Journal of Computing and Electronic Information Management

ISSN: 2413-1660

Cost-Sensitive Learning for Listwise Ranking Functions

Haijiang He
Department of Computer Science and Technology, Changsha University, Changsha, Hunan, P. R. China
haijianghe@sohu.com

Abstract: Learning to rank, whose goal is to effectively construct the ranking model, is an important issue for Information Retrieval. Many methods for learning to rank have been introduced, among these algorithms, the listwise approach usually achieves the best results. However, previous works on the listwise approach have not considered two factors: (1) sufficient position discount in loss function, (2) the number of objects attached to the same query. In this paper, a cost-sensitive listwise ranking algorithm is presented, in which these factors are taken into account. The optimization object of our method is strongly convex minimization problem without any constraints, thus a fast Newton algorithm can be used to acquire the unique optimal solution in the problem. The effectiveness of the proposed method is demonstrated by experimental results on the benchmark dataset LETOR.

Keywords: information retrieval, cost-sensitive, learning to rank, listwise, likelihood loss, position discount

1. Introduction

Learning to rank (ranking algorithm) is a central task of Information Retrieval. In applications such as product recommendation, Question Answering [1], and Web search, ranking functions are employed to assess the relevance degree of objects.

When concerned with learning to rank, without loss of generality, we take document retrieval and ranking as example. Recently, much thought and research has been placed on the development of ranking algorithm. In general, existing learning to rank can be divided into three categories: the pointwise approach [2-3], the pairwise approach [4-5], and the listwise approach [6-7]. The listwise approach considers both position information and the documents difference associated with a query [8].

Therefore the listwise approach has intrinsic advantages as compared to the pointwise approach and the pairwise approach.

The ListMLE [7] is an effective and efficient listwise approach, and it introduces log likelihood of two document permutations as listwise loss. However, there is no sufficient position information in the loss function when applying the ListMLE. In addition, for a given query, loss of the ListMLE is sensitive to the number of attached documents. The ranking model of the ListMLE will bias toward queries having large amounts of documents. Combing cost-sensitive learning technique [9] with the listwise ranking algorithm is a solution for dealing with the problems above.

For the pairwise approach, some studies have addressed the use of cost-sensitive learning [10-12]. However, up to our knowledge, existing listwise ranking algorithms have not taken cost-sensitive learning into account. In this paper, we propose a new listwise approach combined with cost-sensitive learning. Specially, we set different losses for mis-ranked documents having different relevance grades, and listwise loss for a given query having a large number of documents is appropriately degraded.

2. The ListMLE and Its Limitations

Assume that we have a collection of *m* labeled queries, denoted by $\{q^{(i)}\}_{i=1}^{m}$. For each query $q^{(i)}$, we have a collection of *l(i)* documents $d^{(i)} = \{d_j^{(i)}\}_{j=1}^{l(i)}$, whose relevance levels or scores to $q^{(i)}$ are given by a vector $y^{(i)} = \{y_j^{(i)}\}_{j=1}^{l(i)}$, where $y_j^{(i)} \in R^1$. Each document $d_j^{(i)}$ that denotes the *j*-th document of $d^{(i)}$ is represented as a feature vector $x_j^{(i)} = \Phi(q^{(i)}, d_j^{(i)}) \in R^D$, where $\Phi(\cdot)$ is a feature mapping function, and *D* is the dimension size of feature vector. The training set of learning to rank can be denoted as $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$. Let ***f(d,q)*** be the ranking function that takes a document-query pair (*d,q*) and output a real number value. For the list of feature vectors $x^{(i)}$, a list of scores $z^{(i)} = \{z_1^{(i)}, z_2^{(i)}, ..., z_{l(i)}^{(i)})\}$ is obtained by applying a ranking function ***f***, where $z_j^{(i)} = f(x_j^{(i)}) \in R^1$. The goal of the listwise approach is to minimize the total losses with respect to the ranked list of documents attached to a query.

