  denote the set of entities to be ranked and  $n$  denote the number of entities in  $D$ . Let  $\tau_1, \dots, \tau_l$  denote ranking lists on  $D$ , generated by the base rankers in MHR, and  $l$  is the number of base rankers.

BordaCount first assigns a score to each entity based on its positions in the ranking lists. For example, the score of an entity can be defined as the number of entities that are ranked lower than the entity in all the ranking lists. Formally, the score of an entity  $x$  is defined as

$$s(x) = \sum_{k=1}^l s_k(x)$$

where  $s_k(x) = \#\{y | x >_{\tau_k} y, y \in D\}$ , and  $x >_{\tau_k} y$  means that entity  $x$  is ranked higher than entity  $y$  in ranking list  $\tau_k$ . Finally,

BordaCount sorts all the entities according to their scores. One advantage of BordaCount is that it can be run in linear time.

Weighted BordaCount assigns weights to the base rankers:

$$s(x) = \sum_{k=1}^l \alpha_k s_k(x)$$

where  $\alpha_k$  denotes weight. There are several ways to determine the weights. For example, one can use a separate validation set to tune the weights. It is also possible to manually set the weights reflecting human’s prior knowledge.

Note that there are other ways to create the ensemble. For example, Median Rank Aggregation [12] and Markov Chain based Rank Aggregation [11]. Median Rank Aggregation sorts all the entities by the medians of their positions in different ranking lists. Markov Chain based Rank Aggregation defines a Markov Chain model in which entities correspond to states and order relationships between entities correspond to transitions between states. It utilizes the stationary distribution of the Markov Chain to rank the entities.

## 4. EXPERIMENTAL RESULTS

We applied our method MHR to information retrieval. Particularly we compared the performances of MHR with those of RSVM. When conducting the experiments, we performed 4-fold cross-validation. Thus, all the results reported in this section are those averaged over four trials.

### 4.1 Data Sets

We used two information retrieval datasets: one is a public data set named OHSUMED [17][18], and the other is a dataset for definition search, obtained from the authors of [1].

#### 4.1.1 OHSUMED Dataset

OHSUMED is a collection of documents and queries on medicine, consisting of 348,566 references and 106 queries. There are 16,140 query-document pairs in total upon which relevance judgments are made. In this corpus the relevance judgments have three levels: “definitely relevant”, “partially relevant”, and “irrelevant”. For simplicity, we will use rank 1 to denote “definitely relevant”, rank 2 to denote “partially relevant”, and rank 3 to denote “irrelevant”. We extracted 30 features similar to those used in [4][26] from the query-document pairs. Examples of the features are shown in Table 1, including term frequency ( $tf$ ), inverse document frequency ( $idf$ ), document length ( $dl$ ), their combinations, and BM25 score [28].

**Table 1. Feature extracted from the OHSUMED dataset**

$c(w, d)$  represents frequency of query word  $w$  in document  $d$ ;  $C$  represents the entire collection;  $n$  denotes number of words in query;  $|\cdot|$  denotes size of function; and  $idf(\cdot)$  denotes inverse document frequency.

Features	
1 $\sum_{q_i \in q \cap d} \log(c(q_i, d) + 1)$	2 $\sum_{q_i \in q \cap d} \log\left(\frac{ c }{c(q_i, C)} + 1\right)$
3 $\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{ d } idf(q_i) + 1\right)$	4 $\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{ d } + 1\right)$
5 $\sum_{q_i \in q \cap d} \log\left(\frac{c(q_i, d)}{ d } \frac{ c }{c(q_i, C)} + 1\right)$	6 $\sum_{q_i \in q \cap d} \log(idf(q_i))$
7 $\log(\text{BM25 score})$	

#### 4.1.2 Definition Search Dataset

This data set comes from definition search [1], in which given a query (usually a noun phrase representing a terminology), the

system returns a ranked list of paragraphs which are considered as definitions of the term, in descending order of the degree of goodness as definition. This dataset provides three levels of judgment: “good definition” (rank 1), “indifferent definition” (rank 2) and “bad definition” (rank 3). There are about 170 queries and more than 2,000 definition candidates for each query. Features are the same as those in [1]. Table 2 gives example features.

