Listwise Learning to Rank by Exploring Structure of Objects

Ou Wu, Qiang You, Xue Mao, Fen Xia, Fei Yuan, Weiming Hu

Abstract—Listwise learning to rank (LTR) is aimed at constructing a ranking model from listwise training data to order objects. In most existing studies, each training instance consists of a set of objects described by preference features. In a preference feature space for the objects in training, the structure of the objects is associated with the absolute preference degrees for the objects. The degrees significantly influence the ordering of the objects. Nevertheless, the structure of the training objects in their preference feature space has rarely been studied. In addition, most listwise LTR algorithms yield a single linear ranking model for all objects, but this ranking model may be insufficient to capture the underlying nonlinear ranking mechanism among all objects. This study proposes a divide-and-train method to learn a nonlinear ranking model from listwise training data. First, a rank-preserving clustering approach is used to infer the structure of objects in their preference feature space and all the objects in training data are divided into several clusters. Each cluster is assumed to correspond to a preference degree and an ordinal regression function is then learned. Second, considering that relations exist among the clusters, a multi-task listwise ranking approach is then employed to train linear ranking functions for all the clusters (or preference degrees) simultaneously. Our proposed method utilizes both the (relative) preferences among objects and the intrinsic structure of objects. Experimental results on benchmark data sets suggest that the proposed method outperforms state-of-the-art listwise LTR algorithms.

Index Terms—Listwise learning to rank; clustering; multi-task learning; structure

1 Introduction

Learning to rank (LTR) has received great attention in recent years as it is useful in many applications, such as information retrieval, data mining, natural language processing, and speech recognition [1]. Existing LTR methods are grouped into three approaches: pointwise, pairwise, and listwise [6]. Different approaches define different input/output spaces of training data and employ different loss functions. In pointwise LTR, the input space contains the preference features of each object. In pairwise LTR, the input space usually contains the relative preference features between a pair of objects. In listwise LTR, the input space contains a set of objects each of which is described by the preference features. The listwise approach models the ranking problem in a more natural way than the other two approaches, and previous experiments demonstrate the competitive performance of this approach on benchmark data sets [2]. Therefore, this study focuses on the listwise approach in which the training set consists of instances of objects and ranked object lists on each instance.

A number of listwise LTR algorithms have been proposed in previous literature [2, 3, 4, 8, 9, 11, 14, 24]. They can be categorized into two streams. The first stream defines a loss function over a ground-truth ordering and a predicted ordering. A ranking function is then trained according to the minimization of training loss. Two representative methods are ListMLE [2] and ListNet [3]. The former defines a listwise likelihood loss function based on the Packett-Luce Model [2]. The latter defines a loss function based on KL-divergence between two permutation probability distributions. Some other methods include FocusedBoost [19], ListReg [20], and DCMP [21] based on different loss definitions. The second stream attempts to directly maximize the information retrieval performance measures (such as mean average precision (MAP)) in training. Some representative methods include AdaRank [4], ApproxNDCG [22], and SmoothRank [23].

The target ranking models in most listwise ranking studies are linear. There are a limited number of studies that construct nonlinear listwise ranking models which are mainly based on the decision tree. Pavlov et al. [24] combined a sequence of boosted tree as a ranking model based on the bag strategy. Moon et al. [25] also constructed the ranking model based on the combination of weak learners (i.e., decision trees). These ranking algorithms still require the relevance labels during training, although they are claimed to be available for the listwise setting. Some other studies focus on feature selection [14], semi-supervised ranking [15], and cost-sensitive ranking [18] in a slightly different listwise LTR setting.

Existing listwise LTR algorithms require further investigation due to the following reasons

(1) *The structure of objects in the preference feature space is ignored*. Although the listwise approach still pursues a ranking model whose input contains the preference features of an object, it does not directly explore the preference feature space in learning. The structure of objects in the preference feature space reflects absolute preference degrees for objects. In most ranking problems, the absolute preference degrees for objects exist regardless of whether they are given. In information retrieval, a document can be considered highly relevant, relevant, or irrel-

• *Ou Wu (corresponding author), Qiang You, Xue Mao, Fei Yuan, Weiming Hu are with the Institute of Automation, Chinese Academy of Sciences. E-mail: {wuou, qyou, xue.mao, fyuan, wmhu}@nlpr.ia.ac.cn. Fen Xia is with the Big data Laboratory, Baidu Inc., E-mail: xiafen@baidu.com.*

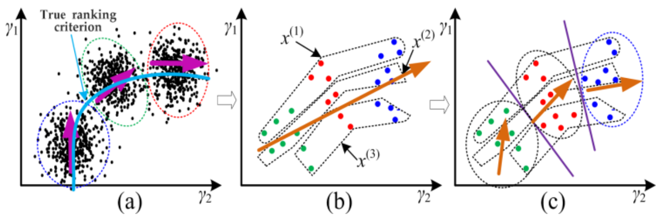


Fig. 1. (a) Objects in preference feature space and underlying ranking criterion (the curve); (b) three ranked lists ($x^{(1)}$, $x^{(2)}$, $x^{(3)}$) and learned ranking criterion (orange arrow) by listMLE; (c) learned ranking criterion (purple partition lines and orange arrows) by considering object structure.

evant to a given query.

