Feature Ranking for Total Ordering Ranking Problems

Yongqing Wang[1] **Daniel Zeng**[1,2]

[1]Key Laboratory of Complex Systems and Intelligence Science, CAS
Beijing 100190, P. R. China
[2]Department of Management Information Systems, The University of Arizona
Tucson AZ 85721, USA

Abstract

In this paper, we first introduce the development of learning to rank and discuss the problems existing in this field especially the ignorance of total ordering ranking. For dealing with the total ordering ranking problem, we assume a method "feature ranking". Based the assumption, we design two algorithms: Feature Rank and BL-FeatureRank. BL-FeatureRank with balance control is a complementary algorithm of Feature Rank for the purpose of performance improvement and robustness.

Keywords: Total ordering ranking, feature ranking, balance control

1. Introduction

Learning to Rank is aim to use technology of machine learning to get rank function that describes a ranking problem in document retrieval, collaborative filtering, recommendation system, text mining and so on. In terms of distinct aspects to this field, learning to rank has been categorized into three major approaches: Pointwise approach, Pairwise approach and Listwise approach[1][2].

Pointwise approach, Pairwise approach and Listwise approach take care of relationship between item and its importance degree, item and item, and item and instance respectively. In this paper, we call the term "instance" as event that triggers a ranking matter that is goal to arrange items involved in this matter. e.g., in document retrieval, query can be an instance and documents listed in the query can be items. According to the recent research[3], Listwise approach is regarded as the most competitive approach because of instance-item structure. However, there are several shortcomings in Listwise approach.

1. Linear ranking function cannot describe a complex ranking problem in most of real ranking solutions.
2. Although previous work on Listwise approach with linear ranking function has noticed top-k problem[4] , it is still difficult to find a prefect linear ranking function to describe top-k items exactly in total ordering problem (total ordering problem is that every item in an instance has a unique relevance degree representing the importance degree in the instance). The limitation in linear ranking function cannot be resolved even in top-1 issue (in most of practical ranking problems).

In this paper, to resolve these listed problems in Listwise approach, we first suggest that perfect ranking function actually exist in practical ranking problem. Then, we provide "feature rank" that is an assumption(each single feature can depict whole ranking problem in certain degree). According to the assumption, we propose a dividing method to partition practical ranking problem into several subproblems based on each single feature. Then we conduct two algorithms: FeatureRank and Balanced FeatureRank (BL-FeatureRank). Though experiments, we demonstrate the prominent improvement in these new algorithms.

2. Background

2.1. Permutation

In this paper, permutation denoted to π is the map from items to its ranking position. $\pi(i)$ represents the rank given to the element valued i. Ranking result(vector form) is a permutation through mapping.

2.2. Performance measure

Most of previous works on learning to rank take *nDCG* and *MAP* as major performance measures. Some algorithms are devised to directly maximize values in performance measures or their modified forms, such as SVM$^{\text{map}}$ [5], LambdaRank [6], SHF-DCG, REG-SHF-DCG [7], PermuRank [8] and NDCG_Boost [9]. However, it is hardly to employ *nDCG* and *MAP* to total ordering problem. Another performance measure *kendall*-τ posted by Kendall[10] is suitable for this issue, which takes care of the relative ranking position of each item. In this paper, we revise *kendall*-τ little so that the modified *kendall*-τ focuses on the similarity between ranking result and ground truth.

3. Methods

3.1. General framework

First we provide a general description on learning to rank. Let $\mathbf{X}$ be the input space, in which each element denotes an instance. Let $\mathbf{Y}$ the output space, in which each element denotes a relevance degree corresponding to an instance.

Take document retrieval as an example, $\mathbf{x_i} \in \mathbf{X}$ represents one retrieval instance and represents the j-th feature vector in this instance. $\mathbf{y_i}$, which is the ground truth of the corresponding retrieval instance i, represents a list of relevance degree and $\pi(\mathbf{y}_i) \in \pi(\mathbf{Y})$ is the permutation of the list. Thus, the training set can be denotes to $S = (\mathbf{x}_i, \pi(\mathbf{y}_i))_{i=1}^{N}$.

