FEATURERANK: A NON-LINEAR LISTWISE APPROACH WITH CLUSTERING AND BOOSTING

Yongqing Wang and Wenji Mao

Key Lab of Complex Systems and Intelligent Science
Institute of Automation, Chinese Academy of Sciences
Beijing, 100190, P. R. China

ABSTRACT

Listwise is an important approach in learning to rank. Most of the existing lisewise methods use a linear ranking function which can only achieve a limited performance being applied to complex ranking problem. This paper proposes a non-linear listwise algorithm inspired by boosting and clustering. Different from the previous listwise approaches, our algorithm constructs weak rankers through directly discovering hidden order in single feature, and then combines these weak rankers using a boosting procedure. To discover the hidden order, we utilize (*KNN*) method. In our preliminary experiment, we compare our approach with other listwise algorithms and show the effectiveness of our proposed algorithm.

***Index Terms*—** Learning to rank, listwise approach

1. INTRODUCTION

Ranking aims to find most useful information according to objects' features, which has been widely used in fields such as document retrieval, information filter, opinion mining, etc. In recent years, machine learning has been applied to ranking, which is called "Learning To Rank". Learning to rank can be classified into three categories: pointwise, pairwise and listwise approaches [1–16].

Pointwise and pairwise approaches, which take ranking problem as regression and classification, neglect the structure of whole ranking order. In contrast, listwise approachaddresses ranking problem directly with the whole structure. Listwise has demonstrated great promise in gaining better performance [3]. However, ranking function in previous listwise methods, for example the typical methods: ListMLE [3], ListNet [1], LambdaRank [4], AdaRank [5], RankCosine [2] and SVM$^{\text{map}}$ [6] are linear which only describes the rank order in two directions, that is, descending or ascending orders. This shortcoming of linear listwise methods has limited the wide adoption of algorithms applications.

In this paper, we propose a novel listwise learning algorithm to overcome the limitation of previous approaches. In our study, we find that single feature reveals hidden order, which can reflect the whole ranking order to certain degree and can be discovered using some clustering methods, such as *KNN*. Based on the hidden order, we design FeatureRank, which takes ranking results from the hidden orders in each single feature, considers them as weak rankers and combines these weak rankers by boosting. The preliminary experimental results illustrate that our algorithm can discover hidden order efficiently through the *KNN* algorithm. Our algorithm also shows significant improvement in comparison with previous listwise method.

This paper is organized as follows. Section 2 introduces the general framework and algorithm. Experimental results are provided in section 3. Finally, section 4 summarizes our concluding remarks and lists some future research directions.

2. METHOD

2.1. General framework

We first provide a basic description on learning to rank. Instance is an event triggering a ranking matter with the aim of arranging items involved in this matter. For example, in document retrieval, a query is an instance and the documents listed in the query are the items. In addition, $\mathbf{X}$ is defined as the input space in which each element denotes an instance, and $\mathbf{Y}$ as the output space in which each element denotes a relevance degree corresponding to an instance.

Take document retrieval as an example. Let $\mathbf{x}_i \in \mathbf{X}$ represent a list of retrieved documents and $\mathbf{x}_{ij} \in \mathbf{x}_i$ represents item (document) vector of the j-th feature in a query (instance). Let $\mathbf{y}_i$ represent a list of relevance degree that is the ground truth of corresponding the query. Let $\pi(\cdot)$ be the permutation function that can convert ranking list to permutation so that $\pi(\mathbf{y}_i) \in \mathbf{Y}$ is the permutation of ranking list of $\mathbf{y}_i$. Based on the above definitions, the training set S can be denoted as $S = \{\mathbf{x}_i, \pi(\mathbf{y}_i)\}_{i=1}^{N}$.

We aim to create a ranking model which has the form

$$F(\mathbf{X}) = \pi\left(\sum_{m=1}^{M} \beta_m h_m(\mathbf{X})\right)$$

978-1-4244-8886-5/10/$26.00 ©2010 IEEE

where $h(\mathbf{X})$ is a weak ranker determined by the hidden order of each single feature; β represents weight of corresponding weak ranker, which indicates the significance of the weak ranker in the whole ranking model; and M is the number of features.

