RESEARCH ARTICLE

Listwise Approaches Based on Feature's Ranking Discovery

Yongqing WANG[1], Wenji MAO[2*], Daniel ZENG[2,3], Fen XIA[4]

[1]Alibaba Inc., Hangzhou 310052, China

[2]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China ([*]Contact Author)

[3]Department of Management Information Systems, University of Arizona, Tucson, AZ 85721, USA

[4]Baidu Inc., Beijing 100085, China

Abstract Listwise approaches are an important class of learning to rank, which utilizes automatic learning techniques to discover useful information. Most previous research on listwise approaches has focused on optimizing ranking models on weights and used imprecisely labeled training data. Optimizing ranking model on features was largely ignored. Thus the continuous performance improvement of these approaches was hindered. To address the limitations in previous listwise work, in this paper, we propose a quasi-*KNN* model to discover feature's ranking and employ rank addition rule to calculate the combination weight. On the basis of this, we propose three listwise algorithms, *FeatureRank*, *BL-FeatureRank* and *DiffRank*. Our proposed algorithms can be perfectly applied to strict ordered ranking training set. The experiment results show that our proposed methods gain better performance on several measures than the state-of-the-art listwise algorithms.

Keywords learning to rank, listwise approach, feature's ranking discovery

1 Introduction

With the rapid development of Internet and mobile technologies, huge amount of open source data are available on the Web. While these abundant online data bring about great advantages in information collection and sharing, they also raise great challenges for information retrieval. Learning to rank, which utilizes machine learning techniques to automatically discover useful information, are in great demand in recent years, and has become a hot topic in information retrieval and knowledge discovery.

Learning to rank is aimed at effective information retrieval and discovery. It can be generally categorized into three classes: pointwise [1, 2], pairwise [3-5] and listwise [6-9] approaches. These three approaches solve the ranking problem with different concerns. Pointwise and pairwise approaches are mainly concerned with the partial ranking issue. They both ignore the key information that a document may have different relevance degrees according to the types of queries. Partial ranking methods have proven to be less effective in learning to rank [10].

Listwise approach is promising in addressing the query-document group structure. For example, the state-of-the-art algorithm *ListMLE* [7] constructs the ranking probability distribution with *Plackett-Luce* model which deals with the query level information. Within listwise approaches, most previous research has adopted quasi feature composition method. For instance, the ranking model $f(\mathbf{x}) = \vec{\omega}^{\mathrm{T}}\mathbf{x}$ (where $\mathbf{x}$ is the feature matrix and $\vec{\omega}$ is the weight vector) introduced in [6, 7] could be decomposed into

$$f(\mathbf{x}) = \omega_1\vec{x}_1 + \omega_2\vec{x}_2 + \ldots + \omega_m\vec{x}_m \quad (1)$$

where $\vec{x}_i$ represents the *i-th* feature vector and ω_i could be regarded as the weight of corresponding feature. Most listwise algorithms linearly combine features to produce rank models. Their performance typically degenerates when these algorithms are applied to

applications where features influence ranking through a non-linear way. Therefore, the potential of performance improvement through non-linearly combining ranking features was largely ignored in existing listwise approaches.

Another problem in previous listwise research is that there exists clear discrepancy between the accuracy of the training data and that required for the ranking results. Ranking model gets strict ordered ranking sequence from imprecisely labeled ratings in training set. Usually, these labeled ratings are merely expressed as, for example, "*definitely relevant*", "*partially relevant*" or "*not relevant*". On the one hand, current progress in Web-based research and applications has made it possible to acquire more and more accurate rating data. On the other hand, it is necessary to rely on more precise inputs for the continuous improvement of ranking methods as well as empirical studies and comparisons of different approaches.

In this paper, we focus on listwise approaches to solve the ranking problem. We consider optimizing ranking model on features and present a new model for feature's ranking discovery. On the basis of this, we propose three listwise algorithms inspired by boosting idea. The algorithm *FeatureRank* is the basic algorithm to handle weight producing under our feature's ranking discovery model. *BL-FeatureRank* is an improved algorithm to ensure decrease of *FeatureRank*'s training error during the learning process. To improve the convergence rate of *FeatureRank* and *BL-FeatureRank*, we further propose a fast convergence algorithm *DiffRank*. We provide theoretical analyses of our proposed algorithms. We use strict ordered data instead of labeled data and conduct experiments to verify the effectiveness of our approaches. Through experimental results, we demonstrate performance improvement of our algorithms in comparison with the state-of-the-art listwise algorithms.

This paper is organized as follows. *Section 2* introduces the background of our work,. *Section* 3 presents our proposed feature's ranking discovery model which refers the *KNN* idea. *Section 4* presents our proposed three listwise algorithms: *FeatureRank*, *BL-FeatureRank* and *DiffRank*, and also provide the theoretical analyses of these algorithms. Experimental studies and comparative results are provided in *Section 5*. The paper concludes in *Section 6*.

2 Background

Our work is grounded in information retrieval and especially Web-based search system, which studies the techniques of discovering related documents based on the input query. According to the query, the system retrieves as many as possible relevant documents and presents them in descending order of relevance degree.

2.1 Document rating and features

In contrast to the labeled rating ranking set, in a training set with strict ordered ratings, every document has been assigned to a unique rating used to determine the document's position in the ranking list. The rating generally comes from practical Web search system.

Each document can be modeled by traditional information retrieval methods, such as term frequency (*tf*), inverse document frequency (*idf*), document length (*dl*), *BM25* [11], *LMIR* [12] algorithms and their combination forms. Moreover, different interests on title, abstract, etc can also establish various features of a document.

2.2 Boosting

Our proposed listwise algorithms are inspired by boosting idea. The significant advantage of boosting is that it can dramatically increase the performance even using a relatively weak learning model. This is due to the predictable bound of the training error, as the error rate can steadily decrease during the training phase [13, 14].

Boosting [13, 15, 16] is essentially one kind of "committee" method. In learning to rank, some pervious work has utilized boosting techniques to gain ranking models [5, 8, 17]. The original motivation is to combine the outputs of many "weak" learners to produce a powerful "committee". Boosting method sequentially applies a weak learning model $h_m(x)$ to

repeatedly modified versions of the data, thereby producing the final prediction function:

$$H(x) = f\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right), \quad (2)$$

where α_m is calculated through the boosting algorithm, and it weights the contribution of each $h_m(x)$. The more accurate weak learner produced in the sequence, the higher influence value represented on α_m.

2.3 Ranking measures

Ranking measures are used to assess the performance of ranking models. The most popular ranking measures in information retrieval are Mean Average Precision (*MAP*) [18], Normalized Discounted Cumulative Gain (*NDCG*) [19], *kendall-*τ distance [20], among others. In terms of the strict ordered ranking set, we adopt *NDCG* and *kendall-*τ in evaluating our proposed algorithms.

NDCG focuses on top-*K* ranking accuracy. The whole process of *NDCG* can be divided into four phases: gain evaluation, gain accumulation, discount and normalization. The discount formulation can be described as

$$DCG[i] = \begin{cases} CG[i], & \text{if } i < b \\ CG[i-1] + G[i]/\log_b i, & \text{otherwise} \end{cases} \quad (3)$$

where *b* is the base number, *i* is the position number, *G*, *CG* and *DCG* represent gain evaluation, gain accumulation and discount respectively. Finally, *NDCG* value is computed as

$$NDCG[i] = \frac{DCG[i]}{DCG_i[i]} \quad (4)$$

where DCG_i is the ideal *DCG* value at position *i*.