**Table 2. Features of definition search dataset**

1. <query> occurs at beginning of paragraph.
2. <query> begins with ‘the’, ‘a’, or ‘an’.
3. All the words in <query> begin with uppercase letters.
4. Paragraph contains predefined negative words, e.g. ‘he’, ‘said’, ‘she’
5. <query> contains pronouns.
6. <query> contains ‘and’, ‘or’, ‘of’, ‘for’, or ‘,’.
7. <query> re-occurs in the paragraph.
8. <query> is followed by ‘is an’, ‘is a’ or ‘is the’.
9. Number of sentences in paragraph.
10. Number of words in paragraph.
11. Number of the adjectives in paragraph.
12. Bag of words: words frequently occurring within a window after <query>

### 4.2 Evaluation Criteria

We used two evaluation criteria in our experiments for ranking accuracy evaluations: Mean Average Precision (MAP) [1] and Normalized Discount Cumulative Gain (NDCG) [20][21]. They are measures widely used in IR.

#### 4.2.1 MAP

MAP is a measure on precision of ranking results. It is assumed that there are two ranks: positive and negative. Precision at  $n$  measures accuracy of top  $n$  results for a query.

$$P(n) = \frac{\text{number of relevant instances in top } n}{n}$$

Average precision of a query is calculated based on precision at  $n$ :

$$AP = \sum_{n=1}^N \frac{P@n \times pos(n)}{\text{number of positive instances}}$$

where  $n$  is position,  $N$  is number of instances retrieved,  $pos(n)$  is a binary function indicating whether the instance at position  $n$  is positive (e.g., relevant). MAP is defined as AP averaged over all queries. In our experiments, the data sets have three ranks. We define the first rank as *positive* and the other two as *negative* when calculating MAP.

#### 4.2.2 NDCG

NDCG is designed for measuring ranking accuracies when there are more than two ranks. Given a query, NDCG at position  $n$  is defined as

$$N_n = Z_n \sum_{j=1}^n \frac{2^{R(j)} - 1}{\log(1+j)}$$

where  $n$  is position,  $R(j)$  is score for rank  $j$ , and  $Z_n$  is a normalization factor to guarantee that a perfect ranking’s NDCG

at position  $m$  is 1. For queries for which the number of retrieved instances is less than  $n$ , NDCG is only calculated for the retrieved instances.

### 4.3 Comparison between MHR and RSVM

We conducted experiments on document retrieval and definition search using MHR and RSVM with the two data sets. We denote the options of using BordaCount and Weighted BordaCount as MHR-BC and MHR-WBC respectively.

In the experiments, as SVM tool we used SVM Light [22]. The default parameter setting in the tool was used. In MHR-WBC, we tuned the weights in an ensemble by using the entire training dataset (note that base rankers should be trained with the subsets of data, while an ensemble should be trained with the entire data set).

Figure 2 shows the ranking accuracies of RSVM and MHR on the OHSUMED data set in terms of both NDCG and MAP.

For most NDCG values, MHR-BC outperforms RSVM with more than 2% relative improvement. For NDCG@1, the relative improvement is as large as 7%. MHR-WBC shows even better performance, for which NDCG@1 is relatively 12% higher than that of RSVM. For MAP, we obtain similar improvements. These results indicate that MHR can outperform RSVM. Moreover, using Weighted BordaCount is better than using BordaCount. Table 3 shows the relative improvements of MHR over RSVM (note that the results are averaged over four trials in cross validation).