$$\sum_{i=1}^{m} L(y^{(i)}, f(q^{(i)}, d^{(i)})) \tag{1}$$

Where ***L*** is a listwise loss function. For the ListMLE [7], $L(y^{(i)}, f(x^{(i)})) = -\log P(y^{(i)} \mid x^{(i)}; f)$, and the likelihood between $y^{(i)}$ and $f(x^{(i)})$ is defined as:

$$P(y^{(i)} \mid x^{(i)}; f) = \prod_{j=1}^{l(i)} \frac{\exp(f(x_{y(j)}^{(i)}))}{\sum_{k=j}^{l(i)} \exp(f(x_{y(k)}^{(i)}))} \tag{2}$$

For $q^{(i)}$, suppose that $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_{l(i)}\}$ is prefect permutation of $y^{(i)}$, where $\Omega_j \in Z^1$ is the identity of one document in $d^{(i)}$. That's to say, $y_{\Omega_1}^{(i)} \geq y_{\Omega_2}^{(i)} \geq ... \geq y_{\Omega_{l(i)}}^{(i)}$. Let $x_{y(j)}^{(i)} = \Phi(q^{(i)}, d_{\Omega_j}^{(i)})$ in Equation (2) be the Ω_j–th element of $x^{(i)}$.

As we have said before, there are two factors one must consider. First, there is insufficient position information in loss function of the ListMLE. For most Information Retrieval systems, the user only browses documents on the top of the result list. So a larger penalty should be added to mis-ranked documents having high relevance degrees. Second, we can prove that the loss of the ListMLE is sensitive to the number of documents attached to the same query, given the following theorem.

Theorem 2.1. For the n objects $\boldsymbol{x}=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{s}=\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$ is any permutation of the score list to $\boldsymbol{x}$ by any ranking function $\boldsymbol{f}$. Suppose that $\boldsymbol{x}'=\boldsymbol{x}+\{a\}$, and $\boldsymbol{s}'=\boldsymbol{s}+\{\boldsymbol{f}(a)\}$, we have $P(s \mid x; f) > P(s' \mid x'; f)$.

Proof: $P(s \mid x; f) = \frac{\exp(s_1)}{\sum_{j=1}^{n} \exp(s_j)} * \frac{\exp(s_2)}{\sum_{j=2}^{n} \exp(s_j)} * \ldots * \frac{\exp(s_n)}{\exp(s_n)}$

$$P(s' \mid x'; f) = \frac{\exp(s_1)}{\sum_{j=1}^{n} \exp(s_j) + \exp(f(a))} * \frac{\exp(s_2)}{\sum_{j=2}^{n} \exp(s_j) + \exp(f(a))} * \ldots * \frac{\exp(s_n)}{\exp(s_n) + \exp(f(a))}$$

It is obvious that Theorem 2.1 holds. Even through $\boldsymbol{f}(a)$ is small than anyone in $\boldsymbol{s}$, the negative log likelihood loss of permutation $\boldsymbol{s}'$ is larger than that of permutation $\boldsymbol{s}$.

3. Cost-Sensitive Listwise Ranking Algorithm

To deal with above problems, we add relevance degree parameter U and documents collection parameter V to adjust the listwise loss. In addition, for making ranking models more robust, we append a regularization item to the loss function. A cost-sensitive listwise ranking algorithm is proposed in this paper, we refer to it as cs-RgList. The optimal problem of the cs-RgList is defined as:

$$\min_{w} \Re(w) = \frac{1}{2} w^2 + \frac{C}{m} \sum_{i=1}^{m} \sum_{j=1}^{l(i)} \frac{U(i,j)}{V(i,j)} * (-\log \frac{\exp(f(x_{y(j)}^{(i)}))}{\sum_{k=j}^{l(i)} \exp(f(x_{y(k)}^{(i)}))}) \tag{3}$$

Where $U(i,j) = pCf^{y_j^{(i)}}$ (pCf >=1 is penalty coefficient in terms of mis-ranked documents), $V(i,j)$ denotes the number of documents whose relevance levels are

equals to $y_j^{(i)}$ in query $q(i)$. C (C>0) is a tradeoff factor between training error and the regularization term. The value of each term in Equation (3) is larger than zero, thus this problem is a strongly convex minimization problem without any constraints. Obviously, there exists a unique solution in the problem. Moreover, the objective function of the cs-RgList is twice differentiable, thus we can use a fast Newton method to solve this problem.