Kendall-τ distance is a statistic method used to measure the association between two ranking list. The value directly represents the similarity between two lists. *Kendall-*τ can be described as

$$T(\pi,\sigma) = \frac{1}{z}\sum_{k<l} I\{[\pi_k - \pi_l][\sigma_k - \sigma_l] < 0\} \quad (5)$$

where π, σ are the two comparable lists, π_k, σ_l mean the elements in the list π, σ, *I* is an indicator function which has value 1 or 0, *z* is the normalization factor. For example, $\pi = [1,3,2]$ and $\sigma = [1,2,3]$, the *kendall-*τ value is

$$T(\pi,\sigma) = \frac{1}{z}\left(I\{[1-3][1-2] < 0\} + \ldots + I\{[3-2][2-3] < 0\}\right)$$
$$= \frac{1}{3} * 1 = 0.\dot{3}$$

3 Feature's ranking discovery

3.1 Problem definition

In this section, we propose to optimize ranking model on features so as to avoid the optimizing bottleneck when weight producing is close to optimal. We first define the notations used in this paper. 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. The *m-th* document feature dimension is represented as $\mathbf{x}_m^{(i)}$ 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. Thus, the training set *S* can be described as $S = (\mathbf{X}, \mathbf{Y}) = \left\{\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right\}_{i=1}^{N_q}$

3.2 Quasi-*KNN* method

We employ the idea similar to *KNN* method in finding hidden feature's ranking order in single feature. For a training set, we first choose one specific feature dimension (e.g. the *m-th* feature dimension) to find the feature's ranking. We execute the following steps for each query in the feature:

1) Separate the training set by queries, and rearrange the feature values according to relevance degree in descending order.
2) Divide each query into *K* intervals (*K* can be determined by need) with equal length. Number each class sequentially from 1 to *K*.
3) Gather values in same intervals and set the mean of these values $I_{m,k}$ as the interval center, the index (*m*, *k*) represents the *k-th* interval in *m-th* feature dimension.

After executing all the steps above for each feature, we gain interval center matrix ***I*** where the row

represents the feature and the column describes the interval. *Figure 1* describes the whole process of quasi-*KNN* method.

Input: training set *S*, parameter *K*, ***I***[*M*, *K*]

1. For m=*1* to *M*
   (a) For *i=1* to *N* do
      i. Divide $\mathbf{x}_m^{(i)}$ into *K* intervals with equal length
      i. Gather the values by intervals
      End for
   (b) Calculate the mean of each interval values as the interval center $I_{m,k}$
   End for
2. Output: interval center matrix ***I***

Fig.1 Quasi-*KNN* method for feature's ranking discovery

After getting the interval center matrix, feature values can be classified into *Interval*s by Euclidean distance according to the matrix ***I*** (like *KNN* method). We first propose that values falling into the *Interval* with larger column number in row will gain the higher relevance preference (see equation (6)).

$$\begin{cases} \{\mathbf{val}_l \mid val_{l,i} \in Interval_l\} \\ \{\mathbf{val}_k \mid val_{k,i} \in Interval_k\} \Rightarrow \mathbf{val}_l \succ \mathbf{val}_k, \\ \qquad l > k \end{cases} \tag{6}$$

where $\mathbf{val}_l$ is the set of feature values belonging to $Interval_l$ and $val_{l,i}$ is the *i-th* value in the set. If the column number *l* is greater than *k*, values in set $\mathbf{val}_l$ can be ranked higher than values in set $\mathbf{val}_k$. All the compared values must belong to the same feature dimension in the same query.

In *Interval*s, we propose inner ranking order is calculated using equation (7) and (8).

$$g\left(val_{l,i}, val_{l,j}\right) > 0 \Rightarrow val_{l,i} \succ val_{l,j}, \tag{7}$$

where

$$g\left(val_{li}, val_{lj}\right) = \begin{cases} \left(val_{l,i} - val_{l,j}\right)\left(I_{m,l} - I_{m,l-1}\right), 1 < l \le K \\ \left(val_{l,i} - val_{l,j}\right)\left(I_{m,l+1} - I_{m,l}\right), l = 1 \end{cases}, \tag{8}$$

and *K* is the *Interval* number.

For example, in the third feature dimension, if *l=3*, *K=8*, *i=5*, *j=7*, $I_{3,3}$*=0.3*, $I_{3,2}$*=0.6*, $val_{3,5}$*=0.4*, $val_{3,7}$*=0.5*, then the feature value *0.4* is ranked higher than *0.5*.

3.3 Theoretical analysis

Our method is aimed to effectively improve the accuracy of feature's ranking discovery in comparison with linear way. Before proving this, we illustrate the general linear way of feature's ranking discovery.

We have mentioned the general ranking model $f(\mathbf{x}) = \vec{\omega}^{\mathrm{T}}\mathbf{x}$. From the perspective of feature division, the general feature's ranking discovery can be seen as a linear way. Take the following matrix as an example of training set,

$$\mathbf{x} = \begin{pmatrix} 0.8 & 0.7 & 0.3 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.4 & 0.6 & 0.7 \\ 0.4 & 0.2 & 0.6 & 0.8 & 0.5 \\ 0.5 & 0.3 & 0.5 & 0.5 & 0.8 \\ 0.2 & 0.4 & 0.8 & 0.9 & 0.9 \end{pmatrix},$$

whatever value weight $\vec{\omega}$ takes (except zero), each column will be added to function *f* in linear way:

$$f = \omega_1 \times \langle 0.8, 0.6, 0.4, 0.5, 0.2 \rangle + \ldots + \omega_5 \times \langle 0.9, 0.8, 0.5, 0.7, 0.4 \rangle$$

If we have the best way to calculate the weight, the bottleneck would be the feature's ranking discovery method.

Theorem 1. Quasi-*KNN* method can identify no less correct feature's ranking order than linear discovery way.

Proof. The proof of *Theorem 1* is given in *Appendix*.

Now we give a formal definition of weak ranker.

Definition 1. Weak ranker is the ranker that can gain ranking order with over intermediate measure value using specific measure function.

In this paper, we set the specific measure function as *kendall-*τ because of the advantages it has (we shall discuss this in *Section 4*). Linear ranking discovery and quasi-*KNN* method can both satisfy the definition of weak ranker when using measure function *kendall-*τ. It is also convenient and effective to construct a weak ranker, as the complexity of establishing quasi-*KNN* model is *O*(*n*). Thus, quasi-*KNN* method can be a good substitute for linear feature's ranking

discovery.

4 Ranking algorithms

4.1 Rank addition

In this section, we discuss feature's ranking combination. We first discuss the rank addition rule, which is different from general algebra addition.

The result of rank addition only focuses on the ranking order. For example, the ranking order of *<0.9, 0.7, 0.8, 0.5, 0.6>* is equal to that of *<0.8, 0.5, 0.7, 0.3, 0.4>*. Both of their orders are *<5, 3, 4, 1, 2>*. Thus, we have *Theorem 2* below.

Theorem 2. The space of rank addition is discrete, which results from the order's preference in the list.