The average NDCG@1, NDCG@10 and MAP scores of BM25 [28] over the 4 trials of OHSUMED data are 0.538, 0.518, and 0.243 separately. It is easy to find that both RSVM and MHR are much better than BM25.

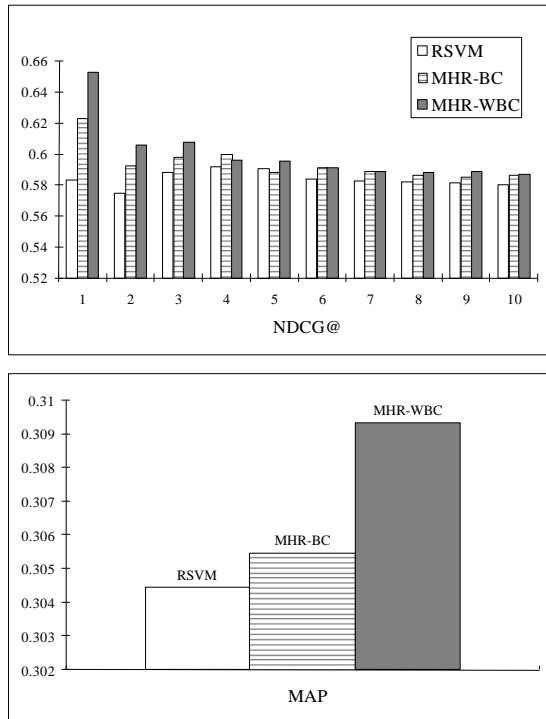


Figure 2. Results for the OHSUMED dataset

Figure 3 shows the experimental results on the definition search dataset. Again, MHR improves upon RSVM in ranking accuracy in terms of all measures. Particularly, MHR-WBC achieves about 15% relative improvement over RSVM in terms of MAP. Table 4 shows the relative improvements of MHR over RSVM.

Table 3. Relative improvements of MHR over RSVM on OHSUMED dataset

	NDCG@1	NDCG@10	MAP
MHR-BC	6.78%	0.92%	0.34%
MHR-WBC	11.96%	1.06%	1.60%

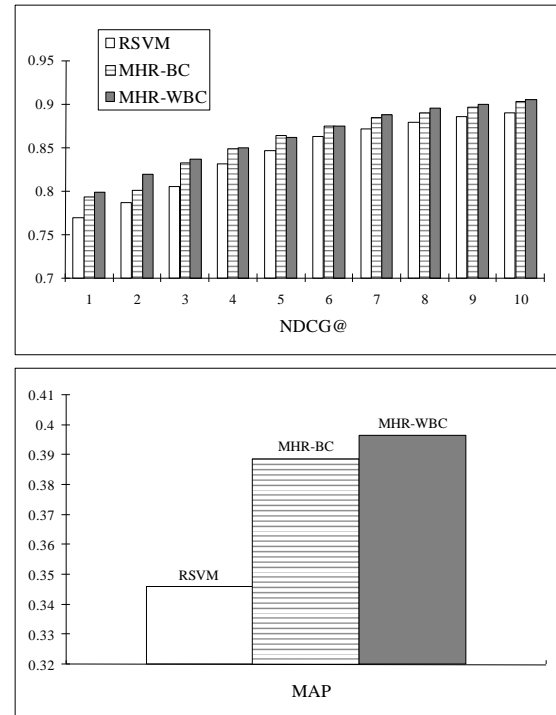


Figure 3. Results for the definition search dataset

Table 4. Relative improvements of MHR over RSVM on definition search dataset

	NDCG@1	NDCG@10	MAP
MHR-BC	3.07%	1.43%	12.38%
MHR-WBC	3.86%	1.70%	14.64%

## 4.4 Discussions

### 4.4.1 Directions of Hyperplanes

In Section 2, we have shown, with an example, that the normal directions of hyperplanes (base rankers) for different rank pairs can point to completely different directions. This can be verified by looking at the cosine similarities between the hyperplanes. We use  $\omega$  to denote the hyperplane of RSVM, and  $\omega_{1,2}, \omega_{1,3}, \omega_{2,3}$  the hyperplanes of MHR for three rank pairs. We define the cosine similarity between two hyperplanes  $\omega_i, \omega_j$  as

$$\cos(\omega_i, \omega_j) = \frac{\langle \omega_i, \omega_j \rangle}{\|\omega_i\| \|\omega_j\|}$$

Table 5 shows cosine similarities between any two hyperplanes in RSVM and MHR.