(2) *A linear ranking criterion is usually considered*. In algorithms such as ListMLE and ListNet, a single linear ranking function is learned on the basis of input training data. However, the ranking mechanisms for many real data by humans are not globally linear. For example, the ranking criterion between two images with high visual quality may rely on the content, while that between an image with high visual quality and another image with low visual quality may rely only on the resolution. Nonlinear ranking functions should be capable of significantly improving the performances of baseline algorithms [6].

Fig. 1 presents an illustrative example for the above analysis for existing studies. Fig. 1(a) shows a set of objects and the underlying ranking criterion (the blue curve) in the preference feature space. The blue curve can be approximately fitted by three piecewise line segments in Fig. 1(a). Each line segment corresponds to an object cluster (denoted by the dotted ellipses). In a viewpoint of pointwise LTR, all the objects can be divided into three clusters corresponding to three preference degrees `highly relevant', `relevant', and `irrelevant'. The ranking criteria of the three line segments are $f(\bullet) = \gamma_1$, $f(\bullet) = \gamma_1 - \gamma_2$, and $f(\bullet) = \gamma_2$, respectively. Fig. 1(b) shows three instances (ranked object lists). Accordingly, the learned ranking criterion (the orange arrow) by using the current listwise approach is $f(\bullet) = \gamma_1 - 0.8\gamma_2$, which is inappropriate to rank objects from the `highly relevant' and `irrelevant' clusters. Fig. 1(c) shows that if object structure is considered and the three clusters are separately processed, the right ranking criterion for each cluster can be obtained. The purple lines indicate the global partition criterion.

Fig. 1 shows that the utilization of object structure in listwise LTR can benefit the ranking criterion learning. Nevertheless, the object structure (or the preference degrees) is usually unattainable. To better utilize the underlying object structure in a preference feature space, this work introduces a divide-and-train listwise LTR method. In the *dividing* step, a rank-preserving clustering (RPC) approach is performed to divide the objects in all training instances into different clusters with orders. The objects in each obtained cluster are assumed to contain the same preference degree. An ordinal regression algorithm is used to train an ordinal regression function based on the ordered clusters. In the *training* step, a multi-task listwise LTR approach is performed to train multiple linear ranking functions for all clusters simultaneously. The obtained ranking model consists of an ordinal regression function and multiple linear ranking functions. Our work is also inspired by recent studies on local linear SVM [26] which utilizes a set of local linear SVM models to replace a nonlinear SVM achieved by the kernel trick. The proposed method offers several advantages:

(1) The object structure in the preference feature space is explored. In the dividing step, all the involved objects are partitioned into clusters in which the ranks among objects are preserved.

(2) The learned ranking model can be nonlinear. The learned model consists of a global ordinal regression function and multiple local linear ranking functions, which is not necessarily linear.

(3) The learned ranking model can predict both the absolute preference degrees of input objects and the full ranked lists of the objects.

The rest paper is organized as follows. Section 2 introduces the proposed method. Section 3 introduces our experimental evaluations. Section 4 concludes the work.

2 The Proposed Method

The notations used in this paper are firstly introduced. Let X be the input space whose instances are sets of objects which are described by preference features, and Y be the output space whose elements are orderings of objects in an instance. A training instance $x^{(i)}$ is represented by $(x_1^{(i)}, \ldots, x_{ni}^{(i)})$, where ni denotes the number of objects in $x^{(i)}$, and an ordering label $y^{(i)} \in Y$ on $x^{(i)}$ is represented by $(y_1^{(i)}, \cdots, y_{ni}^{(i)})$, where $y_j^{(i)}$ is the rank assigned to the object $x_j^{(i)}$. There is no relevance score for $x^{(i)}$ in the training set. Additionally, another space is included, namely, the preference feature space whose elements are objects described by preference features. In existing listwise LTR work, the preference feature space is not referred.