Below we propose our ranking algorithms *FeatureRank*, *BL-FeatureRank* and *DiffRank*. They differ in their treatments of rank addition,

Input: $S=\left\{\mathbf{x}^{(i)},\mathbf{y}^{(i)}\right\}_{i=1}^{N}$, parameters *E* and *M*

1. Initialize $w_0(\mathbf{X})=\mathbf{1}$
2. For *m=1* to *M* do
   (a) Create weak ranker h_m according to the *m-th* feature.
   (b) Computing λ_m

$$\lambda_m=\frac{1}{2}\ln\frac{\sum_{i=1}^{N}w_{m-1}^{(i)}\left(1+E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}{\sum_{i=1}^{N}w_{m-1}^{(i)}\left(1-E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}$$

   (c) Create H_m

$$H_m\left(\mathbf{x}_m^{(i)}\right)=\pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\lambda_m h_m\left(\mathbf{x}_m^{(i)}\right)\right)$$

   (d) Update w_m

$$w_m(\mathbf{X})=\exp\left(-E\left(H_m(\mathbf{X}),\pi(\mathbf{Y})\right)\right)$$

   End for
3. Output ranking model:

$$F(\mathbf{X})=\pi\left(H_{M-1}\left(\mathbf{x}_{M-1}^{(i)}\right)+\lambda_M h_M\left(\mathbf{x}_M^{(i)}\right)\right)$$

Fig.2 *FeatureRank* algorithm

4.2 FeatureRank

Following the general idea of boosting [8, 14], we devise an algorithm *FeatureRank* to combine weak rankers using rank addition. The actual algorithm is given in *Figure 2*.

In *FeatureRank*, we compute ranking order for every feature. The ranking order calculation function is depicted by symbol π; *i* is the query index; *N* is the query number in training set; *M* is the feature number; λ is the weight; *w* can be seen as a distribution used to weight the weak ranker's contribution; and *E* is the evaluation function ranged in [*-1, 1*]. Here, we use *kendall-*τ to measure the difference between $h\left(\mathbf{x}^{(i)}\right)$ and $\pi\left(\mathbf{y}^{(i)}\right)$. $E_{h_k}\left(\mathbf{x}_m^{(i)}\right)$ is the abbreviation of $E\left(h_m\left(\mathbf{x}_m^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)$.

4.2.1 Theoretical analysis

We used *kendall-*τ in our algorithm to measure the similarity between predicted ranking result and ground truth. This is because *kendall-*τ has the following two distinct properties.

Property 1. *Kendall-*τ only expresses the relationship in order.

For example, in *kendall-*τ, *T*(*<1, 0>*, *<0.3, 0.1>*) equals to *0*. The result ignores the specific values in vectors.

Property 2. The symmetrical ranking order has symmetrical measure values at the middle point of *kendall-*τ measure.

For example, if the correct ranking order is *<1,2,3>* (or *<3,2,1>*), ranking order *<2,1,3>* and *<3,1,2>* is symmetrical. And also their *kendall-*τ values are symmetrical at middle point *0.5*. It is easy to get *T*(*<2,1,3>*, *<1,2,3>*) = *1-T*(*<3,1,2>*, *<1,2,3>*) = *1/3*. In this situation, we call *<2,1,3>* the inverse of *<3,1,2>*. These two properties ensure that *kendall-*τ can easily distinct the "weak" ranker we discussed in *Section 3*.

According to the demand in *FeatureRank*, we let the measure function be

$$\begin{aligned}&E\left(h\left(\mathbf{x}^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\&=1-2T\left(h\left(\mathbf{x}^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\end{aligned}. \quad (9)$$

Then, we define the *FeatureRank*'s loss function

in exponent form

$$L\left(\mathbf{y},h\left(\mathbf{X}\right)\right)=\sum_{i=1}^{N}\exp\left(-E\left(h\left(\mathbf{x}^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\right).$$

The detailed form of loss function is

$$\begin{aligned}L\left(\mathbf{y},h\left(\mathbf{X}\right)\right)=\sum_{i=1}^{N}\exp\big(-E\big(&\\ \pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\lambda_m h_m\left(\mathbf{x}_m^{(i)}\right)\right),\pi\left(\mathbf{y}^{(i)}\right)\big)&\end{aligned} \tag{10}$$

Lemma 1. In *FeatureRank*, if the loss function is equation (10), and δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend with $h_m\left(\mathbf{x}_m^{(i)}\right)$, weight λ_k can gain the optimum value to make loss value minimum.

$$\lambda_k=\frac{1}{2}\ln\frac{\sum_{i=1}^{N}w_{k-1}^{(i)}\left(1+E_{h_k}\left(\mathbf{x}_k^{(i)}\right)\right)}{\sum_{i=1}^{N}w_{k-1}^{(i)}\left(1-E_{h_k}\left(\mathbf{x}_k^{(i)}\right)\right)}, \tag{11}$$

where $w_{k-1}^{(i)}=\exp\left(-E_{H_{k-1}}\left(\mathbf{x}_{k-1}^{(i)}\right)\right)$ and

$$\begin{aligned}\delta_i=&E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\&+\lambda_m E\left(h_m\left(\mathbf{x}_m^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\&-E\left(\pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\lambda_m h_m\left(\mathbf{x}_m^{(i)}\right)\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\end{aligned} \tag{12}$$

Proof. The proof of *Lemma 1* is given in *Appendix*.

Although *Lemma 1* proves its effectiveness, *FeatureRank* must rely on some specific conditions to ensure that δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend with $h_m\left(\mathbf{x}_m^{(i)}\right)$. Otherwise the performance curve of *FeatureRank* might have vibration in training process.

4.3 BL-FeatureRank

In most cases, *FeatureRank* cannot strictly satisfy conditions in *Lemma 1*, and this frequently causes performance curve's vibration in training process. Thus, we develop an improved *FeatureRank* algorithm to monitor the weak ranking results from each single feature and control the boosting process. We call the monitor as "balance control" and name the algorithm as *BL-FeatureRank*.

The formal expression of balance control is as follows.

$$\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)=\arg\max_h\begin{pmatrix}E_{h_m}\left(\mathbf{x}_m^{(i)}\right),\\E_{h_m''}\left(\mathbf{x}_m^{(i)}\right),\\E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)\end{pmatrix}. \tag{13}$$

First, we propose a preview method (equation (14)) for feature ranking. It intends to verify that if current ranking result on the feature can make the boosting performance better.

$$\begin{aligned}h_m'\left(\mathbf{x}_m^{(i)}\right)=&E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)\\&+E_{f_m}\left(\mathbf{x}_m^{(i)}\right)h_m\left(\mathbf{x}_m^{(i)}\right)\end{aligned} \tag{14}$$

Then, we take the interaction between $H_{m-1}\left(\mathbf{x}_i\right)$ and $h_m\left(\mathbf{x}_i\right)$ into consideration. Then we get another preview function

$$h_m''\left(\mathbf{x}_m^{(i)}\right)=\beta H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\left(1-\beta\right)h_m\left(\mathbf{x}_m^{(i)}\right)$$

where $\beta=E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)\Big/\left(E_{h_m}\left(\mathbf{x}_m^{(i)}\right)+E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)\right)$.