**Table 5. Cosine similarity between hyperplanes**

(a) Definition search data

trial 1					trial 2				
	$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$		$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$
$\omega$	1.00	0.26	0.56	0.92	$\omega$	1.00	0.29	0.55	0.91
$\omega_{1,2}$		1.00	0.51	0.04	$\omega_{1,2}$		1.00	0.64	0.03
$\omega_{1,3}$			1.00	0.39	$\omega_{1,3}$			1.00	0.32
$\omega_{2,3}$				1.00	$\omega_{2,3}$				1.00

trial 3					trial 4				
	$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$		$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$
$\omega$	1.00	0.35	0.60	0.88	$\omega$	1.00	0.34	0.56	0.90
$\omega_{1,2}$		1.00	0.66	0.05	$\omega_{1,2}$		1.00	0.68	0.07
$\omega_{1,3}$			1.00	0.33	$\omega_{1,3}$			1.00	0.33
$\omega_{2,3}$				1.00	$\omega_{2,3}$				1.00

(b) OHSUMED data

trial 1					trial 2				
	$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$		$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$
$\omega$	1.00	0.61	0.94	0.71	$\omega$	1.00	0.58	0.92	0.71
$\omega_{1,2}$		1.00	0.71	0.39	$\omega_{1,2}$		1.00	0.62	0.38
$\omega_{1,3}$			1.00	0.54	$\omega_{1,3}$			1.00	0.47
$\omega_{2,3}$				1.00	$\omega_{2,3}$				1.00

trial 3					trial 4				
	$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$		$\omega$	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$
$\omega$	1.00	0.32	0.79	0.62	$\omega$	1.00	0.62	0.94	0.34
$\omega_{1,2}$		1.00	0.55	0.08	$\omega_{1,2}$		1.00	0.64	0.33
$\omega_{1,3}$			1.00	0.20	$\omega_{1,3}$			1.00	0.12
$\omega_{2,3}$				1.00	$\omega_{2,3}$				1.00

From Table 5, we can see the following trends. 1) The cosine similarities for the same hyperplane pairs do not change largely across trials, indicating that the normal directions of hyperplanes are stable. 2) The normal directions of the four hyperplanes point away from each other. An extreme case is that hyperplanes  $\omega_{1,2}$  and  $\omega_{2,3}$  are almost perpendicular on ‘definition search’ data (i.e. their cosine similarity is almost zero). This implies that using a single hyperplane to rank instances from different ranks may be inappropriate.

#### 4.4.2 Order Error Rate

To further examine the reason that MHR can achieve higher ranking accuracies than RSVM, we studied the Order Error Rates of RSVM and MHR on training set. Order Error Rate (OER) is defined as follows.

$$OER = \frac{|\text{mistakenly ordered instance pairs}|}{|\text{all instance pairs}|}$$

Table 6 lists OER for RSVM and MHR. While calculating the OER of MHR, we used hyperplane  $\omega_{s,t}$  to rank all the instance pairs in rank pair  $(s,t)$ . That is, OER of RSVM is OER of a single hyperplane on three subsets of instance pairs, and OER of MHR is OER of three hyperplanes on three subsets of instance pairs separately. The last two rows are the average over four trials.

