Cost-Sensitive Listwise Ranking Approach

Min Lu[1], MaoQiang Xie[2,⋆], Yang Wang[1], Jie Liu[1], and YaLou Huang[1,2]

[1] College of Information Technology Science, Nankai University, Tianjin, China
[2] College of Software, Nankai University, Tianjin, China
{lumin,wangyang022,jliu}@mail.nankai.edu.cn,
{xiemq,huangyl}@nankai.edu.cn

Abstract. This paper addresses listwise approaches in learning to rank for Information Retrieval(IR). The listwise losses are built on the probability of ranking a document highest among the documents set. The probability treats all the documents equally. However, the documents with higher ranks should be emphasized in IR where the ranking order on the top of the ranked list is crucial. In this paper, we establish a framework for cost-sensitive listwise approaches. The framework redefines the probability by imposing weights for the documents. The framework reduces the task of weighting the documents to the task of weighting the document pairs. The weights of the document pairs are computed based on Normalized Discounted Cumulative Gain(NDCG). It is proven that the losses of cost-sensitive listwise approaches are the upper bound of the NDCG loss. As an example, we propose a cost-sensitive ListMLE method. Empirical results shows the advantage of the proposed method.

1 Introduction

Learning to rank is a popular research area due to its widespread applications. This paper focuses on document retrieval. When learning to rank has been applied to document retrieval, it aims to learn a real-valued ranking function that induces a ranking order over the documents set of a query.

The existing ranking approaches can be summarized into three categories: pointwise, pairwise and listwise. The pointwise and pairwise methods[1,2] transform ranking problem into regression or classification problem. The listwise approaches[3,4,5] minimize the loss functions defined between the ranked list and the ground truth list. Theoretical analysis about the properties of the listwise loss functions are also conducted. Fen Xia[3] studied the consistency and soundness of the listwise loss functions, and Yanyan Lan[6] investigated the generalization bound of the listwise loss functions based on Rademacher Averages.

Many listwise algorithms are developed, but no research is conducted to elaborate the common characteristic of the listwise loss functions. What is more, the listwise losses are inadequate for IR where the high performance of the top documents in the ranked list is preferred, measure by NDCG. In this paper, we propose a framework for cost-sensitive listwise approaches. The cost-sensitive

⋆ Corresponding author.

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 358–366, 2010.

listwise loss functions are constructed based on the documents with weights. The framework reduces the task of setting weights for the documents to the task of setting weights for the document pairs. The weights of the document pairs are computed based on the NDCG. As an example, we develop a cost-sensitive ListMLE algorithm. Experimental results show that the proposed method outperforms ListMLE[3], RankCosine[5], AdaRank[7] and Ranking SVM[2].

2 Normalized Discount Cumulative Gain (NDCG)

NDCG[8] evaluates the performance of the top documents in the ranked list. Suppose n candidate documents are retrieved for a query. Each candidate document d_i is represented as a pair $(\boldsymbol{x_i}, y_i)$, where $\boldsymbol{x_i}$ denotes the query-document pair feature vector and y_i is the relevance level. The NDCG@k is defined as:

$$\text{NDCG@k} = \frac{DCG_\pi @k}{DCG_g @k} = \frac{\sum_{j=1}^{n} \frac{2^{y_j}-1}{\log_2(1+\pi(j))} I[\pi(j) \le k]}{\sum_{j=1}^{n} \frac{2^{y_j}-1}{\log_2(1+g(j))} I[g(j) \le k]} \tag{1}$$

where k denotes the truncation level. π is the ranked list generated by a ranking function. g denotes the ground truth list obtained in the document relevance level descending order. $g(j)$ and $\pi(j)$ denote the ground truth ranking position and the ranked position of the document d_j respectively. $I[x]$ yields one if x is true and zero otherwise. Assume that the ranking function is f, then the ranked position $\pi(j)$ is computed from:

$$\pi(j) = 1 + \sum_{i=1}^{n} I[f(\boldsymbol{x_i}) > f(\boldsymbol{x_j})] \tag{2}$$

where $f(\boldsymbol{x_i})$ denotes the rank score of the document d_i. If there are many documents sharing the same relevance level with the document d_j, it is impossible to state the definite value of $g(j)$. So the approximate value $\widehat{g}(j)$ is given.