Input: $S=\left\{\mathbf{x}^{(i)},\mathbf{y}^{(i)}\right\}_{i=1}^{N}$, and parameters *E* and *M*

1. Initialize $w_0\left(\mathbf{X}\right)=\mathbf{1}$
2. For *t=1* to *T* do
   (a) For *m=1* to *M* do
      i. Create weak ranker h_m on the *m-th* feature.
      ii. Computing $\lambda_{t,m}$

$$\lambda_{t,m}=\frac{1}{2}\ln\frac{\sum_{i=1}^{N}w_{t,m-1}^{(i)}\left(1+E_{h_{t,m}}\left(\mathbf{x}_m^{(i)}\right)\right)}{\sum_{i=1}^{N}w_{t,m-1}^{(i)}\left(1-E_{h_{t,m}}\left(\mathbf{x}_m^{(i)}\right)\right)}$$

      iii. Create $H_{t,m}$

$$H_{t,m}\left(\mathbf{x}_m^{(i)}\right)=\pi\left(H_{t,m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\lambda_{t,m}h_{t,m}\left(\mathbf{x}_m^{(i)}\right)\right)$$

      iv. Update $w_{t,m}$

$$w_{t,m}\left(\mathbf{X}\right)=\exp\left(-E\left(H_{t,m}\left(\mathbf{X}\right),\pi\left(\mathbf{Y}\right)\right)\right)$$

   End for

   End for
3. Output ranking model:

$$F\left(\mathbf{X}\right)=\pi\left(H_{T,M-1}\left(\mathbf{x}_{M-1}^{(i)}\right)+\lambda_{T,M}h_{T,M}\left(\mathbf{x}_M^{(i)}\right)\right)$$

Fig. 3 *BL-FeatureRank* algorithm

We choose the best ranking as the current ranking order from original feature's ranking order h_m, preview function and last combination result. The algorithm *BL-FeatureRank* is similar to *FeatureRank* except for the construction of weak ranker. The actual algorithm is given in *Figure 3* (where *T* is the cycle time).

4.3.1 Theoretical analysis

Now we discuss the effectiveness of balance control in *BL-FeatureRank*.

Lemma 2. If *FeatureRank* adopts balance control (equation (13)), δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend as that of $h_m\left(\mathbf{x}_m^{(i)}\right)$.

Proof. The proof of *Lemma 2* is given in *Appendix*.

Lemma 2 proves that *BL-FeatureRank* doesn't lead to the vibration of performance curve in the process, therefore, ranking result of *BL-FeatureRank* is improved steadily.

4.4 DiffRank

As the convergence rate of ranking algorithm is an important factor for practical Web search system, we further improve *FeatureRank* and *BL-FeatureRank*, and devise a new algorithm *DiffRank* to enhance the convergence rate of ranking result.

Because of discrete result space, sometimes the rank addition in *FeatureRank* and *BL-FeatureRank* can be problematic. For example, let $\alpha_1\langle 3,2,1\rangle$ plus to $\alpha_2\langle 3,1,2\rangle$, weight $\alpha_1=1$, $\alpha_2=0.8$ and weight $\alpha_1=0.7$, $\alpha_2=0.2$ have the same addition result. Thus both *FeatureRank* and *BL-FeatureRank* cannot calculate the optimal weight at once. To avoid this problem, we change the ranking model to

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

where α^m is the weight value for weak ranker $\mathbf{h}^m$, $\mathbf{H}^{m-1}$ is the combined ranking scores at (*m-1*)-*th* feature and equal to $\sum_{k=1}^{m-1}\alpha^k\mathbf{h}^k$, and *sort* is a sort function in which a higher score can be assigned a higher rank. *DiffRank* algorithm is given in *Figure 4*.

Input: $S=\left\{\mathbf{x}^{(i)},\mathbf{y}^{(i)}\right\}_{i=1}^{N}$ and ranking accuracy *prec*

1. Initialize $\mathbf{D}^0=\mathbf{1}/V_{prec}$
2. For *m=1* to *M* do
   (a) Create weak ranker h_m on the *m-th* feature.
   (b) Computing α_m

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

   (c) Create $\mathbf{H}_m$

$$\mathbf{H}^m=\sum_{m=1}^{M}\alpha^m\mathbf{h}^m(\mathbf{X})$$

   (d) Update $\mathbf{D}^m$

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

   End for
3. Output ranking model: $F(\mathbf{X})=sort\left(\sum^{M}\alpha^m\mathbf{h}^m\right)$

Fig. 4 *DiffRank* algorithm

In the algorithm, V_{prec} is the volume factor (we shall explain it later) and *prec* is the accuracy rate that we want. $\Delta h_{ij}^{(q),m}$ is the ranking scores' difference value between rank *i* and *j*; U_q is the normalization factor for $\Delta h_{ij}^{(q),m}$; $D_{ij}^{(q),m}$ is a distribution which calculates the contribution of current weak ranker; Z^m is the normalization factor for distribution $D_{ij}^{(q),m}$.

4.4.1 Theoretical analysis

In *DiffRank*, the optimization objective is to place documents in descending order according to relevance scores. So we define an evaluation function at the *m-th* feature,

$$Eval=\sum_{N_q}\sum_{i\succ j}\Delta H_{ij}^{(q)} \qquad (15)$$

where $\Delta H_{ij}^{(q)}$ is the ranking scores' difference value between rank *i* and *j*. A ranking model with good performance is able to gain the maximum value on equation (15).

Because of the advantages of exponent function in optimization, we introduce the exponent form of equation (15). So the loss function is

$$Loss_m = \sum_{N_q}\sum_{i\succ j}\exp\left(-\frac{1}{U_q}\left(\Delta H_{ij}^{(q),m}\right)\right) = \sum_{N_q}\sum_{i\succ j}\exp\left(-\frac{1}{U_q}\left(\Delta H_{ij}^{(q),m-1} + \alpha\Delta h_{ij}^{(q),m}\right)\right) \quad (16)$$

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

Definition 2. In ranking issues, if document *i* is ranked higher than document *j*, document *i* must receive the higher ranking scores than document *j*. For any violation to this, we call it a practical ranking error and use an indicator function $\sum_{N_q}[\![1]\!]_{\Delta H_{ij}<0}$ to quantify the error number.

Consider the symmetry of ranking order, we only calculate the practical ranking error when $i \succ j$. Thus, we can rewrite the function for practical ranking error as $\sum_{N_q}\sum_{i\succ j}[\![1]\!]_{\Delta H_{ij}<0}$.

The volume factor records all possible violation number, and particularly, in the top *prec*. Thus,

$$V_{prec} = \sum_{N_q}[\![1]\!]_{i\succ j, i<prec}.$$

Lemma 3. In *DiffRank*, the practical ranking error is bounded by

$$\frac{1}{V}\sum_{N_q}\sum_{i\succ j}[\![1]\!]_{\Delta H_{ij}<0} \le \prod_{m=1}^{M} Z^m \quad (17)$$

Proof. The proof of *Lemma 3* is given in *Appendix*.

Lemma 3 provides guidance to the optimization: reducing the upper bound of practical ranking error so as to decrease practical ranking error.

Lemma 4. In *DiffRank*, if $\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m} = 1$ and $\phi^m \in [0,1]$, for minimizing ranking error in training process, α^m should be assigned to:

$$\alpha^m = \frac{1}{2}\ln\frac{1+\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m-1}\Delta h_{ij}^{(q),m}/U_q}{1-\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m-1}\Delta h_{ij}^{(q),m}/U_q} \quad (18)$$

where

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

Proof. The proof of *Lemma 4* is given in *Appendix*.

In *DiffRank*, as the condition $\sum_{N_q}\sum_{i\succ j} D_{ij}^{(q),m} = 1$ and $\phi^m \in [0,1]$ holds, the training process in the algorithm can be seen as the one leading to optimization.