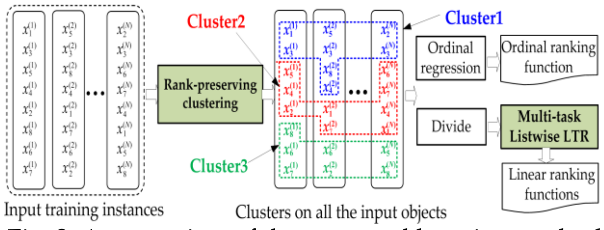
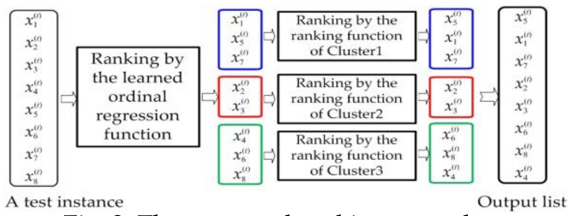


Fig. 2. An overview of the proposed learning method.

The proposed method first divides all the objects into clusters corresponding to underlying preference degrees, and then trains an ordinal regression function among the clusters and a linear ranking function for each cluster. The overview of our method is illustrated in Fig. 2. The two key components are RPC and multi-task listwise LTR. The ranking for new input instances with the learned ordinal regression function and multiple linear ranking functions is shown in Fig. 3. The learned ordinal regression function divides all the objects in a test instance into different degrees. The objects in each degree are then ranked by using the corresponding linear ranking function for that degree. The following subsections introduce RPC and multi-task listwise LTR.



$x_1^{(i)}$ $x_2^{(i)}$ $x_3^{(i)}$ $x_4^{(i)}$ $x_5^{(i)}$ $x_6^{(i)}$ $x_7^{(i)}$ $x_8^{(i)}$
Ranking by the learned ordinal regression function
$x_1^{(i)}$ $x_5^{(i)}$ $x_7^{(i)}$
$x_2^{(i)}$ $x_3^{(i)}$
$x_4^{(i)}$ $x_6^{(i)}$ $x_8^{(i)}$
Ranking by the ranking function of Cluster1
Ranking by the ranking function of Cluster2
Ranking by the ranking function of Cluster3
$x_5^{(i)}$ $x_1^{(i)}$ $x_7^{(i)}$
$x_2^{(i)}$ $x_3^{(i)}$
$x_6^{(i)}$ $x_8^{(i)}$ $x_4^{(i)}$
$x_5^{(i)}$ $x_1^{(i)}$ $x_7^{(i)}$ $x_2^{(i)}$ $x_3^{(i)}$ $x_6^{(i)}$ $x_8^{(i)}$ $x_4^{(i)}$
A test instance
Output list

Fig. 3. The proposed ranking procedure.

2.1 Rank-preserving Clustering

In conventional clustering, the achieved clusters reflect the inherent structure of data and no order exists. In RPC, the obtained clusters are considered to be associated with preference degrees and thus order exists. RPC aims to partition the objects in all training instances into different parts, and the rank relationships among objects in each instance are preserved. Existing algorithms cannot be directly used. Therefore, a two-stage approach is proposed. First, to better utilize the ranking information, metric learning is used to learn a new metric. Second, a heuristic clustering algorithm is proposed based on localsearch [12].

2.1.1 Metric Learning based on Ranking Lists

Different from conventional clustering, RPC has ranks among objects in an instance. Rank information can be used to aid clustering. In this work, metric learning is adopted to utilize rank information. In metric learning, the distance metric is in the following form:

$$d_M\left(x_j^{(i)}, x_k^{(i)}\right) = \sqrt{(x_j^{(i)} - x_k^{(i)})^T M (x_j^{(i)} - x_k^{(i)})} \tag{1}$$

where M is required to be a positive semi-definite matrix. When M is restricted to be diagonal, this corresponds to learning a metric in which the different dimensions are assigned different weights. Current metric learning for clustering usually applies additional constraints to construct two data sets, namely, similar set S, and dissimilar set D [13，16, 17]. No obvious constraints are provided in RPC. Therefore, certain constraints should be established based on the rank information among objects.

Intuitively, a small position gap between two objects in an instance indicates a high possibility that the objects belong to the same cluster. Two sets S_r and D_r are constructed accordingly:

$$\begin{aligned} S_r &= \{(x_j^{(i)}, x_k^{(i)}) \mid \left|y_j^{(i)} - y_k^{(i)}\right| \le \kappa_1,\ i = 1,\ \dots,\ N,\ \ j < k\} \\ D_r &= \{(x_j^{(i)}, x_k^{(i)}) \mid \left|y_j^{(i)} - y_k^{(i)}\right| \ge \kappa_2,\ i = 1,\ \dots,\ N,\ \ j < k\} \end{aligned} \tag{2}$$

where κ_1 and κ_2 are two thresholds. With the two sets, the optimization problem defined in [13] is constructed:

$$\begin{aligned} &\min_M \sum\nolimits_{(x_j^{(i)}, x_k^{(i)}) \in S_r} d_M^2\left(x_j^{(i)}, x_k^{(i)}\right) \\ &s.t.\ \ \sum\nolimits_{(x_p^{(i)}, x_q^{(i)}) \in D_r} d_M\left(x_p^{(i)}, x_q^{(i)}\right) \ge 1, \quad M \ge 0 \end{aligned} \tag{3}$$

The above optimization problem is convex and can be solved using gradient descent and iterative projections.