Then, we define $E(f(\mathbf{x}_i), \pi(\mathbf{y}_i)) \in [-1, 1]$ as performance measure function, which assesses the similarity degree between the result from ranking function and permutation of ground truth. In order to evaluate the soundness of the proposed ranking, we choose *kendall*-τ and revise it little for our algorithm.

$$\begin{aligned} E(f(\mathbf{x}_i), \pi(\mathbf{y}_j)) &= 2T(f(\mathbf{x}_i), \\ &\qquad \pi(\mathbf{y}_i)) - 1, \end{aligned}$$

where

$$\begin{aligned} T(\pi, \sigma) &= \frac{1}{z} \sum_{k<l} I\{[\pi_k - \pi_l] \\ &\qquad [\sigma_k - \sigma_l] > 0\} \end{aligned}$$

$I(\cdot)$ is the indicator function, z is the normalization factor and i and j represent the index of permutation.

3.2. KNN for feature ranking

We suggest features can be regarded as direct influential factors to items' importance degree. First we conduct two assumptions.

- *Assumption 1: every single feature can reflect the whole rank-ing problem in certain degree when using clustering method.*
- *Assumption 2: practical ranking function can be divided as the form of polynomial series,*

$$f\left(\mathbf{x_1},\cdots,\mathbf{x_M}\right)\rightarrow\pi\left(\lambda_1 f_1\left(\mathbf{x_1}\right)\right.\\ \left.+\lambda_2 f_2+\cdots+\lambda_M f_M\left(\mathbf{x_M}\right)\right). \tag{1}$$

In equation (1), M is the quantity of features. $\mathbf{x_i}$ represents specific item's feature. $f(\cdot)$ and $f_i(\cdot)$ represent practical ranking function and its divided part based on features respectively.

Based on these two assumptions, the ranking problem is transferred from approximating practical ranking f to discovering feature ranking f_i. For certain specific feature $\mathbf{x_i}$, we choose an instance, get all items listed in the instance and arrange these items by relevance degree in descending order. Then, feature vector can be selected and defined as $\mathbf{fv}=(\langle val_1, val_2,\cdots,val_N\rangle)$, where N is quantity of items. We divide the feature vector $\mathbf{fv}$ into K intervals with equal length. C_l , where the index l is numbered in sequence, is the mean of elements in each interval. We name the step "Feature Dividing". In practical ranking system, Feature Dividing needs to be operated to every instance with the same steps. Finally, $\hat{C}_l$ is set to be the global interval center, which is the mean of each interval center.

- *Theorem 1: the smaller index the interval has indicates the greater relevance degree.*

$$\left\{\begin{array}{c}\{\mathbf{elm}_l | elm_{li}\in Interval_l,\\ i=1,\cdots,|Interval_l|\}\\ \{\mathbf{elm}_k | elm_{kj}\in Interval_k,\\ j=1,\cdots,|Interval_k|\}\\ l<k\end{array}\right.\\ \Rightarrow \mathbf{elm}_{li}\succ\mathbf{elm}_{kj}, \tag{2}$$

where $\mathbf{elm}_l$ *the set of values in* $\mathbf{interval}_l$, val_{li} *is the value which belongs to* i*-th.*

Then, we devise a mechanism for calculating the inner order in each interval.

- *Assumption 3: inner order in each interval depends on the relationship between two adjacent intervals.*

$$g\left(val_{li}, val_{lj}\right)>0\Rightarrow val_{li}\succ val_{lj},$$

where

$$g\left(val_{li}, val_{lj}\right)=\\ \left\{\begin{array}{c}\left(val_{li}-val_{lj}\right)\left(\hat{C}_l-\hat{C}_{l+1}\right)\\ ,l<K\\ \left(val_{li}-val_{lj}\right)\left(\hat{C}_{l-1}-\hat{C}_l\right)\\ ,l=K\end{array}\right.$$

Though theorem 1 and assumption 3, KNN can be applied to feature ranking.