5 Experiments

5.1 Data set

We conduct experiments on the data set *MQlist-2008* of *LETOR4.0* [21]. *MQlist-2008* collects the data from *TREC 2008* and assigns each document a unique relevance degree for a query. The data set consists of five text files and their combinations. Each document has 46 features in this data set. To facilitate the quasi-*KNN* procedure based on single feature, we separate the data set by features in descending order according to relevance degree and grouped by query id.

5.2 Study on feature's ranking discovery with quasi-*KNN*

In *Section 3*, we have discussed that quasi-*KNN* can recognize no less correct feature's ranking order than linear way if feature values are in homogeneous distribution. Based on the practical data set, we first verify if *MQlist-2008* satisfies the distribution. We first define four types of global center:

1) Normal center. The selection of global interval center is the same as that in *Section 3*.
2) Prior center. The difference between prior center and normal center is that prior center gathers one third values of an interval and these selected elements are prior to others. Set the mean of these elements as the global interval center.
3) Middle center. Different from prior center, the collected elements are located in the middle of the interval.
4) Posterior center. The collected elements are placed in the rear of the interval.

We randomly choose three features — the *19th* feature, *21th* feature and *5th* feature for observation and show them on *Figure 5*, where the horizontal axis represents interval number and vertical axis is interval

center value. As shown in *Figure 5*, all types of global interval center has similar variation tendency. In particular, the up and down sequence on prior center and posterior center is reversed at inflection point (see *19th* feature and *5th* feature). This basically indicates that feature values in *MQlist-2008* are homogeneously distributed.

In addition, from the analysis in *Section 4*, we can infer that quasi-*KNN* method has better performance on feature's ranking discovery than linear way when cluster center has the property shown in the experiment.

5.3 Robustness comparison: FeatureRank versus BL-FeatureRank

In *Section 4*, we conduct theoretical analysis on balance control, and prove that *BL-FeatureRank* can perform more steadily than *FeatureRank* in training process. In this section, we verify theoretical result through experiment.

To compare *BL-FeatureRank*'s training process with that of *FeatureRank*, we add cycle time *T* in *FeatureRank*. *Figure 6* shows the experimental result. The dash lines separate the zone by *T* so that we can observe different phases of both algorithms in boosting process. The result shows that in every phase, the performance curve of *BL-FeatureRank* is more stable than that of *FeatureRank*. *BL-FeatureRank* maintains this persistent improvement all through the training process.

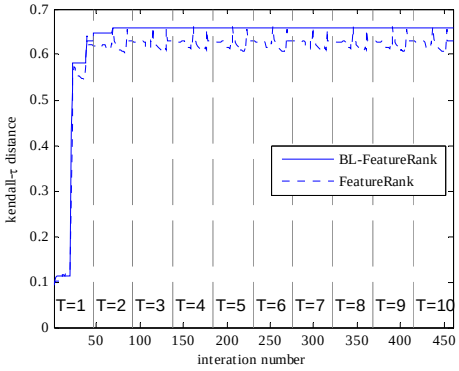
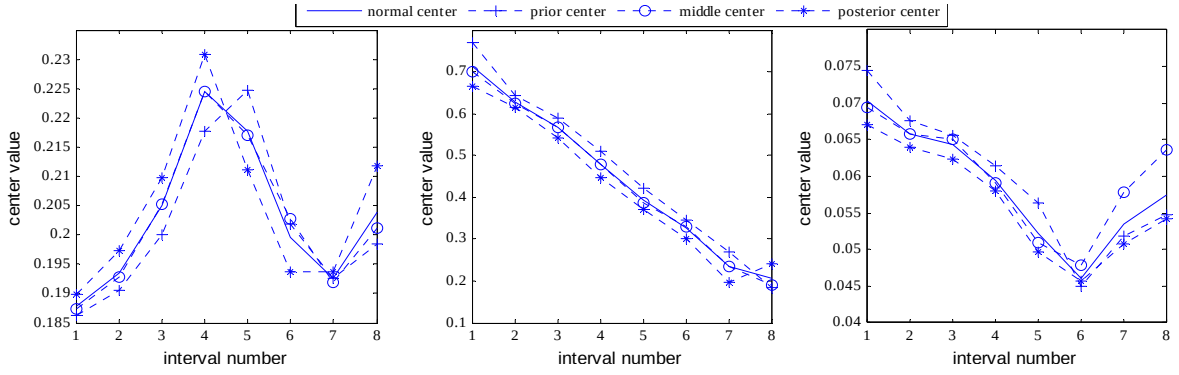


Fig. 6 Robustness comparison of *FeatureRank* and *BL-FeatureRank*

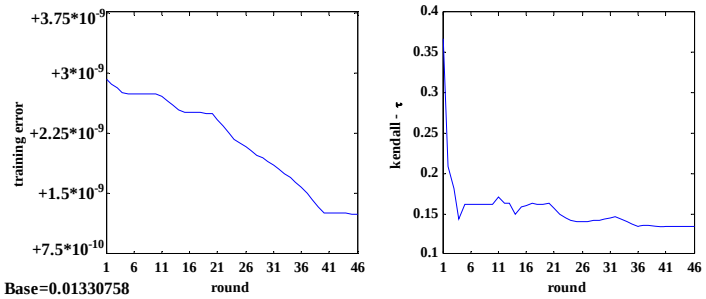
5.4 Practical ranking error in DiffRank

We have proved that in *DiffRank*, practical ranking error is bounded by the loss value. We conduct an experiment to trace the expected loss value in each round of the boosting process, and observe whether practical ranking error is limited by the expected loss value meanwhile. The loss value and practical ranking error are depicted in *Figure 7*. In *Figure 7 (a)*, 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). In *Figure 7 (b)*, the horizontal axis is *kendall-τ* value and the vertical axis represents the round.



(a)19th feature (b)21th feature (c)5th feature

Fig. 5 Four types of interval center when *K=8*



(a) (b)

Fig. 7 (a) expected loss value in each round (b) practical ranking error in each round

We find out that the loss value keeps declining in each round. At the same time, the loss value effectively limits the practical ranking error which generally keeps the declining trend in the training process.

5.5 Comparison with the state-of-the-art algorithms

With strict ordered ranking set, three state-of-the-art algorithms, *ListMLE* [7], *RankCosine* [10] and *SVM*^{map} [9] can accommodate the characteristic of the data set, and thus we choose them for comparison. As to the practical ranking demands on top documents, we use *NDCG*[19] to measure the ranking results. Because of the strict ordered ranking set, we calculate *NDCG* without exponent form. As the accuracy of the top *10* results has significant value for practical ranking system, we also adopt top-*10* persistence rate to measure the ranking performance in the experiment. Top-*10* persistence rate function is computed as

$$\frac{\sum_{i=1}^{10}[\![1]\!]_{P_i=1}}{10}$$

where $[\![1]\!]_{P_i=1}$ equals to 1 when the document is located in top *10* at position *i*. Otherwise, $[\![1]\!]_{P_i=1}$ equals to zero.

The experimental results are given in *Table 1*, *Table 2* and *Figure 8*. The results show that *DiffRank* performs best under *kendall-* τ , top-*10* persistence rate and *NDCG*. Besides *DiffRank*, *FeatureRank* and *BL-FeatureRank* also perform better than other algorithms. The main reason is that *DiffRank* reaches the optimum point fastest than *FeatureRank* and *BL-FeatureRank*, and the optimization directly on feature's ranking results is more effective than the solutions in other methods.