2.1.2 Clustering based on Localsearch

RPC aims at obtaining K clusters and a low cluster index indicates the objects are ranked high in the cluster. Let C be clusters and $\Gamma = \{\mu_k\}$, where μ_k is the center of the kth cluster. Based on k-means, we first define

$$\Omega_M(C, \Gamma) = \sum\nolimits_{k=1}^{K} \sum_{x_j^{(i)} \in x^{(i)}} \sum_{x_j^{(i)} \in C_k} \| x_j^{(i)} - \mu_k \|_M^2 \tag{4}$$

To preserve ranks among objects in each instance, a new optimization problem can be constructed:

$$\begin{aligned} &\arg\min_{C,\,\Gamma} \Omega_M(C, \Gamma) \\ &s.t.,\quad \mathrm{I}(x_{[i:j]}^{(i)}) \le \mathrm{I}(x_{[i:j+1]}^{(i)}) \\ &\qquad i = 1, \dots, N;\ \ j = 1, \dots, n^{(i)} - 1 \end{aligned} \tag{5}$$

where $\mathrm{I}(\cdot)$ represents the cluster index of an object and the notation [*i*:*j*] is the index of the object in the *j*th rank, i.e., $y_{[i:j]}^{(i)} = j$. The constraints in (5) are used to preserve the ranks of objects in all the training instances. Using (5), the obtained clusters correspond to preference degrees, and the objects in a cluster with a lower index (i.e., a higher preference degree) have higher rankings than those in a cluster with a higher index.

Solving (5) directly is difficult. Instead, a heuristic method is used based on the localsearch clustering algorithm. Localsearch is an application of a local-search heuristics [12] to the correlation clustering problem. An initial clustering is given. It then goes through all the objects, and considers each object for placement in a different cluster (maybe a new singleton cluster with the considered object). If one placement can obtain a negative moving cost, then the object is placed in the cluster that yields the minimum moving cost. In our work, localsearch is modified to preserve the rankings for each instance.

An initial clustering is given first. It then goes through all the instances, and considers in each instance the objects whose adjacent object belongs to a different cluster index. These objects in an instance consist of the candidate set for the instance. Informally, the candidate set of an instance $x^{(i)}$ is defined as follows:

$$\begin{aligned} Cs^{(i)} = \ &\{x_{[i:l]}^{(i)} \mid \mathrm{I}(x_{[i:l]}^{(i)}) \ne \mathrm{I}(x_{[i:l+1]}^{(i)}) \text{ or } \mathrm{I}(x_{[i:l]}^{(i)}) \ne \mathrm{I}(x_{[i:l-1]}^{(i)}), \\ &l = 1,\ \dots,\ |x^{(i)}|\} \end{aligned} \tag{6}$$

The costs of placing the objects in $Cs^{(i)}$ into the cluster, which the objects' adjacent belongs to, are calculated. If one placement can achieve moving the minimum negative cost, the object is placed in the cluster that yields the minimum moving cost. This algorithm is called *pLocalsearch* for simplicity.

An illustrated example is used to demonstrate how the placement and costs are operated in an iteration of pLocalsearch. Assuming that an instance $x^{(i)}$ and the cluster indexes of all the objects in $x^{(i)}$ are as follows:

$$\begin{aligned} &x^{(i)} = (x_{[i:1]}^{(i)},\ x_{[i:2]}^{(i)},\ x_{[i:3]}^{(i)},\ x_{[i:4]}^{(i)},\ x_{[i:5]}^{(i)},\ x_{[i:6]}^{(i)},\ x_{[i:7]}^{(i)},\ x_{]i:8]}^{(i)}) \\ &C^{(i)} = (1,\ 1,\ 2,\ 2,\ 2,\ 2,\ 3,\ 3) \end{aligned} \tag{7}$$

Then $Cs^{(i)} = \{x_{[i:2]}^{(i)},\ x_{[i:3]}^{(i)},\ x_{[i:6]}^{(i)},\ x_{[i:7]}^{(i)}\}$. The costs of placing each object in $Cs^{(i)}$ into a different cluster are calculated. Take the object $x_{[i:2]}^{(i)}$ as an example. The current cluster index of this object is 1. The cost of placing this object into cluster '2' is calculated. The placing costs for the rest objects are calculated and the object with the minimum cost is selected. Let k_1 and k_2 indicate two clusters. The placing cost is defined as follows:

$$cost_M(x^{(i)}_{[i:j]}, k_1 \to k_2) = \frac{n(k_2)}{n(k_2)+1} d^2_M(x^{(i)}_{[i:j]}, \mu_{k_2}) - \frac{n(k_1)}{n(k_1)-1} d^2_M(x^{(i)}_{[i:j]}, \mu_{k_1}) \quad (8)$$