$$\widehat{g}(j) = 1 + \sum_{i=1}^{n} I[y_i > y_j] \tag{3}$$

It is obvious that $\widehat{g}(j) \le g(j)$ for each j , which implies the inequality.

$$DCG_g @k \le DCG_{\widehat{g}} @k \tag{4}$$

Then the NDCG@k loss, abbreviated $L_{ndcg@k}$, has a upper bound.

$$L_{ndcg@k} = 1 - NDCG@k \le \frac{1}{DCG_{\widehat{g}} @k} \left(DCG_{\widehat{g}} @k - DCG_\pi @k\right) \tag{5}$$

3 Listwise Approach

3.1 Existing Listwise Loss Function

The listwise approaches take documents lists as an instance, and minimize the loss functions defined between the ranked list and the ground truth list. Typical methods include ListNet[4], RankCosine[5] and ListMLE[3]. We review their loss functions. For the sake of describing simply, the documents have been ranked by the relevance level in descending order, *i.e.,* $y_1 \succeq y_2 \ldots \succeq y_n$.

$$\text{RankCosine} \quad L = \frac{1}{2}\left(1 - \frac{\sum_{j=1}^{n} \phi(y_j)\phi(f(\boldsymbol{x_j}))}{\sqrt{\sum_{j=1}^{n} \phi(y_j)^2}\sqrt{\sum_{j=1}^{n} \phi(f(\boldsymbol{x_j}))^2}}\right)$$

$$\text{ListMLE} \quad L = -\log \prod_{i=1}^{n} \frac{\exp\left(\phi(f(\boldsymbol{x_i}))\right)}{\sum_{j=i}^{n} \exp\left(\phi(f(\boldsymbol{x_j}))\right)}$$

$$\text{ListNet} \quad L = -\sum_{\pi \in \mathcal{Y}} \prod_{i=1}^{n} \frac{\phi\big(\psi_y(\boldsymbol{x_{\pi(t_i)}})\big)}{\sum_{j=i}^{n} \phi\big(\psi_y(\boldsymbol{x_{\pi(t_j)}})\big)} \log \prod_{i=1}^{n} \frac{\phi\big(f(\boldsymbol{x_{\pi(t_i)}})\big)}{\sum_{j=i}^{n} \phi\big(f(\boldsymbol{x_{\pi(t_j)}})\big)}$$

where ϕ is a positive and strictly increasing function. $f(\boldsymbol{x_i})$ is the rank score of the document d_i computed by the ranking function f. ψ_y is a mapping function defined on the relevance level and preserves ground truth list, *i.e.,* $\psi_y(y_1) \geq \psi_y(y_2) \ldots \psi_y(y_n)$. $\boldsymbol{x_{\pi(t_i)}}$ denotes the query-document pair feature vector of the document with ranked position being at i.

3.2 Analysis of Listwise Loss Function

Several listwise approaches are developed, but no analysis is studied to elaborate the common characteristic of the listwise loss functions. In this study, we point out that the listwise loss functions are in essence built upon the probability of ranking a document highest among the documents set, defined as

$$h(\boldsymbol{x_i}|\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}; \psi) = \frac{\exp\left(\psi(f(\boldsymbol{x_i}))\right)}{\sum_{j=1}^{n} \exp\left(\psi(f(\boldsymbol{x_j}))\right)} \tag{6}$$

where ψ denotes an increasing function. The existing listwise loss functions can be expressed in terms of the function h.