Table 1 Performance comparison of ranking algorithms

	kendall- τ	*top-10* persistence rate
DiffRank	**0.180**	**0.337**
BL-FeatureRank	0.196	0.322
FeatureRank	0.208	0.322
ListMLE	0.294	0.099
SVM^{map}	0.204	0.207
RankCosine	0.448	0.014

Table 2 Performance comparison on *NDCG*

NDCG@k	1	2	3	4	5	6	7	8	9	10
DiffRank	**0.976**	**0.973**	**0.971**	**0.970**	**0.970**	**0.970**	**0.969**	**0.969**	**0.969**	**0.969**
BL-FeatureRank	0.971	0.968	0.967	0.967	0.966	0.966	0.966	0.966	0.966	0.966
FeatureRank	0.971	0.968	0.967	0.967	0.966	0.966	0.966	0.966	0.966	0.966
ListMLE	0.861	0.857	0.850	0.850	0.848	0.848	0.847	0.845	0.845	0.844
SVM^{map}	0.953	0.948	0.947	0.947	0.946	0.947	0.946	0.945	0.944	0.944
RankCosine	0.628	0.617	0.611	0.609	0.608	0.608	0.606	0.606	0.606	0.605

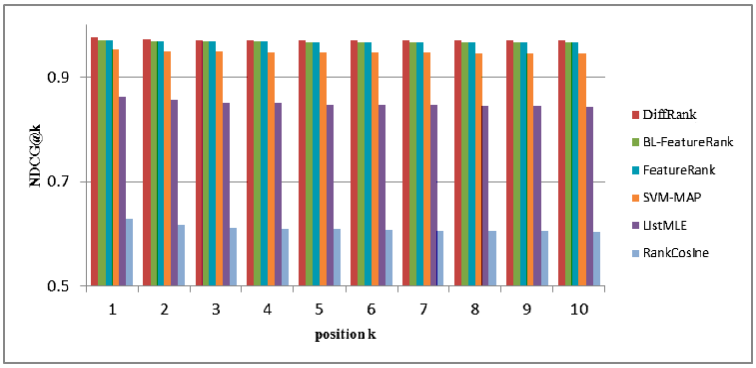


Fig. 8 Performance comparison on *NDCG*

6 Conclusions

Listwise approaches are an important class of learning to rank, which has key applications in information retrieval and Web-based search. This paper discusses the limitations in previous listwise approaches and proposes a quasi-*KNN* method to conduct feature's ranking discovery. In addition, we propose three new ranking algorithms, *FeatureRank*, *BL-FeatureRank* and *DiffRank*, to combine feature's ranking together, and provide theoretical analyses as well as experimental results to verify the effectiveness of the proposed model and algorithms. The experimental results show the consistency of theory analysis and experimental verification. We also conduct an experiment to compare the performance of our approaches with other state-of-the-art algorithms. The results show the improvement by our proposal on several measures. Our ongoing work is exploring how to apply our ranking model and listwise algorithms to practical search environment and improve the performance of Web search systems.

References

1. Crammer K, Singer Y. Pranking with ranking. Advances in Neural Information Processing Systems, 1: 641-648, 2002.
2. Li P, Burges C, Wu Q, Platt J, Koller D, Singer Y, Roweis S. Mcrank: learning to rank using multiple classification and gradient boosting. Advances in Neural Information Processing Systems, 2007.
3. Cao Y, Xu J, Liu T, Li H, Huang Y, Hon H. Adapting ranking SVM to document retrieval. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 186-193. ACM, 2006.
4. Tsai M, Liu T, Qin T, Chen H, Ma W. FRank: a ranking method with fidelity loss. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 383-390. ACM, 2007.
5. Freund Y, Iyer R, Schapire R, Singer Y. An efficient boosting algorithm for combining preferences. The Journal of Machine Learning Research, 4: 933-969, 2003.
6. Cao Z, Qin T, Liu T, Tsai M, Li H. Learning to rank: from pairwise approach to listwise approach. Proceedings of the 24th International Conference on Machine Learning, 129 - 136. ACM, 2007.
7. Xia F, Liu T, Wang J, Zhang W, Li H. Listwise approach to learning to rank: theory and algorithm. Proceedings of the 25th International Conference on Machine Learning, 1192-1199. ACM, 2008.
8. Xu J, 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,391-398. ACM, 2007.
9. Yue Y, Finley T, Radlinski F, Joachims T. A support vector method for optimizing average precision. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development

in Information Retrieval, 271–278. ACM, 2007.
10. Qin T, Zhang X, Tsai M, Wang D, Liu T, Li H. Query-level loss functions for information retrieval. Information Processing & Management, 44(2): 838-855, 2008.
11. Robertson S E. Overview of the okapi projects. Journal of Documentation, 53(1): 3-7, 1997.
12. Zhai C, Lafferty J. A study of smoothing methods for language models applied to ad hoc information retrieval. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 334-342. ACM, 2001.
13. Freund Y, Schapire R. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1): 119-139, 1997.
14. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: A statistical view of boosting. The Annals of Statistics, 28(2): 337-373, 2000.
15. Schapire R, Singer Y. Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3): 297-336, 1999.
16. Zheng Z, Zha H, Zhang T, Chapelle O, Chen K, Sun G. A general boosting method and its application to learning ranking functions for web search. Advances in Neural Information Processing Systems, 20: 1697-1704, 2007.
17. Hastie T, Tibshirani R, Friedman J, Franklin J. The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer, 27(2): 83-85, 2005.
18. Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley Reading, MA, 1999.
19. Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20(4): 422-446, 2002.
20. Kendall M. A new measure of rank correlation. Biometrika, 30(1-2): 81-93, 1938.
21. Liu T, Xu J, Qin T, Xiong W, Li H. Letor: benchmark dataset for research on learning to rank for information retrieval. Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval, 3-10, 2007.

Appendix. Proofs of Theorem 1 and Lemmas 1-4

Theorem 1. Quasi-*KNN* method can identify no less correct feature's ranking order than linear discovery way.

Proof. In the worst case, quasi-*KNN* method is the same as linear discovery way. That is, if the feature's values are ranked in linear in training set, quasi-*KNN* method simply works like linear discovery way.

However, when feature's ranking order is depicted in curve, quasi-*KNN* gains improvement in recognizing the correct ranking order compared with linear discovery way. Take the ranking order depicted in equation (19) as an example:

$$y = \begin{cases} -x^2 + 3, -5 < x < 0 \\ -0.25x^2 + 3, 0 <= x < 4 \end{cases} \tag{19}$$

where x is the feature value and y represents the ranking order. The larger the value of y, the higher ranking order feature value x gets. (In this paper, we use bigger number to represent higher ranking position.) We suppose that feature values are in homogeneous distribution. We use quasi-*KNN* to calculate the feature's ranking order. We divide the curve into three pieces with lines *y=2* and *y=-1* as executing K intervals in quasi-*KNN* method. Apparently, the center in *Interval 1* is 0.5, the center in *Interval 2* is *1.5*, and the center of *Interval 3* is equal to *-2.5* (see *Figure 9*).