For simplicity, we use a linear combination of model parameters in this paper, i.e. $f(x_j^{(i)}) = w \otimes x_j^{(i)}$, where $w \in R^D$ is a model parameter vector, and $\otimes$ denotes the inner product. The gradient of Equation (3) concerning $\boldsymbol{w}$ can be computed as follows:

$$\frac{\partial \Re(w)}{\partial w} = w + \frac{C}{m}\sum_{i=1}^{m}\sum_{j=1}^{l(i)} U(i,j) * \left(\frac{\sum_{k=j}^{l(i)} x_{y(k)}^{(i)} * \exp(w \otimes x_{y(k)}^{(i)})}{\sum_{k=j}^{l(i)} \exp(w \otimes x_{y(k)}^{(i)})} - x_{y(j)}^{(i)}\right) \tag{4}$$

The second derivative of Equation (3) is as follows:

$$\frac{\partial^2 \Re(w)}{\partial w^2} = 1 + \frac{C}{m}\sum_{i=1}^{m}\sum_{j=1}^{l(i)} U(i,j) * \frac{\sum_{k=j}^{l(i)} (x_{y(k)}^{(i)})^2 * \sum_{k=j}^{l(i)} \exp(w \otimes x_{y(k)}^{(i)}) - (\sum_{k=j}^{l(i)} x_{y(k)}^{(i)} \exp(w \otimes x_{y(k)}^{(i)}))^2}{(\sum_{k=j}^{l(i)} \exp(w \otimes x_{y(k)}^{(i)}))^2} \tag{5}$$

The detail steps of optimization strategy for the cs-RgList are listed as follows:

Step 1. Inputting training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, the gradient tolerance ε, the maximum iteration T, the penalty coefficient pCf, and the tradeoff factor C; initialing linear model $\boldsymbol{w}$ as $\{1.0/m\}$.

Step 2. Calculating the gradient with Equation (4), and calculating the second derivative with Equation (5).

Step 3. Obtaining the steepest descent direction $dirn$ by Cholesky

$$\frac{\partial^2 \Re(w)}{\partial w^2} * dirn = -\frac{\partial \Re(w)}{\partial w}$$

Step 4. Updating $\boldsymbol{w} = \boldsymbol{w} + dirn$.

Step 5. If change of 1-norm $dirn$ is below the gradient tolerance or iterations has exceeded T, outputting the model parameter $\boldsymbol{w}$; otherwise, going Step 2.

4. Experiments and Discussions

The experiments in the paper are conducted on the learning to rank benchmark datasets LETOR [13]. All datasets of LETOR have packaged the extracted features, queries, and relevance judgments. We primarily use seven datasets of LETOR 3.0 and two datasets of LETOR 4.0. The former consists of OHSUMED, topic distillation (TD)

2003 and 2004, named page finding (NP) 2003 and 2004, home page finding (HP) 2003 and 2004. The latter consists of MQ2007 and MQ2008. There are three grades of relevance judgment in OHSUMED, MQ2007, and MQ2008: definitely relevant, possibility relevant and not relevant, their corresponding numerical levels are 2, 1, and 0, respectively. However, the relevance grades of documents in the rest of datasets are judged on two grades: relevant and irrelevant, their corresponding numerical levels are 1 and 0, respectively. With the purpose of fulfilling cross validation, LETOR has divided each dataset into five parts. For each fold, LETOR uses three parts for training, one part for validation, and the rest for testing. For the cs-RgList and the ListMLE, validation set was employed to adjust the parameters. The experimental results reported in this paper are five-fold cross validation results.

Many evaluation metrics using multi-level relevance judgment have emerged in Information Retrieval community. Among these metrics, Normalized Discount Cumulative Gain (NDCG) [14] is the most popular. The NDCG value at position *n* of a ranked list is calculated by the following equation:

$$NDCG@n = \frac{1}{Z_n}\sum_{j=1}^{n}\frac{2^{r(j)}-1}{\log_2(j+1)}$$

Where *r*(*j*) is the level assigned to the *j*-th document of the ranked list, and *Zn* is a normalized constant that is chosen so that a perfect ordering of the result will receive NDCG score of 1.0. Fox example, if one document in OHSUMED dataset is judged as definitely relevant, then its *r*(*j*) takes value of 2, and *U*(.,*j*) in Equation (3) takes *pCf** *pCf*. In general, the user wish to make sure that the top portion of the document list is correctly retrieved. Thus, when choosing the parameters to determine final ranking model, we apply $\mathrm{AvgNDCG} = \frac{1}{10}\sum_{j=1}^{10} NDCG@j$ to measure the performance over validation set.