3.3. Combination with boosting idea

Following the general framework and previous work, we revise general boosting method and employ it for computing distribution λ.

We define exponential loss function as the form

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

where performance measure function $E(\cdot)$ ranges from -1 to +1.

Based on assumption 2, practical ranking f is substituted by feature ranking, and the loss function is equivalent to

$$L(\mathbf{y}, f(\mathbf{x})) = \sum_{i=1}^{N} \exp\left(- E \left(\sum_{k=1}^{M} \lambda_k f_k(\mathbf{x}_i), \pi(\mathbf{y}_i) \right) \right).$$

Then, we propose a boosting process for calculating distribution $\lambda = \langle \lambda_1, \lambda_2, \cdots, \lambda_m \rangle$, where

$$\lambda_i = \frac{1}{2} \ln \frac{\sum\limits_{i=1}^{N} w_i^{m-1}\left(1 + E_{f_m}(\mathbf{x}_i)\right)}{\sum\limits_{i=1}^{N} w_i^{m-1}\left(1 - E_{f_m}(\mathbf{x}_i)\right)},$$

$w_i^{m-1} = \exp\left(-E_{h_{m-1}}(\mathbf{x}_i)\right)$ where $h_{m-1}(\mathbf{x}_i) = \sum_{k=1}^{m-1} \lambda_k f_k(\mathbf{x}_i)$ and $E_{f_m}(\mathbf{x}_i)$ is the convenient abbreviation of $E(f_m(\mathbf{x}_i), \pi(\mathbf{y}_i))$.

3.4. Balance control

Based on the fact that performance measure can describe the effectiveness of ranking result directly, we introduce it to monitor added feature ranking and control the boosting pro-cess. First, we revise feature ranking

$$\begin{aligned} f'_m(\mathbf{x}_i) &= E_{h_{m-1}}(\mathbf{x}_i)\, h_{m-1}(\mathbf{x}_i) \\ &+ E_{f_m}(\mathbf{x}_i)\, f_m(\mathbf{x}_i) \end{aligned}$$

Meanwhile, we take performance comparison between $h_{m-1}(\mathbf{x}_i)$ and $f_m(\mathbf{x}_i)$ into consideration. Then feature ranking is revised to

$$\begin{aligned} f''_m(\mathbf{x}_i) &= \beta E_{h_{m-1}}(\mathbf{x}_i)\, h_{m-1}(\mathbf{x}_i) \\ &+ (1-\beta)\, E_{f_m}(\mathbf{x}_i)\, f_m(\mathbf{x}_i), \end{aligned}$$

where $\beta = E_{f_m}(\mathbf{x}_i)/(E_{f_m}(\mathbf{x}_i) + E_{h_{m-1}}(\mathbf{x}_i))$.

At last, we choose the highest value in performance measure among f_m, f'_m and f''_m as the optimal feature ranking $\hat{f}_m$.

The major difference between FeatureRank and BL-FeatureRank is that BL-FeatureRank adopts the modified feature ranking.

3.5. Algorithms

We propose FeatureRank and BL-FeatureRank as Table 1 and Table 2.

Table 1: FeatureRank algorithm

```
Input: $S = \{\mathbf{x}_i, \pi(\mathbf{y}_i)\}_{i=1}^{N}$, and
  parameters $E$, $K$ and $M$.
Initialize $w_0(\mathbf{X}) = \mathbf{1}$
For $m = 1, 2, \cdots, M$
  Generate feature ranking $f_m$
    according to the $m$-th feature
  Computing $\lambda_m$
End for
Output ranking model: $F(\mathbf{X}) =$
  $\pi(\sum_{m=1}^{M} \lambda_m f_m(\mathbf{X}))$
```