The function $E\left(f\left(\mathbf{x}_i\right), \pi\left(\mathbf{y}_i\right)\right) \in [-1, 1]$ assesses the similarity degree between the result from ranking function f and permutation of ground truth $\mathbf{y}$. In order to evaluate the ranking performance, we choose *kendall*-τ [17] (as better candidate than MAP [12], NDCG [18], as *kendall*-τ can fit our algorithm better than other two methods) as the performance measure. We also use *kendall*-τ measure to compute the similarity between ranking result and ground truth as follow:

$$E\left(f\left(\mathbf{x}_i\right), \pi\left(\mathbf{y}_j\right)\right) = 1 - 2T\left(f\left(\mathbf{x}_i\right), \pi\left(\mathbf{y}_i\right)\right), \qquad (1)$$

where

$$T\left(\pi, \sigma\right) = \frac{1}{z} \sum_{k<l} I\left\{\left[\pi_k - \pi_l\right]\left[\sigma_k - \sigma_l\right] < 0\right\}, \qquad (2)$$

and I is the indicator function; π and σ are permutations representing the ranking result and ground truth. π_k and σ_k are the elements in the k-th position of pemutations. If π_k is prior to π_l, $(\pi_k - \pi_l) > 0$. Otherwise, $(\pi_k - \pi_l) < 0$; and z is the normalization factor, and i and j represent the index of permutation.

2.2. Clustering for discovering hidden order

We employ *KNN* as the clustering technique to find hidden order of each single feature. First, for a specific feature, we choose a query, get item vector corresponding to this feature from input space $\mathbf{X}$. According to the relevance degree, each element in the vector is reassigned in descending order and distributed into K intervals (also called "class" in this paper) with equal length. Then each interval is numbered sequentially from K to 1. We choose the mean of elements as interval center. In practical ranking system, feature dividing needs to be applied to every feature and every instance using the same steps.

The order of elements in each query satisfies equations (3) and (4),

$$\begin{cases} \{\mathbf{val}_l | val_{li} \in Class_l, i = 1, \cdots, |Class_l|\} \\ \{\mathbf{val}_k | val_{kj} \in Class_k, j = 1, \cdots, |Class_k|\} \\ l > k \end{cases} \qquad (3)$$
$$\Rightarrow val_l i \succ val_k j,$$

where $\mathbf{val}_l$ is the set of values in class l, val_{li} is the i-th value which belongs to class l and C_l is the center of class l.

$$g\left(val_{li}, val_{lj}\right) > 0 \Rightarrow val_{li} \succ val_{lj}, \qquad (4)$$

where

$$g\left(val_{li}, val_{lj}\right) = \begin{cases} \left(val_{li} - val_{lj}\right)\left(C_l - C_{l-1}\right), l < K \\ \left(val_{li} - val_{lj}\right)\left(C_{l+1} - C_l\right), l = K \end{cases}.$$

2.3. FeatureRank algorithm

We follow the boosting framework [19] and design an algorithm to combine weak rankers to find hidden order in features respectively (as described in Section 2.2). The algorithm is showed in Table 1.

Table 1. FeatureRank algorithm

```
Input: $S = \{\mathbf{x}_i, \pi(\mathbf{y}_i)\}_{i=1}^{N}$, and parameters $E$, $K$ and $M$
Initialize $\omega_0(\mathbf{X}) = \mathbf{1}$
For $m = 1, 2, \cdots, M$
  Create weak ranker $h_m$ according to the $m$-th feature
  Computing $\beta_m$
      $\beta_m = \frac{1}{2} \ln \frac{\sum_{i=1}^{N} w_{m-1}(\mathbf{x}_i)(1+E(h_m(\mathbf{x}_i),\pi(\mathbf{y}_i)))}{\sum_{i=1}^{N} w_{m-1}(\mathbf{x}_i)(1-E(h_m(\mathbf{x}_i),\pi(\mathbf{y}_i)))}$
  Create $f_m$
      $f_m(\mathbf{X}) = \pi\left(\sum_{k=1}^{m} \beta_k h_k(\mathbf{X})\right)$
  Update $\omega_m$
      $w_m(\mathbf{X}) = \exp(-E(f_m(\mathbf{X}), \mathbf{Y}))$
End for
Output ranking model: $F(\mathbf{X}) = \pi\left(\sum_{m=1}^{M} \beta_m h_m(\mathbf{X})\right)$
```

In FeatureRank, the input is a training set $S = \{\mathbf{x}_i, \pi\left(\mathbf{y}_i\right)\}_{i=1}^{N}$ and the output is the ranking model

$$F\left(\mathbf{X}\right) = \pi\left(\sum_{m=1}^{M} \beta_m h_m\left(\mathbf{X}\right)\right).$$

The algorithm runs M rounds. In each round, the algorithm generates a weak ranker h_m by *KNN* technique and calculates the weight β of each weak ranker in the ranking model.