As can be seen from the table, RSVM make more errors in ranking instance pairs than MHR. Especially for rank pair (1,2) of definition search data, OER of RSVM is as high as 15% to 20%. In contrast, OER of MHR is only about one third of that of RSVM. This is consistent with the results shown in Table 5(a): the normal directions of hyperplanes  $\omega$  and  $\omega_{1,2}$  differ largely. These results indicate that to address complex ranking problems like those in our experiments, the divide-and-conquer strategy is really needed.

**Table 6. Order error rate on training set**

(a) Definition search data

		rank pair (1,2)	rank pair (1,3)	rank pair (2,3)
trial 1	RSVM	0.2005	0.0413	0.1049
	MHR	0.0735	0.0159	0.0935
trial 2	RSVM	0.1778	0.0418	0.1285
	MHR	0.0558	0.0182	0.1138
trial 3	RSVM	0.1662	0.0508	0.1567
	MHR	0.0457	0.0180	0.1257
trial 4	RSVM	0.1515	0.0354	0.1123
	MHR	0.0432	0.0152	0.1008
average	RSVM	0.1740	0.0423	0.1256
	MHR	0.0546	0.0168	0.1085

(b) OHSUMED data

		rank pair (1,2)	rank pair (1,3)	rank pair (2,3)
trial 1	RSVM	0.3939	0.3032	0.3902
	MHR	0.3629	0.2977	0.3833
trial 2	RSVM	0.3975	0.3118	0.3943
	MHR	0.3702	0.3085	0.3833
trial 3	RSVM	0.3819	0.2981	0.3834
	MHR	0.3527	0.2941	0.3743
trial 4	RSVM	0.3964	0.3250	0.4096
	MHR	0.3626	0.3207	0.4017
average	RSVM	0.3924	0.3095	0.3944
	MHR	0.3621	0.3052	0.3856

#### 4.4.3 Training Time

As aforementioned, in training MHR partitions data into subsets and uses each of the subsets to create one base ranker (hyperplane). In this way, it can reduce the computation time in training. We made a comparison on training time between MHR and RSVM. The experiments were conducted using SVM Light and on a PC with 2.2GHz CPU and 2G memory. Table 7 shows the results.

From Table 7, we can see that in general MHR is efficient in training when compared with RSVM. This is particularly true when the data size becomes larger. In the definition search dataset,

there are about 10,000 instance pairs in each trial, and the total training time of MHR is about two-thirds of that of RSVM. In the OHSUMED dataset, there are about 500,000 instance pairs in each trial, and the total training time of MHR is only one-third of that of RSVM.

In summary, the experimental results in this section validate the correctness of our claim: MHR can not only improve the rank accuracy but also training efficiency with its divide-and-conquer strategy. Therefore MHR is a reasonably good choice for practical ranking problems.

**Table 7. Training time**

(a) Definition search data

Seconds	MHR				RSVM
	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$	Sum	
trial 1	0.07	0.06	1.17	1.30	1.90
trial 2	0.11	0.07	2.78	2.96	3.10
trial 3	0.13	0.07	1.41	1.61	2.83
trial 4	0.08	0.07	1.67	1.82	3.76

(b) OHSUMED data

Minutes	MHR				RSVM
	$\omega_{1,2}$	$\omega_{1,3}$	$\omega_{2,3}$	Sum	
trial 1	17	90	175	282	823
trial 2	17	75	200	292	841
trial 3	16	78	154	248	663
trial 4	22	92	196	310	887

#### 4.4.4 Limitations

Although experimental results show that MHR can outperform Ranking SVM in some cases, it also has limitations. First, MHR requires training data with multi-level relevance judgments, while Ranking SVM only requires training data with pairwise preference [23]. In other words, MHR is not as general as Ranking SVM. Second, since the model of MHR is more complex than that of Ranking SVM, the overfitting probability of MHR may be higher than that of Ranking SVM. As a result, to get reliable and robust models, MHR may require more training data.

