BOOSTING RANK WITH PREDICTABLE TRAINING ERROR

Yongqing Wang[1], *Wenji Mao*[1], *Daniel Zeng*[1,2] *and Ning Bao*[3]

[1]Key Lab of Complex Systems and Intelligent Science, CAS, Beijing, China
[2]Department of Management Information Systems, The University of Arizona
[3]School of Technical Physics, Xidian University, Xi'An, China

ABSTRACT

Listwise approach is an important method to solve practical Web search problem in learning to rank. In this paper, we first analyze the practical Web search problem and construct the model to solve it. Then we propose an algorithm called DiffRank which can apply boosting technology to learning to rank in listwise. Through theoretical analysis, we prove that the upper bound of training error can be reduced in our proposed algorithm. The experimental results further verify our theoretical analysis and demonstrate that our approach can better perform in practical Web search than other state-of-the-art listwise algorithms.

***Keywords*—** learning to rank, boosting, DiffRank

1. INTRODUCTION

Ranking is a central part of many information retrieval problem, including document retrieval, Web search. Along with the increasing demand in Web search for ranking massive retrieved objects, learning to rank has intensively gained attention in recent years. Learning to rank aims at using machine learning technologies to get a ranking model which can appropriately describe ranking for groups of objects. Generally, in the learning phase, the ranking model is trained by a fixed number of features (which describe an object) and a large amount of training data (that contain lots of objects). Previous work in [1, 2] categories learning to rank into three groups, pointwise, pairwise and listwise approaches. Meanwhile, it is points out in [1] that existing pointwise and pairwise approaches, such as Ranking SVM [3] and RankNet [4], have their inherent deficiencies in consideration of query level information. Therefore listwise approach has been the main learning to rank approach.

Listwise approach has several distinct advantages. First, it takes the query level information instead of the pointwise and pairwise ranking; second, it naturally reflects the real Web search instances; third, it can be directly optimized through kinds of evaluation measures. Many methods have been proposed including ListNet [2] and ListMLE [5].

In this paper, we propose a new algorithm called DiffRank inspired by Adaboost in listwise. We try to overcome three major difficulties, i.e., 1) how to provide a new loss function which can achieve the boosting object from weak rank to strong rank; 2) the loss function can be found its upper bound of training error; 3) the upper bound could be continuously reduced through the boosting process. Then, we provide a simple theoretical analysis on our proposed algorithm and choose a standard data set, which can closely describe practical Web search, to verify our analysis.

The rest paper is organized as follows. Section 2 introduces the definition of problem. Section 3 presents our proposed algorithm and its analysis. Experimental results are provided in Section 4.

2. PROBLEM DEFINITION

We first illustrate a Web search instance using learning to rank and its training data set. Instead of traditional empirical methods of content-based (BM25) and link-based (PageRank), retrieving a set of documents and their ranks will only depend on a fixed number of primary document features, such as the referred number in other Websites, term frequency, title relevance degree etc. When a user types in keywords as the query to inquire related information, search engine will feedback the query and retrieve documents according to their major features. Therefore, we can abstract the process of the Web search and create the training data set.

Let $\mathbf{X}=\left\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},\cdots,\mathbf{x}^{(N_q)}\right\}$ be the document space, where $\mathbf{x}^{(i)}$ is the retrieved document set corresponding to the i-th query and $\mathbf{x}_j^{(i)}\in\mathbf{x}^{(i)}$ is the j-th document in the set $\mathbf{x}_i$. Let $\mathbf{Y}=\left\{\mathbf{y}^{(1)},\mathbf{y}^{(2)},\cdots,\mathbf{y}^{(N_q)}\right\}$ be the ground truth space, where $\mathbf{y}^{(i)}$ represents the true ranking order in the i-th query. As we have introduced above, each document has fixed number of features, that is, $\mathbf{x}_j^{(i)}$ is a feature vector which a dimension denotes to one of primary features. Thus, the training set S can be described as $S=(\mathbf{X},\mathbf{Y})=\left\{\left(\mathbf{x}^{(i)},\mathbf{y}^{(i)}\right)\right\}^{N_q}$.

We aim to establish a ranking model with the form

$$F(\mathbf{X})=sort\left(\sum_{m=1}^{M}\alpha^m\mathbf{h}^m\right)=sort\left(\mathbf{H}^{m-1}+\alpha^m\mathbf{h}^m\right),$$

where M is the dimension number, $\mathbf{h}^m$ is the weak ranking orders corresponding to m-th feature, α^m is the weight value for weak ranker $\mathbf{h}^m$, $\mathbf{H}^{m-1}$ is the ranking scores at (m-1-th feature and equals to $\sum_{k=1}^{m-1}\alpha^k\mathbf{h}^k$, and sort is the sort function in which the higher score can be arranged in a higher rank.