$$\text{RankCosine} \quad L = \frac{1}{2}\left(1 - \sum_{j=1}^{n} \frac{\phi(y_j)}{\sqrt{\sum_{i=1}^{n} \phi(y_i)^2}} \cdot \sqrt{h\left(\boldsymbol{x_j}|\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}; 2\log\phi\right)}\right)$$

$$\text{ListMLE} \quad L = -\log \prod_{i=1}^{n} h(\boldsymbol{x_i}|\boldsymbol{x_i}, \boldsymbol{x_{i+1}}, \ldots, \boldsymbol{x_n}; \phi)$$

$$\text{ListNet} \quad L = -\sum_{\pi \in \mathcal{Y}} \prod_{i=1}^{n} \frac{\phi\big(\psi_y(\boldsymbol{x_{\pi(t_i)}})\big)}{\sum_{j=i}^{n} \phi\big(\psi_y(\boldsymbol{x_{\pi(t_j)}})\big)} \log \prod_{i=1}^{n} h(\boldsymbol{x_{\pi(t_i)}}|\boldsymbol{x_{\pi(t_{i+1})}} \cdots, \boldsymbol{x_{\pi(t_n)}}; \phi)$$

To minimize the listwise loss function is equivalent to minimizing the function h. The reason for the goodness of these loss functions is that h is closely related to the pairwise classification error. The conclusion is simply proved in the following.

$$\frac{1}{n-i}\sum_{j=i+1}^{n} I[\phi(f(\boldsymbol{x_j})) > \phi(f(\boldsymbol{x_i}))] \leq \frac{1}{n-i}\sum_{j=i+1}^{n} \log_2\left[1+\exp\left(\phi(f(\boldsymbol{x_j}))-\phi(f(\boldsymbol{x_i}))\right)\right]$$

$$\leq \log_2\left(1+\frac{1}{n-i}\sum_{j=i+1}^{n}\exp(\phi(f(\boldsymbol{x_j}))-\phi(f(\boldsymbol{x_i})))\right) \leq -\log_2 h(\boldsymbol{x_i}|\boldsymbol{x_i}, \boldsymbol{x_{i+1}}, \ldots, \boldsymbol{x_n}; \phi)$$

The $I[\phi(f(\boldsymbol{x_j}) > \phi(f(\boldsymbol{x_i})]$ is the pairwise classification error because of $y_i \succeq y_j$. The above inequalities can be verified with $I[z > 0] \leq \log_2(1+\exp(z))$ and $\frac{1}{n}\sum_{j=1}^{n}\log_2(z_j) \leq \log_2(\sum_{j=1}^{n} z_j/n)$ when each z_j is nonnegative. The [1] shows that the pairwise classification error is inadequate for IR where the ranking order on the top of the ranked list is crucial. Hence, the listwise losses are also inadequate for IR since their key component h is closely related to the pairwise classification error.

4 Cost-Sensitive Listwise Approach Framework

4.1 Cost-Sensitive Listwise Loss Function

To make the listwise losses focus on the ranking order on the top of the ranked list, a good way is to take account of cost-sensitive in the listwise losses, more precisely, to set different weights for the documents. The function h is redefined by imposing weights for the documents:

$$h(\boldsymbol{x_i}|\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}; \psi, \boldsymbol{\alpha_i}) = \frac{\alpha_{i,i}\exp(\psi(f(\boldsymbol{x_i})))}{\sum_{j=1}^{n}\alpha_{i,j}\exp(\psi(f(\boldsymbol{x_j})))} \tag{7}$$

where $\boldsymbol{\alpha_i} = (\alpha_{i,1}, \ldots, \alpha_{i,n})$ and its components are nonnegative. $\alpha_{i,j}$ is the weight of the document d_j. The function h can be also expressed as in the form based on the document pairs with weights.

$$h(\boldsymbol{x_i}|\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}; \psi, \boldsymbol{\alpha_i}) = \frac{1}{\sum_{j=1}^{n}\alpha_{i,j}/\alpha_{i,i}\exp(\psi(f(\boldsymbol{x_j})) - \psi(f(\boldsymbol{x_i})))} \tag{8}$$

where $\alpha_{i,j}/\alpha_{i,i}$ is the weight of the document pair (d_i,d_j). The cost-sensitive listwise approaches focus on the performance of the top documents in the ranked list, measured by NDCG. Therefore, one important issue is to relate the weights of the documents with the NDCG. We formulate the problem of setting weights for the documents into the problem of setting weights for the document pairs. In this study, the weights of the document pairs are theoretically given.