2.4. Time complexity of FeatureRank

According to the algorithm listed above, we can conduct an analysis on time complexity. In the step of using *KNN* for clustering and discovering hidden order, the time complexity is $T(n + n \log n)$, where n is the maximum modulus of item vector, $T(n)$ is the time complexity for *KNN* and $T(n \log n)$ is the time complexity for assigning inner order in each interval. Then, in boosting step, time complexity becomes $T(M * N * (n + n \log n))$, where M is the number of features and N is the number of instances in the input space. Finally, the complexity of FeatureRank is $O(n \log n)$ if $n \to \infty$. Table 2 shows the time complexity of each listwise algorithms in the best situation.

Table 2. Time complexities of listwise algorithms

Feature Rank	ListMLE	AdaRank [5]
$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

3. EXPERIMENTS AND RESULTS

3.1. The data set

In this paper, we conduct experiments on the standard benchmark for learning to rank data set MQ2008-list of LETOR4.0 [20]. MQ2008-list collects the data from TREC 2008 and assigns each document a unique relevance degree in a query. A large relevance degree means a higher position in permutation. This data set consists of five text files and their combinations. For every document in this data set, the feature vector has 46 dimensions. To facilitate the classification procedure based on single feature, we rearrange the data set by features in descending order according to relevance degree and grouped by the query *id*.

3.2. Experimental illustration of hidden order

We have supposed that hidden order in the feature can be discovered through clustering algorithm. Here, we propose an experiment to find this hidden order. We conduct an experiment on data set that calculates four kinds of class centers for classification to verify the conjecture: 1).Normal center. The selection of class center is the same as that in section 2; 2).Prior center. The difference between prior center and normal center is that prior class center gather first one third of an interval in a query. Then we set the mean of these values as the class center; 3).Middle center. Different from prior center, in the case of middle center, the collected elements are located in the middle of the interval. 4).Posterior center. The collected elements are those in the rear of the interval. We choose a single feature and show these class centers in Fig.1 where the horizontal axis represents the class number and vertical axis is the value of center. We can see from this figure that when the value of the normal center is lower than the following class center, the prior center in current class is below the posterior center and vice versa. This demonstrates that the inner order of the divided class follows the equation (4) in section 2; And for the original order, it is clear that the higher class number has the higher position in the permutation which satisfies equation (3).

3.3. Comparison

We choose ListMLE as a typical linear listwise approach to compare ranking results. For convenience, we execute normalization on permutation of ranking results when calculating Euclidean distance. Table 3 shows the performance on test data of MQlist-2008 with performance measure *kendall*-τ. Smaller value denotes less dissimilarity between ground truths and predicted ranking results. From Table 3, we can see that comparing with ListMLE, FeatureRank has a smaller value on *kendall*-τ than ListMLE, which means FeatureRank has achieved a notable improvement. This is mainly because of the limitation of linear ranking function $f(x)$ adopted in listwise approach which lead the problem of addressing complex ranking problem.

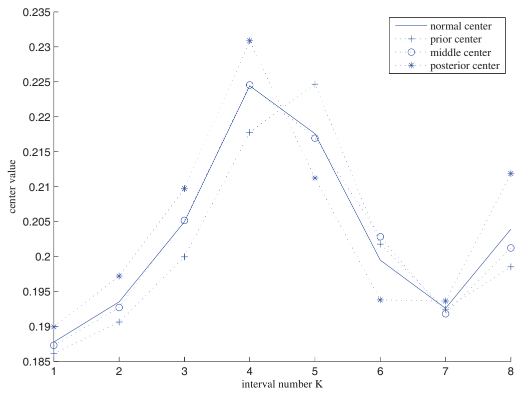

Fig. 1. Four types of class center when $K = 8$ in *KNN*

Table 3. Performance on test data MQlist-2008

	ListMLE	Feature Rank		
		K=8	*K*=16	*K*=32
kendall-τ distance	0.3013	0.2027	0.1926	0.1917