Table 2: BL-FeatureRank algorithm

```
Input: $S = \{\mathbf{x}_i, \pi(\mathbf{y}_i)\}_{i=1}^{N}$, and
  parameters $E$, $K$ and $M$.
Initialize $w_0(\mathbf{X}) = \mathbf{1}$
For $t = 1, 2, \cdots, T$
  For $m = 1, 2, \cdots, M$
    Generate feature ranking $\hat{f}_{tm}$
      according to the $m$-th feature
    Computing $\lambda_m$
End for
Output ranking model: $F(\mathbf{X}) =$
  $\pi(\sum t = 1^T \sum_{m=1}^{M} \lambda_{tm} f_{tm}(\mathbf{X}))$
```

4. Experiment

We conduct experiments on the data set MQ2008-list in LETOR 4. The row representing document in MQ2008-list is a 46-dimension feature[11], belongs to specific query and is described by a unique relevance degree.

4.1. Case study: feature ranking discovery

We conduct an experiment to verify *assumption 1* and *assumption 3*. First, we define:

1. normal center. The center of selected global interval is same as the operation in section 3.1.
   Then, we continue to divide each selected interval into three parts and set center for each part:
2. prior center. Prior center gathers one third of elements in an interval and these selected elements are prior than others. The center is the mean of these elements;
3. middle center. Differ with 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: 19-th feature, 21-th feature and 5-th feature for observation and show them on Figure 1 where the horizontal axis represents interval number and vertical axis is interval center value.

We find that,

1. The variation tendency implies single feature can reflect whole ranking problem when using clustering method;
2. All types of global interval center has similar variation tendency;
3. Values of posterior center, middle center and prior center have a descending order when variation tendency in normal center is towards up; If the variation tendency in normal center reverse, so do the order.

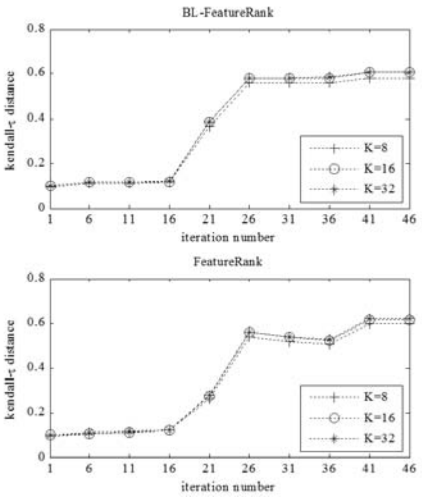

Fig. 2: Performance curve when $K = 8, 16, 32$ in FeatureRank and BL-FeatureRank

According to the three facts, we verify the effectiveness of *assumption 1* and *assumption 3*.

4.2. Robustness of FeatureRank and BL-FeatureRank

We conduct two experiment discuss the robustness of FeatureRank and BL-FeatureRank.

In first experiment, we choose $K = 8$, $K = 16$, $K = 32$ and show the result in Figure 2 where horizontal axis represents iteration number m in boosting process and vertical axis is modified *kendall*-τ that is range from -1 to 1. For the convenience of comparison, we set $T = 1$ in BL-FeatureRank. The higher value shows in vertical axis, the more accuracy reflects in ranking results.