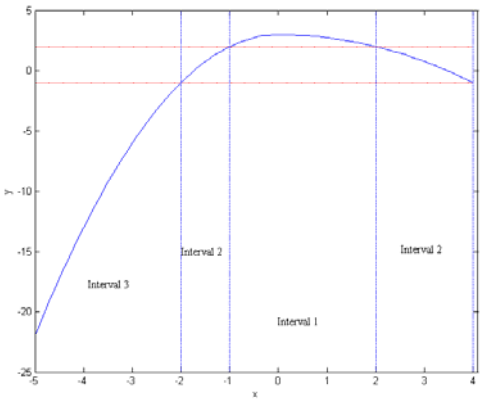


Fig.9 The example for quasi-*KNN* method

According to fundamental *KNN* idea and equation (6), points located in range [*-1,2*] are ranked higher than the points in range [*2,4*] and [*-5,-1*], and points located in range [*-2,-1*] and [*2,4*] are ranked higher than the points in [*-5,-2*]. Then according to equations (7) and (8), all points are arranged as *-1→2,2→4,-1→2,2→-5*, the arrow represents the sequence. Compared with linear feature's ranking discovery, where the points are ranked in *4→-5*, quasi-*KNN* method is more accurate in ranking discovery in this kind of cases.
End of the proof. ■

Lemma 1. In *FeatureRank*, if the loss function is equation (10), and δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend with $h_m\left(\mathbf{x}_m^{(i)}\right)$, weight λ_m can gain the optimum value to make loss value minimum.

$$\lambda_m = \frac{1}{2}\ln\frac{\sum_{i=1}^{N} w_{m-1}^{(i)}\left(1 + E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}{\sum_{i=1}^{N} w_{m-1}^{(i)}\left(1 - E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}, \tag{20}$$

where $w_{m-1}^{(i)} = \exp\left(-E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)\right)$ and

$$\begin{aligned} \delta_i &= E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ &+\lambda_m E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ &-E\left(\pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + \lambda_m h_m\left(\mathbf{x}_m^{(i)}\right)\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \end{aligned} \tag{21}$$

Proof. We want to get λ_m and h_m that satisfy the equation

$$\begin{aligned} (\lambda_m, h_m) &= \arg\min_{\lambda_m, h_m} L\left(\mathbf{y}, h(\mathbf{X})\right) \\ &= \arg\min_{\lambda_m, h_m} \sum_{i=1}^{N} \exp(-E(\pi(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) \\ &\quad + \lambda_m h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right))) \\ &= \arg\min_{\lambda_m, h_m} \sum_{i=1}^{N} \{\left(-E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)\right. \\ &\quad \left. -\lambda_m E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) + \delta_i\right)\} \end{aligned}$$

$$\begin{aligned} \Rightarrow (\lambda_m, h_m) &= \arg\min_{\lambda_m, h_m} \\ &\sum_{i=1}^{N} \{\exp(-E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ &\exp\left(-\lambda_m E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)\right)\exp(\delta_i))\} \end{aligned} \tag{22}$$

We get the optimal function h_m in equation (23), and then calculate the weight λ_m.

$$h_m(\mathbf{x}_m) = \arg\min_{h_m} \sum_{i=1}^{N} \{\exp\left(-E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)\right) \exp\left(-\lambda_m E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)\right) \exp(\delta_i)\} \quad (23)$$

For calculating the weight λ_m, we should separate λ_m from equation (23). According to the property of convex function, we can change equation (23) into

$$\begin{aligned} h_m(\mathbf{x}_m) &= \arg\min_{h_m} J_m \\ &= \arg\min_{h_m} \sum_{i=1}^{N} \{\exp\left(-E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)\right) \\ &\quad \exp(\delta_i)[\frac{1+E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(-\lambda_m) \\ &\quad + \frac{1-E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(\lambda_m)]\} \end{aligned}$$

$$\begin{aligned} \Rightarrow h_m(\mathbf{x}_m) = \arg\min_{h_m} \sum_{i=1}^{N} \{ & w_i \exp(\delta_i) \\ & \left[\frac{1+E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(-\lambda_m)\right. \\ & \left. + \frac{1-E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(\lambda_m)\right]\} \end{aligned} \quad (24)$$

The result of $\exp(\delta_i)$ is bigger than zero permanently. If the result of $\exp(\delta_i)$ is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend with $h_m\left(\mathbf{x}_m^{(i)}\right)$. Equation (24) can be approximate to

$$\begin{aligned} h_m(\mathbf{x}_m) = \arg\min_{f_m} \sum_{i=1}^{N} \{ & w_i \\ & \left[\frac{1+E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(-\lambda_m)\right. \\ & \left. + \frac{1-E\left(h_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right)}{2} \exp(\lambda_m)\right]\} \end{aligned} \quad (25)$$

Thus, we can get the optimum weight λ_m

$$\frac{\partial J_m}{\partial \lambda_m} = 0 \Rightarrow \lambda_m = \frac{1}{2} \ln \frac{\sum_{i=1}^{N} w_{m-1}^{(i)} \left(1 + E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}{\sum_{i=1}^{N} w_{m-1}^{(i)} \left(1 - E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right)}$$

which equals to the conclusion in *Lemma 1*.

End of the proof. ■

Lemma 2. If *FeatureRank* adopts balance control (equation (13)), δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or has the same variation trend as that of $h_m\left(\mathbf{x}_m^{(i)}\right)$.

Proof. We discuss three different situations.

1) If $\hat{h}_m\left(\mathbf{x}_m^{(i)}\right) = h_m\left(\mathbf{x}_m^{(i)}\right)$

In this situation, $\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)$ results from weak ranker, thus it can be approximate to

$$E\left(\pi\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \to 0 \Rightarrow \lambda_m \to 0 .$$

Then we can get δ_i by the following computation

$$\begin{aligned} \delta_i = & E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ & + \lambda_m E\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ & - E\left(\pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + \lambda_m \pi\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)\right)\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \\ \approx & E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right), \pi\left(\mathbf{y}^{(i)}\right)\right) \end{aligned} .$$

Thus we can consider δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$.

2) If $\hat{h}_m\left(\mathbf{x}_m^{(i)}\right) = h_m''\left(\mathbf{x}_m^{(i)}\right)$

In this situation, ranking combination can be converted into the following form:

$$\begin{aligned} & H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + \lambda_m h_m\left(\mathbf{x}_m^{(i)}\right) \\ & = H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + \lambda_m h_m''\left(\mathbf{x}_m^{(i)}\right) \\ & = H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) \\ & \quad + \lambda_m \left(\beta H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + (1-\beta) h_m\left(\mathbf{x}_m^{(i)}\right)\right) \\ & = (1+\beta\lambda_m) H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right) + (1-\beta)\lambda_m h_m\left(\mathbf{x}_m^{(i)}\right) \end{aligned}$$

We can find that the weight values $1+\beta\lambda_m$ and $(1-\beta)\lambda_m$ rely on the value of λ_m:

i. If $\lambda_m > 1/1-2\beta$, $(1-\beta)\lambda_m > 1+\beta\lambda_m$;
ii. If $\lambda_m < 1/1-2\beta$, $(1-\beta)\lambda_m < 1+\beta\lambda_m$.

Generally, $E_{h_m}\left(\mathbf{x}_m^{(i)}\right)$ is less than $E_{H_{m-1}}\left(\mathbf{x}_{m-1}^{(i)}\right)$. This causes $\beta > 1/2$.

Then we discuss if there exists possibility to make $\lambda_m < 0$.