4.2 Bound NDCG Errors by Cost-Sensitive Listwise Loss Function

In this section, it is proven that the cost-sensitive listwise losses are the upper bound of NDCG loss. The theorem states definite values of the document pairs' weights. Theoretical proof is based on the lemma 1. For notational simplicity, let $a(i) = 2^{y_i} - 1$, $b(j)$ and its gradient $\nabla b(j)$ are defined as:

$$b(j) = \begin{cases} 1/\log_2(1+j) & j \le k \\ 1/\log_2(1+k) & j > k \end{cases} \quad \nabla b(j) = \begin{cases} \dfrac{-\log 2}{(1+j)\left[\log(1+j)\right]^2} & j \le k \\ 0 & j > k \end{cases} \tag{9}$$

It is simple to prove that the NDCG@k loss $L_{ndcg@k}$ has the following upper bound on the basis of equation (5). Due to space limitation, we omit the proof.

$$L_{ndcg@k} \le \frac{1}{DCG_{\widehat{g}}@k}\left(\sum_{i=1}^{n} a(i)\left(b(\widehat{g}(i)) - b(\pi(i))\right)\right) + \frac{1}{DCG_{\widehat{g}}@k}\sum_{i=1}^{n}\frac{a(i)}{\log_2(1+k)}$$

Lemma 1. *The NDCG@k loss $L_{ndcg@k}$ is upper bounded by weighted pairwise classification loss.*

$$L_{ndcg@k} \le \frac{1}{DCG_{\widehat{g}}@k}\sum_{y_j \succ y_i}\left[-a(j)\nabla b(\widehat{g}(j)) - a(i)\nabla b(\widehat{g}(i))\right] I\left[f(\boldsymbol{x_j}) < f(\boldsymbol{x_i})\right] + C_2$$

where C_2 denotes a constant. The proof is given in the Appendix A. For each query in the training set, the ranking position $\widehat{g}(j)$ of the document d_j is easily computed. Therefore, the weights of the document pairs of each query can be calculated at once in training. Based on the lemma, we can justify the correlation between cost-sensitive listwise loss and NDCG@k loss[1].

Theorem 1. *The cost-sensitive listwise loss is the upper bound of NDCG loss $L_{ndcg@k}$.*

$$L_{ndcg@k} \le -\frac{1}{DCG_{\widehat{g}}@k}\sum_{j=1}^{n}\beta_j \log_2 h(\boldsymbol{x_j}|\boldsymbol{x_{j+1}}, \boldsymbol{x_{j+2}}, \ldots, \boldsymbol{x_n}; \psi, \boldsymbol{\alpha_j}) + C_2 \tag{10}$$

C_2 denotes a constant that is same to the constant in the lemma 1. Based on lemma 1, the above inequality is easily verified with $I[z > 0] \le \log_2(1 + \exp(z))$ and $\sum_{j=1}^{n} w_j \log_2(z_j) \le \left(\sum_{j=1}^{n} w_j\right) \cdot \left(\log_2\left(\sum_{j=1}^{n} w_j z_j / \sum_{t=1}^{n} w_t\right)\right)$ when each z_j and w_j are nonnegative. The theorem demonstrates that many cost-sensitive listwise approaches can be proposed to directly optimize NDCG. For example, we can transform the existing listwise approaches to cost-sensitive listwise methods. Meanwhile, the theorem states definite values for all the weights β_j and $\boldsymbol{\alpha_j}$.

4.3 Differences from Cost-Sensitive Ranking SVM

The cost-sensitive Ranking SVM[1] and the framework for cost-sensitive listwise approaches both make use of cost-sensitive learning in learning to rank, but there are several differences between them.

[1] In this paper, we use NDCG loss and NDCG@k ranking loss exchangeably.