After one iteration is performed, a new clustering is obtained. Assuming that for $x^{(i)}$ in an iteration, an object in $x^{(i)}$ ($x^{(i)}_*$) is placed from cluster k_1 to k_2. We obtain

$$\Omega_M(C', \Gamma') - \Omega_M(C, \Gamma) = cost_M(x^{(i)}_*, k_1 \to k_2) < 0 \quad (9)$$

which indicates that in each placement, the value of the objective function in (5) decreases.

In the above pLocalsearch iterations, the ranks of all the objects are preserved. As shown in Fig. 2, when all the objects are partitioned into clusters with orders, an ordinal regression method is used to train an ordinal regression function. This step appears to be a learning approach from listwise to pointwise.

2.2 Multi-task Listwise LTR

When all the objects are divided into clusters, new training subsets can be constructed based on the original training instances and clusters. The new training subsets are used to learn linear ranking functions for all the clusters. Considering that the learning tasks for a ranking function for each training subset are similar and correlated, an MTL strategy is used to achieve all the functions. Learning multiple related tasks simultaneously has been shown to significantly improve the performance relative to learning independently [10].

MTL has two major technical lines. Considering that the number of clusters in our work is not large, the mean regularization-based strategy proposed by Evgeniou and Pontil [10] is adopted. Let **W** = {w_1, …, w_K}, where w_k represents the linear ranking function of the kth cluster. The optimization function for **W** is

$$\min_{\mathbf{W}} \Psi_C(\mathbf{W}) = \min_{\mathbf{W}} \sum_{k=1}^{K} \sum_{i=1}^{|C_k|} -\log P(y'^{(i)} \mid x'^{(i)}, w_k) + \lambda \| w_k - \frac{1}{K}\sum_{l=1}^{K} w_l \|_2^2 \quad (10)$$

where –log($P(y'^{(i)} \mid x'^{(i)}, w_k)$) is the likelihood loss calculated by the Packett-Luce model [2], and λ is a trade-off parameter. The regularization term of (10) penalizes the functions that are significantly different from other functions. The solution of (10) is obtained by iteratively operating:

$$\bar{\mathbf{w}}^{(t)} = \frac{1}{K}\sum_{l=1}^{K} w^{(t)}_l$$
$$\mathbf{W}^{(t+1)} = \arg\min_{\mathbf{W}} \sum_{k=1}^{K} \sum_{i=1}^{|C_k|} -\log P(y'^{(i)} \mid x'^{(i)}, w_k) + \lambda \| w_k - \bar{\mathbf{w}}^{(t)} \|_2^2 \quad (11)$$

To minimize (11), stochastic gradient descend (SGD) can be used for each function parameter w_k. The objective value of (10) is decreased at each iteration of (11). The proof is omitted due to lack of space.

2.3 Algorithmic Steps

The previous subsections introduce the key components of the proposed method. The proposed method integrates RPC and multi-task listwise LTR, and is thus called RPC-MTL for brevity. In summary, RPC-MTL seeks an ordinal regression function $\boldsymbol{f}_{odr}$ and a set of multiple linear ranking functions (parameterized by **W)** corresponding to K clusters. The algorithmic steps are presented in Algorithm 1. When K equals 1, RPC-MTL is reduced to the listwise LTR algorithm ListMLE because there is only one cluster.

Algorithm 1 RPC-MTL

```
Input: $\{x^{(i)}, y^{(i)}\}$, $i = 1, ..., N$, $\mathbf{W}^{(0)}$, $K$, $t$ = 1, and MaxT
Output: An ordinal regression function $f_{odr}$ and W
Initial: All the objects in each instance are placed into K
clusters according to their positions.
Steps:
    1. Learn a new metric matrix M by optimizing (3).
    2. Perform pLocalsearch to cluster all the objects:
       While (t < MaxT or ~convergence)
         2.1 Construct the candidate set $Cs^{(i)}$ based on (6) .
         2.2 Search the object in $Cs^{(i)}$ which has the minimum and negative moving cost for each instance. If the minimum moving costs for all instances are non-negative, break.
         2.3 Place the object searched in Step 2.2 for each instance into its adjacent cluster; t= t + 1.
    3. Learn an ordinal regression function $f_{odr}$ based on the obtained clusters (The relevance score for the training object $x^{(i)}_j$ is set as K-I( $x^{(i)}_j$ )).
    4. Learn multiple linear ranking functions (W) by iteratively performing the two steps in (11) using the training subsets on the obtained clusters.
```