If $\lambda_m < 0$, we can get the inequation:

$$\begin{aligned} & \sum_{i=1}^{N} w_{m-1}^{(i)} \left(1 + E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right) < \sum_{i=1}^{N} w_{m-1}^{(i)} \left(1 - E_{h_m}\left(\mathbf{x}_m^{(i)}\right)\right) \\ & \Rightarrow \sum_{i=1}^{N} w_{m-1}^{(i)} + \sum_{i=1}^{N} w_{m-1}^{(i)} E_{h_m}\left(\mathbf{x}_m^{(i)}\right) \\ & \qquad < \sum_{i=1}^{N} w_{m-1}^{(i)} - \sum_{i=1}^{N} w_{m-1}^{(i)} E_{h_m}\left(\mathbf{x}_m^{(i)}\right) \\ & \Rightarrow \sum_{i=1}^{N} w_{m-1}^{(i)} E_{h_m}\left(\mathbf{x}_m^{(i)}\right) < 0 \end{aligned} .$$

In the inequality, $w_{m-1}^{(i)}$ is always bigger than 0. Consider that if $E_{h_m}\left(\mathbf{x}_m^{(i)}\right)<0$, the weak ranker violates *Definition 1* in *Section 3*. Thus $\lambda_m<0$ is impossible.

According to the limits on λ_m and β, we can get $\delta_i \to \lambda_m E\left(\pi\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)\right),\pi\left(\mathbf{y}^{(i)}\right)\right)$. So δ_i has the same variation trend as that of $h_m\left(\mathbf{x}_m^{(i)}\right)$.

3) If $\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)=H_{m-1}\left(\mathbf{x}_i\right)$

We can calculate value δ_i

$$\begin{aligned}\delta_i &= E\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\ &+\lambda_m E\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\ &-E\left(\pi\left(H_{m-1}\left(\mathbf{x}_{m-1}^{(i)}\right)+\lambda_m\pi\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)\right)\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\\ &=\lambda_m E\left(\pi\left(\hat{h}_m\left(\mathbf{x}_m^{(i)}\right)\right),\pi\left(\mathbf{y}^{(i)}\right)\right)\end{aligned}$$

Thus it is clear that δ_i and $h_m\left(\mathbf{x}_m^{(i)}\right)$ have the same variation trend.

Through the proofs above, we can conclude that δ_i is insensitive to $h_m\left(\mathbf{x}_m^{(i)}\right)$ or they have the same variation trend in balance control.

End of the proof. ■

Lemma 3. In *DiffRank*, the practical ranking error is bounded by

$$\frac{1}{V}\sum\nolimits_{N_q}\sum\nolimits_{i\succ j}[\![1]\!]_{\Delta \mathrm{H}_{ij}<0}\le\prod_{m=1}^{M}Z^m \tag{26}$$

Proof. First, we unfold the distribution $D_{ij}^{(q),m}$ in *DiffRank*

$$\begin{aligned}D_{ij}^{(q),m}&=\frac{D_{ij}^{(q),m-1}\exp\left(-\alpha^m\Delta h_{ij}^{(q),m}/U_q\right)}{Z^m}\\ &=\frac{D_{ij}^{(q),m-2}\exp\left(-\alpha^{m-1}\Delta h_{ij}^{(q),m-1}/U_q\right)\exp\left(-\alpha^m\Delta h_{ij}^{(q),m}/U_q\right)}{Z^{m-1}Z^m}\end{aligned}$$

$$\Rightarrow D_{ij}^{(q),m}=\frac{\exp\left(-\Delta H_{ij}^{(q),m}/U_q\right)}{V\prod_{m=1}^{M}Z^m}. \tag{27}$$

There is a fact that

$$\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m}=1. \tag{28}$$

Let equation (27) substitutes the distribution in equation (28), we can get the equation

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

$$\Rightarrow\sum_{N_q}\sum_{i\succ j}\exp\left(-\Delta H_{ij}^{(q),m}/U_q\right)=V\prod_{m=1}^{M}Z^m \tag{29}$$

Clearly, when $\Delta H_{ij}^{(q),m}\le 0$, $\exp\left(-\Delta H_{ij}^{(q),m}/U_q\right)\ge 1$.

Thus, the inequation holds

$$[\![1]\!]_{\Delta H_{ij}^{(q),m}<0}\le\exp\left(-\Delta H_{ij}^{(q),m}/U_q\right)$$

Extend the situation into all queries in training set:

$$\sum_{N_q}\sum_{i\succ j}[\![1]\!]_{\Delta H_{ij}^{(q),m}<0}\le\sum_{N_q}\sum_{i\succ j}\exp\left(-\Delta H_{ij}^{(q),m}/U_q\right) \tag{30}$$

Let equation (29) substitutes the right side of inequation (30), then we get

$$\sum_{N_q}\sum_{i\succ j}[\![1]\!]_{\Delta H_{ij}^{(q),m}<0}\le V\prod_{m=1}^{M}Z^m$$

$$\Rightarrow\frac{1}{V}\sum_{N_q}\sum_{i\succ j}[\![1]\!]_{\Delta H_{ij}^{(q),m}<0}\le\prod_{m=1}^{M}Z^m$$

Thus *Lemma 3* holds.

End of the proof. ■

Lemma 4. In *DiffRank*, if $\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m}=1$ and $\phi^m\in[0,1]$, for minimizing ranking error in training process, α^m should be assigned to:

$$\alpha^m=\frac{1}{2}\ln\frac{1+\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m-1}\Delta h_{ij}^{(q),m}/U_q}{1-\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m-1}\Delta h_{ij}^{(q),m}/U_q} \tag{31}$$

where

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

Proof. According to *DiffRank* algorithm, Z^m satisfies the equation

$$\begin{aligned}Z^m&=\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m-1}\exp\left(-\alpha^m\Delta h_{ij}^{(q),m}/U_q\right)\\ &\le\sum_{N_q}\sum_{i\succ j}D_{ij}^{(q),m-1}\left(\frac{1+\Delta h_{ij}^{(q),m}/U_q}{2}\exp\left(-\alpha^m\right)\right.\\ &\left.+\frac{1-\Delta h_{ij}^{(q),m}/U_q}{2}\exp\left(\alpha^m\right)\right)\end{aligned}$$

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

$\exp\left(-\alpha^m\right)=\sqrt{\frac{1-\phi^m}{\phi^m}}$ substitute to the above inequation,

$$Z^m \le \phi^m\sqrt{\frac{1-\phi^m}{\phi^m}}+\left(1-\phi^m\right)\sqrt{\frac{\phi^m}{1-\phi^m}}$$

$$\Rightarrow Z^m \le \sqrt{4\phi^m\left(1-\phi^m\right)}$$

$$\Rightarrow Z^m \le \sqrt{1-\left(1-2\phi^m\right)^2}$$

As $\phi^m\in[0,1]$, $\sqrt{1-\left(1-2\phi^m\right)^2}\le 1$. So we have

$$\prod_{k=1}^{m} Z^k \le \prod_{k=1}^{m-1} Z^k$$

In this case, the upper bound of practical ranking error can be continually reduced in the training process. And the weight α_m is computed as

$$\alpha^m=-\ln\sqrt{\frac{1-\phi^m}{\phi^m}}=-\frac{1}{2}\ln\frac{1-\phi^m}{\phi^m}$$

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

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

Thus *Lemma 4* holds.

End of the proof. ■