First, the training instance assigned weights is different. The cost-sensitive Ranking SVM weights on the document pairs. The framework credits the weights for the documents. To solve the problem of setting weights for the documents, the framework reduces it into the task of setting weights for document pairs.

Second, the way to calculate the weights of the document pairs is different. The cost-sensitive Ranking SVM sets the weights by the heuristic method. The framework directly computes the weights based on NDCG@k.

Third, the relationship between the loss functions and NDCG loss is also different. The cost-sensitive Ranking SVM do not solve the problem. The framework proves that the listwise losses are the upper bound of the NDCG loss.

4.4 A Case: Cost-Sensitive ListMLE Ranking Approach

To verify the framework, we propose a novel cost-sensitive approach named CS-ListMLE(cost-sensitive ListMLE). The loss function on a query is defined as:

$$L = \frac{1}{DCG_{\hat{g}}@k} \sum_{j=1}^{n} \beta_j \log_2 \left(1 + \sum_{t=j+1, y_j \succ y_t}^{n} \alpha_{j,t} \exp\left(f(\boldsymbol{x_t}) - f(\boldsymbol{x_j})\right) \right) \quad (11)$$

The weights are computed according to Theorem 1. The ranking function of CS-ListMLE is linear, *i.e.*, $f(x) =< w, x >$, where $< \cdot, \cdot >$ is the inner product and w denotes model parameters. We takes gradient descent method to optimize the loss function. Since the loss function is convex, the model parameters converge to a global optimal solution. The algorithm is provided in Figure 1.

Input: training set, learning rate η, tolerance rate ε, NDCG@k
Initialize: w and compute the weights with respect to each query using Eq. (10)
Repeat: do gradient descent until the change of the loss function is below ε
Return w

Fig. 1. Cost-sensitive ListMLE Ranking Approach

5 Experiments

The experiments are conducted on three datasets OHSUMED, TD2003 and TD2004 in Letor2.0[2]. The experiments validate the two points. The one is that whether CS-ListMLE outperforms the ListMLE. The other is that whether CS-ListMLE can obtains higher performance than the other baselines on the top documents of the ranked list. MAP and NDCG@k(N@k) are used as evaluation measures. The truncation level k in NDCG@k is usually 1, 3, 5 and 10. The performances of ListMLE and RankCosine are directly taken from [9], where both approaches do not provide their performance at NDCG@5 and MAP.

[2] `https://research.microsoft.com/en-us/um/beijing/projects/letor/Letor2.0/dataset.aspx`

To validate the effectiveness of CS-ListMLE, we train the algorithm with different parameters. CS-ListMLE@k claims that cost-sensitive ListMLE focuses on the top k documents rank order in the ranked list. It means that CS-ListMLE@k directly optimizes NDCG@k. In the experiments, the k takes 1, 3, 5 and 10.

5.1 Ranking Accuracy of ListMLE and Cost-Sensitive ListMLE

We report the performance of cost-sensitive ListMLE and ListMLE on three datasets in Table 1, 2 and 3. The cost-sensitive ListMLE significantly outperforms ListMLE on the datasets TD2003 and TD2004 in terms of all evaluation measures, while their performance is comparable on OHSUMED.

Table 1. Ranking accuracies on OSHUMED

Methods	N@1	N@3	N@5	N@10	MAP
ListMLE	0.548	0.473	—	0.446	—
CS-ListMLE@1	0.555	0.482	0.464	0.446	0.442
CS-ListMLE@3	0.539	0.480	0.458	0.446	0.444
CS-ListMLE@5	0.548	0.468	0.453	0.438	0.443
CS-ListMLE@10	0.536	0.471	0.452	0.439	0.443

Table 2. Ranking accuracies on TD2003

Methods	N@1	N@3	N@5	N@10	MAP
ListMLE	0.24	0.253	—	0.261	—
CS-ListMLE@1	0.48	0.400	0.362	0.359	0.262
CS-ListMLE@3	0.48	0.400	0.364	0.358	0.253
CS-ListMLE@5	0.48	0.391	0.358	0.355	0.264
CS-ListMLE@10	0.48	0.398	0.362	0.350	0.262