2.4 Computational Complexity

The computational complexity of the solving of (3) in metric learning is $O(n_d^2)$ [13], where n_d is the feature dimension of the training objects. Let n be the number of all involved objects. The computational complexity of localsearch is $O(n^2) + O(C^2n)$ [12], where $O(n^2)$ is for the calculation of the pairwise distances for all involved objects. Nevertheless, the computational complexity of pLocalsearch is significantly lower than that of localsearch because it only calculates the pairwise distances for all the adjacent objects. The complexity of pLocalsearch is $O[(C^2 + 1)n]$. Assume that the average number of iterations for the solving of (11) is T_1 and the average number of SGD iterations in each iteration is T_2, the computational complexity of the multi-task listwise LTR is approximate $O(C^*T_1^*T_2^*n^*n_d)$. The total computational complexity is approximate $O[n_d^2 + (C^2 + 1)n + C^*T_1^*T_2^*n^* n_d]$.

3 Experiments

Because our study focuses on the standard setting of listwise LTR in which relevance labels are unavailable during training, this section compares the proposed method RPC-MTL against two classical linear LTR algorithms including ListMLE and ListNet. Three data sets are used. The first two are the MQ2007-list and MQ2008-list data sets used in various listwise LTR studies. The third is a data set for ranking the visual quality of images.

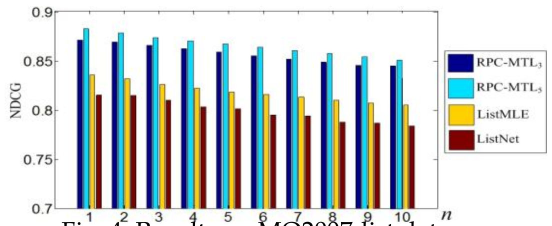


Fig. 4. Results on MQ2007-list data.

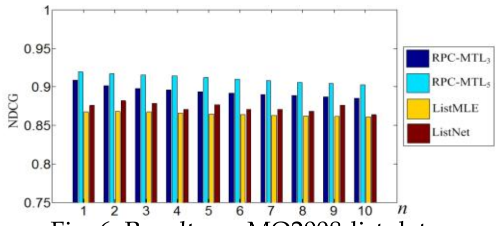


Fig. 6. Results on MQ2008-list data.

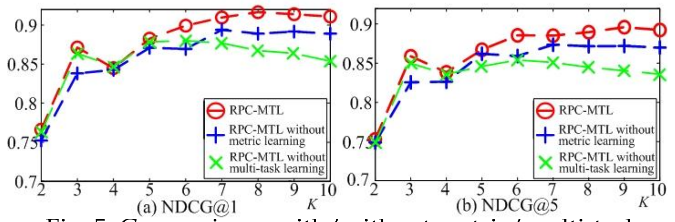


Fig. 5. Comparisons with/without metric/multi-task learning on MQ2007-list.

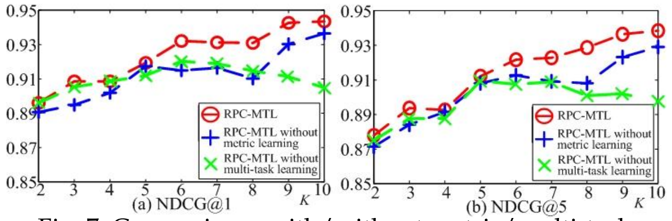


Fig. 7. Comparisons with/without metric/multi-task learning on MQ2008-list.

In all the experiments, the parameters of our proposed method are set as follows. In metric learning, κ_1 should be small and κ_2 should be large. They are heuristically set to 2 and 5, respectively. In RPC, the range K is determined according to the average objects in the instances in each experimental data set; the value of MaxT is set to 1000. In multi-task listwise learning, the value of λ is searched in [0.001, 0.01, 0.1, 1, 2] based on validation data. For MQ2007-list and MQ2008-list, the five-fold settings compiled by LETOR package [5] are adopted. Finally, the normalized discounted cumulative gain (NDCG) [4] is calculated. When this metrics has higher values, improved results are achieved.

3.1 Results on MQ2007-list and MQ2008-listData

Both MQ2007-list and MQ2008-list are also provided by the LETOR package [6]. These two data sets are compiled based on Gov2 Web page collection and two query sets from Million Query track of TREC 2007 and TREC 2008. There are about 1700 queries in MQ2007-list with ranked documents and about 800 queries in MQ2008-list with ranked documents. There are 46-dimensional features for each query-document pair. The 5-fold cross validation strategy is adopted and the 5-fold partition setting in LETOR is followed. In each fold, there are three subsets for learning: training, validation and testing. On both data sets, the relevance scores are not provided. Therefore, to calculate NDCG, the normalized ranks are used as the relevance scores which are defined as follows. For the instance $x^{(i)}$, the relevance score of the document in $x^{(i)}$, whose rank is k, is defined as $(|x^{(i)}|-k)/(|x^{(i)}|-1)$. The scores are only used for NDCG calculation.