We first describe the parameter settings, and then show the experimental results and discussions. The tolerance rate and the learning rate are two important parameters of the ListMLE. For the ListMLE, we manually fixed the learning rate at 1e-3 in our empirical study, and its effectiveness had been verified in the experiments. And we ran a grid search over different tolerance rates {1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3}. For the cs-RgList, the gradient tolerance and the maximum iteration are trivial. In most case, the optimization procedure converged in 3~5 iterations. The gradient tolerance was varied from 1e-3 to 1e-6, the cs-RgList exhibited the same performance. Therefore, for the cs-RgList, the gradient tolerance was fixed at 1e-4, and the maximum iteration *T* was fixed at 20. We used two stages strategy to choose the tradeoff factor *C*. First, we ran a grid search over different *C* {1e-4, 1e-3, 0.01, 0.1, 1}, and let *C** be the current optimal *C*. Then, the final *C* was selected from 0.6*C**, 0.8*C**, *C**, 1.2*C**, and 1.4*C**. In the experiments, the penalty coefficient *pCf* was chosen from 1~6.

A set of experiments was developed for the cs-RgList with different *pCf*. In summary, the cs-RgList performed always better than the ListMLE no matter which *pCf* was used in the cs-RgList. In Table 1, the results of AvgNDCG on ListMLE are compared with the results from cs-RgList when fixing *pCf* at 3. From the results, we can see that the performance of cs-RgList is stationary better than that of ListMLE. It is worth noting that, in general, with large *pCf* the performance tends to better as well. Table 2 gives the comparison of applying different *pCf* to cs-RgList on TD2003. However, increasing *pCf* will not improve the performance much when conducting on some datasets. In fact, large *pCf* may be subject to over-fitting the noise documents.

Table 1. The results of AvgNDCG from ListMLE and cs-RgList when fixing *pCf* at 3

Dataset	OHSUMED	MQ 2007	MQ 2008	HP 2003	NP 2003	TD 2003	HP 2004	NP 2004	TD 2004
ListMLE	0.451	0.335	0.38	0.639	0.672	0.251	0.642	0.662	0.225
cs-RgList	0.473	0.4	0.438	0.756	0.692	0.298	0.656	0.7	0.261
improve	4.9%	19.4%	15.3%	18.3%	2.9%	18.7%	2.2%	5.8%	16.1%

Table 2. Comparison results on TD2003 over different *pCf*

NDCG	AvgNDCG	NDCG@1	NDCG@3	NDCG@5	NDCG@7	NDCG@9
1	0.26	0.18	0.275	0.273	0.269	0.267
2	0.294	0.26	0.3	0.304	0.294	0.29
3	0.298	0.28	0.306	0.303	0.3	0.292
4	0.324	0.36	0.323	0.322	0.318	0.315
5	0.33	0.36	0.325	0.333	0.325	0.316
6	0.34	0.38	0.342	0.341	0.331	0.326

The relative AvgNDCG improvements of cs-RgList over ListMLE are mild when the experiments are performed on OHSUMED, NP2003, HP2004, and NP2004. For OHSUMED, this is likely due to the problem of over-fitting. For the other datasets, we deeply examined their construction. There are only 1.01, 1.08, and 1.03 relevant documents per query in NP2003, HP2004, and NP2004, respectively. Even through top *k* NDCG was improved, AvgNDCG was made thinner by averaging 10 NDCGs. Table 3 gives detail results of comparing the cs-RgList with the ListMLE on NP2004, where *pCf* is fixed at 3. As seen in Table 3, significant improvement can be observed on NDCG@1, while improvement on AvgNDCG is small. Considering there are only 1.03 relevant

documents per query in NP2004, we can say that the cs-RgList significantly outperforms the ListMLE on this dataset.