Table 3. Ranking accuracies on TD2004

Methods	N@1	N@3	N@5	N@10	MAP
ListMLE	0.4	0.351	—	0.356	—
CS-ListMLE@1	0.467	0.427	0.422	0.444	0.364
CS-ListMLE@3	0.467	0.429	0.419	0.449	0.366
CS-ListMLE@5	0.467	0.420	0.407	0.444	0.366
CS-ListMLE@10	0.453	0.409	0.406	0.445	0.364

Table 4. Test Results on OSHUMED

Methods	N@1	N@3	N@5	N@10	MAP
ListNet	0.523	0.478	**0.466**	**0.449**	**0.450**
RankCosine	0.523	0.475	—	0.437	—
AdaRank	0.514	0.462	0.442	0.437	0.442
RSVM	0.495	0.465	0.458	0.441	0.447
CS-ListMLE@1	**0.555**	**0.482**	0.464	0.446	0.442

Table 5. Test Results on TD2003

Methods	N@1	N@3	N@5	N@10	MAP
ListNet	0.46	**0.408**	**0.382**	**0.374**	**0.273**
RankCosine	0.36	0.346	—	0.322	—
AdaRank	0.42	0.291	0.242	0.194	0.137
RSVM	0.42	0.379	0.347	0.341	0.256
CS-ListMLE@1	**0.48**	0.400	0.362	0.359	0.262

Table 6. Test Results on TD2004

Methods	N@1	N@3	N@5	N@10	MAP
ListNet	0.440	**0.437**	0.420	**0.458**	**0.372**
RankCosine	0.439	0.397	—	0.405	—
AdaRank	0.413	0.402	0.393	0.406	0.331
RSVM	0.44	0.410	0.393	0.420	0.350
CS-ListMLE@1	**0.467**	0.427	**0.422**	0.444	0.364

We explain why CS-ListMLE remarkably outperforms ListMLE on datasets TD2003 and TD2004. On one hand, each query in datasets TD2003 and TD2004 has 1000 documents. The ratio of the queries containing at most 15 relevant documents in all queries is 95% and 89.3% respectively. Since ListMLE considers all the document pairs, 95% and 89.3% queries loss functions of ListMLE induces the losses caused by 484620 irrelevant document pairs. As is well known, the NDCG loss is not sensitive to the losses generated by the irrelevant document

pairs. Thus, ListMLE introduces a large deviation from the NDCG loss. However, CS-ListMLE only cares about the document pairs composed of relevant documents and irrelevant documents. On the other hand, ListMLE treats the all documents equally. But CS-ListMLE assigns the weights for the document based on NDCG@k. In summary, the loss of CS-ListMLE is more close to NDCG@k loss than ListMLE on datasets TD2003 and TD2004.

As far as dataset OHSUMED, 81.1% queries have less than 200 documents, while 85.89% queries contain at least 10% relevant documents. Under such situation, the weights of the documents does not affect much in CS-ListMLE. The loss of CS-ListMLE is approximate to the loss of ListMLE.

5.2 Comparison with the Other Baselines

We take CS-ListMLE@1 as an example to compare the performance with the other baselines, including RankCosine[5], ListNet[4], RSVM[2] and AdaRank[7]. Experimental results are presented in Table 4, 5 and 6. CS-ListMLE almost outperforms RankCosine, AdaRank and Ranking SVM on three datasets at all evaluation measures. Compared to ListNet, CS-ListMLE obtains higher performance at NDCG@1. We conduct t-test on the improvement of CS-ListMLE over ListNet, Ranking SVM and AdaRank on the three datasets in terms of NDCG@1. The results indicate that the improvement of NDCG@1 over Ranking SVM and AdaRank on dataset OHSUMED is statistically significant(p-value$<$0.05). There is no statistically significant difference on dataset TD2003 in spite of rising 6% over AdaRank and Ranking SVM.

Experiments results demonstrate that the CS-ListMLE can achieve high performance on NDCG@1, which meets the goal that the CS-ListMLE focuses on the top one documents ranking order of the ranked list. Meanwhile, the cost-sensitive ListMLE obtains comparable performance to the baselines at MAP.