## 5. RELATED WORK

Learning to rank is to automatically create a ranking function which assigns scores to instances and then rank the instances by using the scores. Many methods have been proposed to address the issue. One major approach to learning to rank is that of transforming it into binary classification on instance pairs. Methods of the approach include Ranking SVM [4][18][23] [1], RankBoost [13], and RankNet [3][25]. (For other approaches, see [6][7][16][29].) Our method of MHR also falls into this category. There are several existing works [13][25] which are closely related to our current work.

RankBoost [13] is an algorithm to automatically create multiple “weak” rankers and linearly combine them to obtain a “strong” ranker, within the framework of Boosting. In RankBoost, all the instance pairs are used in training. In MHR, instance pairs are separated into groups and used. In that regard, MHR is more efficient than RankBoost. There are also similarities between RankBoost and MHR. In RankBoost, each weak ranker is created so that it is complementary to the previous weak rankers. In MHR, each base ranker is a hyperplane constructed with a subset of

training data and thus is also complimentary to the other base rankers.

Multiple Nested Ranker (MNR) [25] is a ranking method which manages to improve the accuracies at the tops of rank lists by iteratively re-ranking the top ranked results. The major differences between MNR and MHR are as follows. (1) In MNR, the nested rankers are trained sequentially; there is a dependency of the current ranker on its previous rankers. As a result, the training of MNR cannot be conducted in parallel, while this is not the case for MHR. (2) In MNR the size of training dataset for rankers shrinks step by step. In contrast, in MHR the training data set is partitioned into subsets and is used separately. As a result, an instance pair is only used once in training of MHR, while it might be used multiple times in training of MNR.

Ranking SVM for IR (RSVM-IR) is a method proposed in [4]. It creates a Ranking SVM model for document retrieval by modifying the loss function such that the model is trained with more considerations on the tops of ranking lists and queries with a small number of associated documents. RSVM-IR treats different rank pairs differently by changing the loss function. The model for final ranking is a unified model. In contrast, MNR treats different rank pairs differently by introducing base rankers. The model for final ranking is an ensemble of base rankers.

## 6. CONCLUSIONS

In this paper, we have proposed a new method called “Multiple Hyperplane Ranker” (MHR) for learning to rank. MHR takes the divide-and-conquer strategy to train multiple hyperplanes to rank instances and finally aggregates the rankings by the hyperplanes. MHR contains Ranking SVM as a special case, and is more flexible, effective, and efficient than Ranking SVM. Experimental results show that MHR can outperform Ranking SVM in ranking on two information retrieval datasets.

As future work, we plan to address the following issues.

- 1) Base ranker construction is an important component of MHR. In this paper, we have only investigated one option for it, namely, creating one base ranker for each rank pair. One can choose other options as well, for example, random sampling and overlapping partition. We will conduct a comprehensive study on the issue.
- 2) Ranking aggregation is another important component of MHR. In this paper, we have examined the effectiveness of using BordaCount and Weighted BordaCount. There are many other rank aggregation methods proposed, and we will further investigate whether they can work for MHR.
- 3) We will conduct more experiments with large scale datasets, to further verify the advantages of MHR.
- 4) We will investigate on the generalization ability of MHR.

## 7. REFERENCES

- [1] Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval, Addison Wesley, 1999.
- [2] Borda, J. C. Mémoire sur les élections au scrutin. *Histoire de l’Académie Royale des Sciences*, 1781.
- [3] Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G. Learning to Rank using Gradient Descent, 22nd International Conference on Machine Learning, Bonn, 2005.