5. Conclusion

Position discount in loss function and the number of documents attached to a query are two factors influencing the performance of the listwise approach to learning to rank. In this paper, we present a cost-sensitive listwise ranking algorithm for flexibly combining these two factors. Experiments show that with the proposed algorithm, we are able to find ranking models with better performance, compared to the ListMLE.

Table 3. Comparison results on NP2004 when fixing *pCf* at 3

NDCG	AvgNDCG	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5
ListMLE	0.662	0.493	0.611	0.648	0.682	0.69
cs-RgList	0.7	0.573	0.657	0.681	0.71	0.717
improve	5.8%	16.3%	7.6%	5.1%	4.1%	4%
NDCG	NDCG@6	NDCG@7	NDCG@8	NDCG@9	NDCG@10	
ListMLE	0.697	0.697	0.697	0.701	0.701	
cs-RgList	0.725	0.73	0.734	0.738	0.738	
improve	4%	4.7%	5.3%	5.2%	5.2%	

Acknowledgements

This paper was financially supported by funding for science and technology project of Hunan Province (Project No. **2015GK3071**).

References

[1] S. Verberne, H. van Halteren, D. Theijssen, S. Raaijmakers, and L. Boves, “Learning to rank QA data”, Proceedings of the Learning to Rank Workshop at SIGIR 2009, Boston, USA, 2009, p41-48

[2] P. Li, C. J. C. Burges, and Q. Wu, “McRank: learning to rank using multiple classification and gradient boosting”, Advances in Neural Information Processing Systems 2007, Vancouver, Canada, 2007, p845-852

[3] W. Chu and S. S. Keerthi,” Support vector ordinal regression”, Neural Computation, Vol. 19(3), 2007, p792-815

[4] Z. Sun, T. Qin, Q. Tao, and J. Wang, “Robust sparse rank learning for non-smooth ranking measures”, Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, USA, 2009, p259-266

[5] N. Usunier, D. Buffoni, and P. Gallinari, “Ranking with ordered weighted pairwise classification”, Proceedings of the 26th International Conference on Machine Learning, Montreal, Canada, 2009, p1057-1064

[6] M. N. Volkovs and R. S. Zemel, “BoltzRank: learning to maximize expected ranking gain”, Proceedings of the 26th International Conference on Machine Learning, Montreal, Canada, 2009, p1089-1096

[7] F. Xia, T.-Y. Liu, J. Wang, and H. Li, “Listwise approach to learning to rank: theory and algorithm”, Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008, p1192-1199

[8] T. -Y. Liu, “Learning to rank for information retrieval”, Foundation and Trends on Information Retrieval, Vol. 3(3), 2009, p225-331

[9] H. B. He and E. A. Garcia, “Learning from imbalanced data”, IEEE Transactions on Knowledge and Data Engineering, Vol. 21(9), 2009, p1263-1284

[10] J. Xu, Y. B. Cao, H. Li, and Y. L. Huang, “Cost-sensitive learning of SVM for ranking”, Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006, p833-840

[11] H. -J. He, “Smooth ranking support vector machine adapting to Web retrieval”, Moshi Shibie yu Rengong Zhineng/Pattern Recognition and Artificial Intelligence, Vol. 22(6), 2009, p891-897

[12] Y. B. Cao, J. Xu, T.-Y. Liu, H. Li, Y. L. Huang, and H.-W. Hon, “Adapting ranking SVM to document retrieval”, Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, 2006, p186-193

[13] T. Qin, T.-Y. Liu, J. Xu, and H. Li, “LETOR: a benchmark collection for research on learning to rank for information retrieval”, Information Retrieval, Vol. 13(4), 2010, p 346-374

[14] K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of IR techniques”, ACM Transactions on Information Systems, Vol. 20(4), 2002, p422-226

[15] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, “Quality versus efficiency in document scoring with learning-to-rank models“, Information Processing and Management. Vol.52, 2016, p 1161–1177

[16] Rodrigo Tripodi Calumby, Marcos André Gonçalves, Ricardoda Silva Torres, “On interactive learning-to-rank for IR: Overview, recent advances, challenges, and directions”, Neurocomputing, Vol.208, 2016, p3–24