6 Conclusion

In this paper, we point out that the existing listwise losses are inadequate IR where the documents with higher ranks should be emphasized. To address the issue, we propose a framework for cost-sensitive listwise approaches. The framework credits weights for the documents. The framework reduces the problem of setting weights for the documents to the problem of setting weights for the document pairs. The weights of the document pairs are computed based on the NDCG. It is proven that the cost-sensitive listwise loss is the upper bound of NDCG loss. As an example, we develop a cost-sensitive ListMLE approach. Experimental results show that the cost-sensitive ListMLE outperforms ListMLE on two benchmark datasets in terms of all evaluation measures. In addition, the cost-sensitive ListMLE almost outperforms the baselines, such as RankCosine, AdaRank and Ranking SVM, on the three datasets at all evaluation measures.

References

1. Xu, J., Cao, Y., Li, H., Huang, Y.: Cost-sensitive learning of svm for ranking. In: ECML, pp. 833–840 (2006)
2. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 133–142. ACM, New York (2002)
3. Xia, F., Liu, T.Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th international conference on Machine learning, pp. 1192–1199. ACM, New York (2008)
4. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning, pp. 129–136. ACM, New York (2007)
5. Qin, T., Zhang, X.D., Tsai, M.F., Wang, D.S., Liu, T.Y., Li, H.: Query-level loss functions for information retrieval. Inf. Process. Manage. 44(2), 838–855 (2008)
6. Lan, Y., Liu, T.Y., Ma, Z., Li, H.: Generalization analysis of listwise learning-to-rank algorithms. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 577–584. ACM, New York (2009)
7. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 391–398. ACM, New York (2007)
8. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. 20(4), 422–446 (2002)
9. Xia, F., Liu, T.Y., Li, H.: Statistical consistency of top-k ranking. In: Advances in Neural Information Processing Systems, pp. 2098–2106 (2009)

A Theoretical Justification of Lemma 1

$$L_{ndcg@k} \leq \frac{1}{DCG_{\hat{g}}@k}\left(\sum_{i=1}^{n} a(i)\left(b(\widehat{g}(i)) - b(\pi(i))\right)\right) + \frac{1}{DCG_{\hat{g}}@k}\sum_{i=1}^{n}\frac{a(i)}{\log_2(1+k)}$$

$$C_1 = \frac{1}{DCG_{\hat{g}}@k}\sum_{i=1}^{n}\frac{a(i)}{\log_2(1+k)} \quad C_2 = C_1 + \frac{1}{DCG_{\hat{g}}@k}\sum_{y_j=y_i}\left[-a(j)\nabla b(\widehat{g}(j)) - a(i)\nabla b(\widehat{g}(i))\right]$$

$\because b$ is convex function $\Longleftrightarrow b(x) - b(y) \leq \nabla b(x)(x-y)$

$$\leq \frac{1}{DCG_{\hat{g}}@k}\sum_{j=1}^{n} a(j)\cdot\nabla b(\widehat{g}(j))\cdot(\widehat{g}(j) - \pi(j)) + C_1$$

$$= \frac{1}{DCG_{\hat{g}}@k}\sum_{j=1}^{n} a(j)\cdot(-\nabla b(\widehat{g}(j)))\cdot\left\{\left(1 + \sum_{i=1}^{n} I[f(\boldsymbol{x_i}) > f(\boldsymbol{x_j})]\right) - \left(1 + \sum_{i=1}^{n} I[y_i > y_j]\right)\right\} + C_1$$

$\because I[x>0] - I[y>0] \leq I[xy<0] + I[y=0]$

$$\leq \frac{1}{DCG_{\hat{g}}@k}\sum_{y_j \succ y_i}\left[-a(j)\nabla b(\widehat{g}(j)) - a(i)\nabla b(\widehat{g}(i))\right] I[f(\boldsymbol{x_j}) < f(\boldsymbol{x_i})] + C_2$$