In another experiment, we compare the effectiveness of balance control in

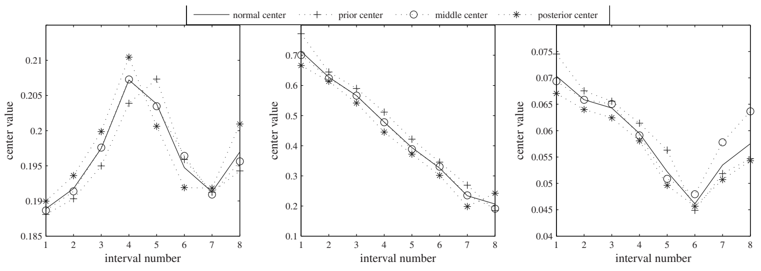

Fig. 1: Four types of interval center when $K = 8$

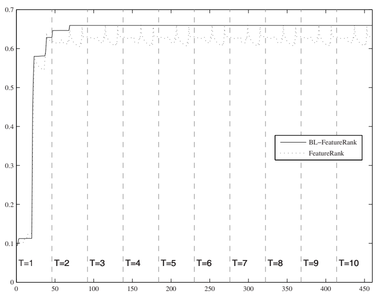

Fig. 3: Robustness comparison between FeatureRank and BL-FeatureRank

BL-FeatureRank. Figure 3 shows the experimental results where horizontal axis represents iteration number $t * m$, vertical axis is the modified *kendall-*τ and dash lines separate the zone by T. We can observe that performance curves in both two algorithms have no great change by the variation of parameter K, although they appear the positive correlation to parameter K and BL-FeatureRank is more stable in performance curve than FeatureRank.

4.3. Comparison

We compare the performance on *kendall-*τ distance among FeatureRank, BL-FeatureRank and ListMLE[12]. ListMLE is one of the current state-of-the-art with linear ranking function in listwise approach. We revise it in order that it can calculate total ordering problem. To convenience, all distances have already been normalized to

$$kendall - \tau : T(\pi, \sigma) = |S|^{-1} \sum_{i \in S} \frac{1}{z_i}$$
$$\sum_{k<l} I\{[\pi_{i,k} - \pi_{i,l}][\sigma_{i,k} - \sigma_{i,l}] < 0\}.$$

Table 3 shows the comparison on data set MQ2008-list. The smaller value denotes less dissimilarity between ground truths and predicted ranking results. We can observe that FeatureRank and BL-FeatureRank have more accurate ranking results than ListMLE, which can prove FeatureRank, BL-FeatureRank notably improve current linear listwise method. Then, balance control is further improved by FeatureRank and the improvement shows on the comparison among FeatureRank $K = 8$, BL-FeatureRank $T = 1$, $K = 8$ and BL-FeatureRank $T = 3, K = 8$.

5. Conclusion

In this paper, we first introduce flaws in current learning to rank approaches. Based on previous work, we proposed

Table 3: Performance on test data MQlist-2008

	ListMLE	FeatureRank			BL-FeatureRank	
		K=8	K=16	K=32	T=1, K=8	T=3, K=8
kendall-τ	0.3013	0.2027	0.1926	0.1917	0.2085	0.1981

new methods FeatureRank and BL-FeatureRank to deal with these existing problems. We provided the detailed information about FeatureRank and BL-FeatureRank, did experiments on feature ranking discovery, robustness test and performance analysis. We prove assumptions by experiments, find the effectiveness of balance control and the advantages of FeatureRank and BL-FeatureRank.

Acknowledgements

This work was supported in part by NNFSC(60875049, 60921061, 70890084, and 71071152), by MOST(2006AA010106), and by CAS(2F07C01).

References

[1] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, volume 227, pages 129 – 136, 2007.

[2] T. Qin, X.D. Zhang, M.F. Tsai, D.S. Wang, T.Y. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing & Management*, 44(2):838–855, 2008.

[3] T.Y. Liu. *Learning to rank for information retrieval.* Now Pub, 2009.

[4] F. Xia, T.Y. Liu, and H. Li. Statistical consistency of top-k ranking. *Advances in Neural Information Processing Systems*, pages 2098–2106, 2009.

[5] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. page 278. ACM, 2007.

[6] Q. Wu, CJC Burges, KM Svore, and J. Gao. Ranking, boosting, and model adaptation. *Tecnical Report, MSR-TR-2008-109*, 2008.

[7] Mingrui Wu, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Smoothing dcg for learning to rank: a novel approach using smoothed hinge functions, 2009.

[8] J. Xu, T.Y. Liu, M. Lu, H. Li, and W.Y. Ma. Directly optimizing evaluation measures in learning to rank. pages 107–114. ACM, 2008.

[9] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. Citeseer, 2010.

[10] M.G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81, 1938.

[11] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. pages 3–10. Citeseer, 2007.

[12] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.