Then, we propose a new loss function. In learning to rank, especially listwise approach, optimization objects focus on placing the items which own higher query level relevance scores upon the top. The higher rank, the higher relevance score. So we define our proposed evaluation function at the m-th feature:

$$Eval_m = \sum_{N_q}\sum_{i \succ j} \Delta H_{ij}^{(q),m},$$

where $\Delta H_{ij}^{(q),m}$ is the ranking scores' difference value between rank i and j.

For optimizing ranking, we changed the evaluation function to loss function which is like the following form

$$\begin{aligned} Loss_m &= \sum_{N_q}\sum_{i \succ j} \exp\left(-\frac{1}{U_q}\left(\Delta H_{ij}^{(q),m}\right)\right) \\ &= \sum_{N_q}\sum_{i \succ j} \exp\left(-\frac{1}{U_q}\left(\Delta H_{ij}^{(q),m-1} + \alpha \Delta h_{ij}^{(q),m}\right)\right) \end{aligned},$$

where U_q is the normalization factor for $\Delta h_{ij}^{(q),m}$.

3. PROPOSED METHOD

3.1. Proposed algorithm

Within the framework provided in [6], we propose DiffRank which can be predicted its upper bound of training error in each boosting step.

```
Input: $S = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_q}$
Initialize $\mathbf{D}^0 = \frac{1}{V}$
For m=1 to M
  Get weak ranking order $\mathbf{h}^m$ according to the $m$-th feature
  Calculate weight $\alpha^m$
      $\alpha^m = \frac{1}{2}\ln\frac{1+\sum_{N_q}\sum_{i>j} D_{q_{ij}}^{m-1}\,\Delta h_{q_{ij}}^m / U_q}{1-\sum_{N_q}\sum_{i>j} D_{q_{ij}}^{m-1}\,\Delta h_{q_{ij}}^m / U_q}$
  Calculate $\mathbf{H}^m$
      $\mathbf{H}^m = \sum_{k=1}^{m} \alpha^k \mathbf{h}^k$
  Update Distribution $\mathbf{D}^m$
      $D_{q_{ij}}^m = \frac{D_{q_{ij}}^{m-1}\exp\left(-\alpha \Delta h_{q_{ij}}^m / U_q\right)}{Z^m}$
      where $Z^m$ is a normalization factor
End For
Output ranking model: $F(\mathbf{X}) = sort\left(\sum_{m=1}^{M} \alpha^m \mathbf{h}^m\right)$
```

Fig.1 DiffRank

DiffRank does not require any additional parameters except training data set $S = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_q}$ to obtain the ranking model

$$F(\mathbf{X}) = sort\left(\sum_{m=1}^{M} \alpha^m \mathbf{h}^m\right).$$

We first initialize distribution D at the very beginning of the algorithm. V (called volume factor) which equals to equals to $\sum_{N_q} 1_{i \succ j}$ represents the all possible situations that can trigger our calculation on rank's difference value, where $1_{i \succ j}$ is an indicator function.

Then, DiffRank generates a weak ranking order $\mathbf{h}^m$ for boosting in each round. In previous works, there are some methods for building weak ranker. Here, we propose the simplest method to construct the weak ranker, that is, judging the general order (descending or ascending) with a specific feature. Once getting the weak ranking order $\mathbf{h}^m$, we calculate the weight α^m for $\mathbf{h}^m$, calculate the current ranking scores $\mathbf{H}^m$ and update the distribution $\mathbf{D}^m$.

In the rest of the section, we will analyze DiffRank's upper bound of training error so that we can guarantee the continuous optimizing in the boosting process.

3.2. Theoretical analysis

In Section 2, we have mentioned that ranking prefers to place the items which obtain higher scores on the top rank positions. Thus, we define the basic rule for general ranking problem.

Definition 1. If item *i* is ranked higher than item *j*, item *i* must receive the higher ranking scores than item *j*, then we call it as practical ranking error and use an indicator function $1_{\Delta H_{ij}<0}$ to quantify the error number.

Now we give the upper bound for DiffRank.