3.4. Robustness of Feature Rank

To measure the influence on the number of divided classes, we choose K=8, K=16, K=32 and show the result in Fig.2, where the horizontal axis represents the iteration number in running process and the vertical axis is the revised *kendall*-τ(see equation (2)), which locates i in interval $[0, 1]$. The higher value *kendall*-τ gets the more accuracy the ranking result is. As indicted by Fig.3, we find that algorithm performance is changed with parameter *K*, and express some positive correlations. However, this kind of positive correlation is reduced by an increase of *K* value. Thus, our algorithm is robust with respect to the changes of *K* and performance on *kendall*-τ decrease its change with *K* rising.

4. CONCLUSION AND FUTURE WORK

In this paper, we propose a non-linear listwise algorithm to address ranking problem. We find that hidden order exists in single feature, and design an algorithm. Based on hidden order, we devise weak rankers for each feature and combined these weak rankers through boosting framework. In addition, the time complexity is $O(n \log n)$ which sustains the advantage of previous listwise approach. In our experiments, we

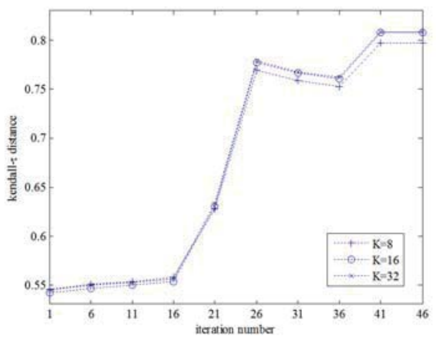

Fig. 2. Performance on iteration when parameter *K*=8, 16, 32 in *KNN*

first design an experiment to illustrate the assumption of hidden order. Then we compare with ListMLE on performance measure *kendall-τ* and design a robustness test on our algorithm. According to experiments, we conclude that our algorithm shows a notable improvement, great effectiveness and robustness.

In our going work, we plan to make comparisons in listwise approaches on data sets which every document in a query has unique relevance degree, and we analyze the performance and effectiveness through experiments.

5. 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*, 2007, vol. 227, pp. 129 – 136.

[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*, vol. 44, no. 2, pp. 838–855, 2008.

[3] 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*. 2008, pp. 1192–1199, ACM.

[4] CJC Burges, R. Ragno, and Q.V. Le, "Learning to rank with nonsmooth cost functions," *Advances in Neural Information Processing Systems*, vol. 19, pp. 193, 2007.

[5] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 2007, p. 398, ACM.

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

[7] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *The Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.

[8] D. Cossock and T. Zhang, "Statistical analysis of bayes optimal subset ranking," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 5140–5154, 2008.

[9] O. Chapelle, Q. Le, and A. Smola, "Large margin optimization of ranking measures," 2007, Citeseer.

[10] K. Crammer and Y. Singer, "Pranking with ranking," *Advances in Neural Information Processing Systems*, vol. 1, pp. 641–648, 2002.

[11] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," 1999, vol. 1, pp. 97–102, Citeseer.

[12] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*, Addison-Wesley Reading, MA, 1999.

[13] L Mason, J Baxter, P Bartlett, and M Frean, "Boosting algorithms as gradient descent in function space," 1999, vol. 12, p. 512C518, Citeseer.

[14] G. Lebanon and J. Lafferty, "Cranking: Combining rankings using conditional probability models on permutations," 2002, pp. 363–370, Citeseer.

[15] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," 2005, p. 96, ACM.

[16] P. Li, C. Burges, Q. Wu, J.C. Platt, D. Koller, Y. Singer, and S. Roweis, "Mcrank: Learning to rank using multiple classification and gradient boosting," *Advances in Neural Information Processing Systems*, 2007.

[17] M.G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81, 1938.

[18] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 446, 2002.

[19] J. Friedman, T. Hastie, and R. Tibshirani, "Special invited paper. additive logistic regression: A statistical view of boosting," *The annals of statistics*, vol. 28, no. 2, pp. 337–374, 2000.

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