Figs. 4 and 5 are the results on MQ2007-list data. Figs. 6 and 7 are the results on MQ2008-list data. In both Figs. 4 and 6, to compare RPC-MTL with ListMLE and ListNet more comprehensively, the value of K in RPC is set to 3 and 5. Two concrete methods are then obtained, namely, RPC-MTL$_3$ and RPC-MTL$_5$. The values of NDCG@n (n = 1, 2, …, 10) are displayed. By conducting the t-test, both RPC-MTL$_3$ and RPC-MTL$_5$ outperform ListMLE and ListNet (p-value < 0.01) significantly. To evaluate whether the introduced metric learning and multi-task learning are useful and to test the robustness of RPC-MTL in terms of K, we compare the values of NDCG@1 and 5 under different K values when RPC-MTL uses or does not use the metric learning/multi-task learning introduced in Section 2.1. When K increases, the performance of RPC-MTL increases and becomes stable when K > 4. The partial reasons lies in that more piecewise-linear functions may construct more complex nonlinear functions. According the experimental results, K can be set to 6 in real applications. When the learned metric is used, RPC-MTL can obtain higher NDCG values according to both Figs. 5 and 7 (p-value < 0.05). When K is larger than 5, RPC-MTL outperforms RPC-MTL without multi-task learning significantly (p-value < 0.01). Both metric learning and multi-task learning do improve the learning performance.

3.2 Results on Image Visual Quality Ranking

Automated evaluation of the visual quality of photos can facilitate next-generation image retrieval [7]. In Web image search, for example, the visual quality of an image can be incorporated into ranking so that the most relevant and best looking photos can be returned. Existing studies focus on the construction of a model to score or classify the visual quality of images. The current study considers the construction of a visual quality ranking model that can be directly used in Web image search.

The third dataset was compiled by Datta et al. [7] based on public data from photo.net. This dataset consists of 3581 photos, with each photo described by 58 features and associated with an average visual quality score rated by photo.net users. The average visual quality score ranges in [3 7], and a higher score indicates higher visual quality. Based on the photos and average scores, a new image visual quality ranking data set is compiled as follow: every 15 photos take turns being selected from all the 3581 photos to form an instance, and the average scores for each of the 15 photos are used to obtain the photo ranking label for the 15 photos. In total, 238 instances are obtained, with each instance containing 15 photos and an ordering label. The 238 instances are divided into 5 folds. In each experimental run, two folds take turns being selected as the test and validation sets, and the rest are used for training. The experimental run is repeated 20 times and the average results are recorded.

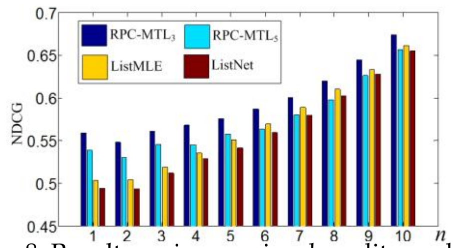


Fig. 8. Results on image visual quality ranking. data.

The values of NDCG@n (n = 1, 2, …, 10) of the competing algorithms are displayed in Fig. 8. With the help of t-test, in terms of NDCG, RPC-MTL$_3$ achieves significantly better results than ListMLE and ListNet (p-value < 0.01). RPC-MTL$_5$ outperforms ListMLE and ListNet when $n \le 4$ (p-value < 0.01), and achieves comparable results when n> 4. The comparison between RPC-MTLs with and without using metric/multi-task learning is also similar to the observations of the experiments on the former two data sets. The corresponding figure results are omitted.

3.3 Discussion

The experimental results indicate that RPC-MTL outperforms ListMLE and ListNet. The main reason is that RPC-MTL utilizes more useful information that is usually ignored in existing algorithms. Compared with existing algorithms, there are two additional components in RPC-MTL: metric learning and object partition based on RPC. The metric learning explores the full ranking information in the training instances to assign distinct weights to different raw features, whereas existing algorithms such as ListMLE and ListNet take all the raw features equally. RPC explores the intrinsic structural information for objects in all training instances, whereas existing algorithms do not consider the relationships among objects in different training instances. In addition, ListMLE is a special case of RPC-MTL when K is set to 1, which guarantees that RPC-MTL should not be inferior to ListMLE. In many real applications, it is difficult to obtain full-ordering ground-truth labels. The proposed method can also be easily to be extended for top-k listwise ranking applications.