Theorem 1. In DiffRank, expected ranking error in training phrase is bounded by:

$$\frac{1}{V}\sum_{N_q}\sum_{i \succ j} 1_{\Delta H_{ij}<0} \le \prod_{m=1}^{M} Z^m$$

Proof:

$$D_{ij}^{(q),m} = \frac{D_{ij}^{(q),m-1}\exp\left(-\alpha^m \Delta h_{ij}^{(q),m} / U_q\right)}{Z^m} = \frac{\exp\left(-\Delta H_{ij}^{(q),m} / U_q\right)}{V\prod_{m=1}^{M} Z^m}$$

Clearly, $1_{\Delta H_{ij}<0} \le \exp\left(-\Delta H_{ij}^{(q),m} / U_q\right)$, where we set $\exp\left(-\Delta H_{ij}^{(q),m} / U_q\right)$ as loss function. Here, we just consider the situations when $i \succ j$:

$$\sum_{N_q}\sum_{i \succ j} 1_{\Delta H_{ij}^{(q),m}<0} \le \frac{1}{V}\sum_{N_q}\sum_{i \succ j}\exp\left(-\Delta H_{ij}^{(q),m} / U_q\right) \le \prod_{m=1}^{M} Z^m$$

In Theorem 1, we must choose an appropriate weight α to guarantee the upper bound holds, and the upper bound can be eliminated in boosting process so that we can optimize the ranking error in the training process.

Theorem 2. To minimize the ranking error in the training process, α^m should be assigned to:

$$\alpha^m = \frac{1}{2}\ln\frac{1+\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m-1}\,\Delta h_{ij}^{(q),m} \Big/ U_q}{1-\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m-1}\,\Delta h_{ij}^{(q),m} \Big/ U_q}$$

The proof is omitted here, which is similar to that in [7]. The approach is targeted at classification problem.

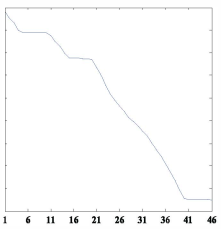
4. EXPERIMENT

4.1. Dataset

We choose a standard dataset MQ2008-list in LETOR 4.0 to implement our experiments. In the dataset, retrieved documents have already been assigned with a fixed number of features, and each document belongs to related queries. Besides, MQ2008-list organizes the data from TREC (Text REtrieval Conference) 2008 which collects the source data from practical Web search.

4.2. Training error

As we have mentioned in Section 3, we calculate the upper bound of training error in DiffRank and trace the value on each round in boosting process. The results are shown in Fig.1(a) where the horizontal axis represents the round and the vertical axis describes the maximum training error on ranking (the coordinate values in vertical axis is based on Base number).



(a) (b)

Fig.2 (a) expected loss value on each round (b) practical training error on each round

According to Fig.1(a), we observe that the curve keeps declining in each round. Then, we do another experiment to verify if the practical training error is declining along with the upper bound descending. We use kendall-τ to monitor the practical training error and draw the curve in Fig.2, where the horizontal axis is kendall-τ value and the vertical axis represents the round.

As we observe from the two figures, the practical training error generally has the descending trend with the upper bound of training error. That means if we can find enough features for ranking objects, we can accurately obtain the ranking model to describe the ranking.

4.3. Comparison of results

We choose some state-of-the-art listwise algorithms, SVM^{map} [8], ListMLE and FeatureRank[9], to compare with DiffRank. The results are shown in Table.1.

We use kendall-τ to quantify the whole ranking error and top-10 resistance rate to describe ranking accuracy in top-10. We choose the first 5% of the retrieved documents to train SVM-MAP, and consider first 50 ranking status to train DiffRank.

Table.1 Performance on MQ2008-list

	kendall-τ	top-10 resistance rate
SVM-MAP	0.204	0.207
ListMLE	0.294	0.099
FeatureRank	0.208	0.322
DiffRank	**0.134**	**0.336**

In the table, DiffRank has the lowest loss on kendall-τ and highest ranking accuracy on top-10 resistance rate. The results show that DiffRank has a great improvement on both whole ranking situation and top ranking accuracy.

5. REFERENCES

[1] T. Qin, *et al.*, "Query-level loss functions for information retrieval," *Information Processing & Management,* vol. 44, pp. 838-855, 2008.

[2] Z. Cao, *et al.*, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 129 - 136

[3] R. Herbrich, *et al.*, "Support vector learning for ordinal regression," in *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, 1999, pp. 97-102.

[4] C. Burges, *et al.*, "Learning to rank with nonsmooth cost functions," in *Advances in Neural Information Processing Systems*, 2006, pp. 395-402.

[5] F. Xia, *et al.*, "Listwise approach to learning to rank: theory and algorithm," in *Proceedings of the 25th International Conference on Machine Learning* 2008, pp. 1192-1199.

[6] J. Friedman, *et al.*, "Special invited paper. additive logistic regression: A statistical view of boosting," *The annals of statistics,* vol. 28, pp. 337-374, 2000.

[7] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning,* vol. 37, pp. 297-336, 1999.

[8] Y. Yue, *et al.*, "A support vector method for optimizing average precision," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007, pp. 271-278.

[9] Y. Wang and W. Mao, "Featurerank: A non-linear listwise approach with clustering and boosting," in *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, 2011, pp. 81-84.