- [4] Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W. Adapting ranking SVM to document retrieval. In SIGIR 2006: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, USA.
- [5] Chirita, P.A., Diederich, J., and Nejdl, W. MailRank: using ranking for spam detection, Proceedings of the 14th ACM international conference on Information and knowledge management, 2005.
- [6] Chu, W., and S. Sathya Keerthi, New approaches to support vector ordinal regression. Proceedings of the 22nd international conference on Machine learning, p.145-152, August 07-11, 2005, Bonn, Germany.
- [7] Cohen, W.W., Schapire, R.E., and Singer, Y. Learning to order things, Proceedings of the 1997 conference on Advances in neural information processing systems 10, p.451-457, July 1998, Denver, Colorado, United States.
- [8] Collins, M. Ranking algorithms for named-entity extraction: boosting and the voted perceptron, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July 07-12, 2002, Philadelphia, Pennsylvania.
- [9] Dave, K., Lawrence, S., Pennock, D.M. Mining the peanut gallery: opinion extraction and semantic classification of product reviews, Proceedings of the 12th international conference on World Wide Web, May 20-24, 2003, Budapest, Hungary.
- [10] Dietterich, T.G. and Bakiri, G. Solving multiclass learning problems via error correcting output codes. Journal of artificial Intelligence Research, 2:263-286, January 1995.
- [11] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*. 2001, 613-622.
- [12] Fagin, R., Kumar, R., and Sivakumar, D. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on management of data*. San Diego, 2003, 301-312.
- [13] Freund, Y., Iyer, R., Schapire, R., & Singer, Y., An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 2003 (4).
- [14] Gyongyi, Z., Garcia-Molina, H., and Pedersen, J. Combating web spam with TrustRank. In Proceedings of the 30th VLDB Conference, Sept. 2004.
- [15] Harrington, E. F. Online Ranking/Collaborative filtering using the Perceptron Algorithm. Proceedings of the 20th International Conference on Machine Learning, pages 250--257, 2003.
- [16] Har-Peled, S., Roth, D., and Zimak, D. Constraint classification: A new approach to multiclass classification and ranking. In Proc. Advances in Neural Information Processing Systems (NIPS'02), 2002.
- [17] Hersh, W. R., Buckley, C., Leone, T. J., and Hickam, D. H.. OHSUMED: An interactive retrieval evaluation and new large test collection for research. Proceedings of the 17th Annual ACM SIGIR Conference, pages 192-201, 1994.
- [18] Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers, MIT Press, Pages: 115-132.
- [19] Hsu, C.W., and Lin, C.J. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks, 13(2):415-425, 2002.
- [20] Jarvelin, K., & Kekalainen, J. (2000). IR evaluation methods for retrieving highly relevant documents. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, p.41-48, July 24-28, 2000, Athens, Greece.
- [21] Jarvelin, K., & Kekalainen, J. Cumulated Gain-Based Evaluation of IR Techniques, ACM Transactions on Information Systems, 2002.
- [22] Joachims, T. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [23] Joachims, T. Optimizing Search Engines Using Clickthrough Data, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.
- [24] Kong, E.B., and Dietterich, T.G. Error-correcting output coding corrects bias and variance. In Proceedings of the Twelfth International Conference on Machine Learning, pages 313-321, 1995.
- [25] Matveeva, I., Burges, C., Burkard, T., Laucius, A., and Wong, L. High Accuracy Retrieval with Multiple Nested Ranker. In SIGIR 2006: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, USA.
- [26] Nallapati, R. Discriminative models for information retrieval. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, July 25-29, 2004, Sheffield, United Kingdom.
- [27] Pang, B., and Lee, L. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. ACL 2005
- [28] Robertson, S. E. Overview of the okapi projects, Journal of Documentation, Vol. 53, No. 1, 1997, pp. 3-7.
- [29] Shashua, A., and Levin, A. Taxonomy of Large Margin Principle Algorithm for Ordinal Regression Problems. Advances in Neural Information Processing Systems 15. Cambridge, MA: MIT Press, 2000.
- [30] Weston, J. and Watkins, C. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998.
- [31] Xu, J., Cao, Y.B., Li, H., Zhao, M. Ranking definitions with supervised learning methods, Special interest tracks and posters of the 14th international conference on World Wide Web, May 10-14, 2005, Chiba, Japan