4 Conclusions

This study has proposed a divide-and-train listwise LTR method RPC-MTL. The new method employs an RPC algorithm to divide involved objects into clusters. Each cluster in the preference feature space is assumed to correspond to a preference degree. Metric learning is used to pursue new metrics in order to improve RPC. A multi-task listwise LTR procedure is used to train linear ranking functions for each cluster. The learned ranking model then consists of an ordinal regression function and multiple linear ranking functions, which are combined to achieve orders for objects in a test instance. Experimental results indicate the effectiveness of the proposed method.

Acknowledgements

This work is supported by NSFC (61379098). The first and second authors contributed equally.

References

[1] T. Qin, et al., "Global Ranking Using Continuous Conditional Random Fields", Proc. NIPS, pp. 1212-1218, 2008.
[2] F. Xia, et al., "Listwise Approach to Learning to Rank - Theory and Algorithm", Proc. ICML, pp. 1192-1199, 2008.
[3] Z. Cao, et al., "Learning to Rank: from Pairwise Approach to Listwise Approach", Proc. ICML, pp. 129-136, 2007.
[4] J. Xu and H. Li, "Adarank: A Boosting Algorithm for Information Retrieval", Proc. SIGIR, pp. 391–398, 2007.
[5] J. Xu, et al., "Direct Optimization of Evaluation Measures in Learning to Rank", Proc. SIGIR, pp. 107-114, 2008.
[6] T.-Y. Liu, "Learning to Rank for Information Retrieval", Springer, ISBN 978-3-642-14266-6, I-XVII, 1-285, 2011.
[7] R. Datta, et al., "Algorithmic Inferencing of Aesthetics and Emotion in Natural Images: An Exposition", Proc. IEEE ICIP, pp. 105-108, 2008.
[8] X. Geng, et al., "A Noise-tolerant Graphical Model for Ranking", Information Process Management, 48(2), pp. 374-383, 2012.
[9] X.-Q. Cheng, et al., "Ranking on Data Manifold with Sink Points", IEEE TKDE, Vol. 25, No. 1, pp. 177-191, 2013.
[10] T. Evgeniou and M. Pontil, "Regularized Multi-task Learning", Proc. ACM SIGKDD, pp. 109–117, 2004.
[11] Y. Lan, et al., "Statistical Consistency of Ranking Methods in a Rank-Differentiable Probability Space", NIPS, 1241-1249, 2012.
[12] O. Wu, et. al, "Efficient Clustering Aggregation based on Data Fragments", IEEE TSMC, Part B, 42(3), pp. 913-926, 2012.
[13] E. Xing, et. al, "Distance Metric Learning with Application to Clustering with Side-information", NIPS, pp. 505-512, 2003.
[14] Y. Wang, et al., "Listwise Approaches based on Feature Ranking Discovery", Front. Comput. Sci., 6(6), pp. 647–659, 2012.
[15] Y. Yang, et al., "A Multimedia Retrieval Framework based on Semi-Supervised Ranking and Relevance Feedback", IEEE TPAMI, 34(4), pp. 723-742, 2012.
[16] F. Nie, et al., "Clustering and Projected Clustering with Adaptive Neighbors", Proc. ACM SIGKDD, pp. 977-986, 2014.
[17] S. Xiang, et al., "Learning a Mahalanobis Distance Metric for Data Clustering and Classification", Pattern Recognition, vol. 41, no. 12, pp. 3600-3612, 2008.
[18] J. Xu, et al., "Cost-Sensitive Learning of SVM for Ranking", Proc. ECML, pp. 833-840, 2006.
[19] S. Niu, et al., "Top-k learning to rank: Labeling, ranking and evaluation", Proc. SIGIR, pp. 751–760, 2012.
[20] J. Wu, et al., "Learning to rank using query-level regression", Proc. SIGIR, pp. 1091–1092, 2011.
[21] C. Renjifo and C. Carmen, "The discounted cumulative margin penalty: Rank-learning with a list-wise loss and pair-wise margins". Proc. IEEE MLSP, pp. 1–6, 2012.
[22] T. Qin, et al., "A general approximation framework for direct optimization of information retrieval measures", Information Retrieval, 13(4), pp. 375-397, 2010.
[23] O. Chapelle and M. Wu, "Gradient descent optimization of smoothed information retrieval metrics", Information Retrieval, 13(3), pp. 216–235, 2010.
[24] D. Y. Pavlov, et al., "BagBoo: A scalable hybrid bagging-the-boosting model", Proc. CIKM, pp. 1897–1900, 2010.
[25] T. Moon, et al., "IntervalRank: Isotonic regression with listwise and pairwise constraints", Proc. WSDM, pp. 151–160, 2010.
[26] Q. Gu and J. Han, "Clustered support vector machines", Proc. AISTATS, pp. 307–315, 2013.