Smoothing DCG for Learning to Rank: A Novel Approach Using Smoothed Hinge Functions

Mingrui Wu, Yi Chang, Zhaohui Zheng
Web Search Ranking Group, Yahoo! Labs, 701 First Avenue, Sunnyvale, CA 94089
{mingrui,yichang, zhaohui}@yahoo-inc.com

Hongyuan Zha
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
zha@cc.gatech.edu

ABSTRACT

Discounted cumulative gain (DCG) is widely used for evaluating ranking functions. It is therefore natural to learn a ranking function that directly optimizes DCG. However, DCG is non-smooth, rendering gradient-based optimization algorithms inapplicable. To remedy this, smoothed versions of DCG have been proposed but with only partial success. In this paper, we first present analysis that shows it is ineffective using the gradient of the smoothed DCG to drive the optimization algorithm. We then propose a novel approach, SHF-SDCG, for smoothing DCG by using smoothed hinge functions (SHF). It has the advantage of seamlessly transition from driving the optimization mimicking pairwise learning when the ranking function does not fit the data well, to driving the optimization using DCG when the ranking function becomes more accurate. SHF-SDCG is then extended to REG-SHF-SDCG, an algorithm which gradually transits from pointwise and pairwise to listwise learning. Finally experimental results are provided to validate the effectiveness of SHF-SDCG and REG-SHF-SDCG.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Retrieval functions*

General Terms

Algorithms, Experiments

Keywords

ranking function, learning to rank, machine learning, optimization, gradient descent, gradient boosting

1. INTRODUCTION

Search engines are widely used tools for effectively exploring the information on the web. The core of a search engine is its ranking function: When a search engine receives a user query, this function assigns a real valued score to each of a given set of documents (or web URLs). The search engine then returns the list of documents according to the decreasing order of scores, which means the the larger the score of a document, the more relevant this document is to the user query.

Many approaches have been proposed for constructing the ranking function due to its important role in search engines. In particular, adopting the machine learning methods to learn the ranking function, or learning to rank, has recently attracted more and more efforts. In these methods, the training data contains a set of queries, a set of documents for each query and a label or grade for each document indicating the degree of relevance of this document to its corresponding query. For example, each grade can be one element in the ordinal set:

$$\{\text{perfect, excellent, good, fair, bad}\} \tag{1}$$

which is generally assigned by human editors.

The *Discounted Cumulative Gain* (DCG) has been widely used as a main measurement to assess relevance in the context of search engines [2]. Therefore, when constructing a learning to rank approach, it is important to consider how to optimize model parameters with respect to the DCG value. Many machine learning algorithms apply the gradient based techniques for parameter optimization. Unfortunately, the DCG metric is not smooth. This makes it difficult to directly optimize it with gradient based approaches. Consequently many current ranking algorithms turn to optimize other objectives, such as regression error and pairwise preferences.

To solve this problem, recently the SoftRank approach [4] treats the deterministic output values of a ranking function as Gaussian random variables, based on which a smooth approximation of DCG is derived.

In this paper, we also start from the idea of constructing a smooth DCG objective function. Similar as SoftRank, it is a smooth approximation of DCG. However, our approximation is deterministic and random distributions are not involved. More importantly, our main contribution is to show the problems of this approach and elucidate that directly optimizing this approximation of DCG is still not a good way to learn the ranking function, even it is smooth. We then provide solutions to overcome these problems to construct our algorithms for the learning to rank problem. Specifically, we propose a SHF-SDCG algorithm, which uses *smooth hinge functions* (SHF). This algorithm has the advantage of seamlessly transition from driving the optimization mimicking pairwise learning when the ranking function does not fit the data well, to driving the optimization using DCG when the ranking function becomes more accurate. Then SHF-SDCG is further improved and extended to REG-SHF-SDCG, which firstly reaches a reasonably well solution by regression.

The remaining of this paper is organized as follows. In section 2, we formulate a smooth DCG objective, which is the start of our approach. Section 3 is the main part, which explains the problems of learning a ranking function by directly optimizing this approximation of DCG, even it is smooth. And we provide our solutions to these problems which lead to two ranking algorithms SHF-SDCG and REG-SHF-SDCG. Experimental results are provided in section 4 and we conclude the paper in the last section.

2. SMOOTHING DCG

2.1 DCG Metrics

For one query, and a list of n documents, suppose the relevance grade of the i-th document with respect to this query is y_i, then the DCG is defined as

$$\text{DCG@k} = \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1 + r_i)} W_k(r_i) \tag{2}$$

In equation (2), r_i is the rank of the i-th document, k is a positive integer, while the weight function $W_k(r_i)$ equals 1 if $r_i \leq k$, and 0 otherwise. Another widely used DCG metric is the Normalized DCG (NDCG) [2], which is defined as

$$\text{NDCG@k} = \text{DCG@k}/Z \tag{3}$$

where Z is the normalization factor such that the perfect ranking of the list gives an NDCG value of 1.

In the following, we will focus on DCG@k metric to derive our ranking algorithm. But our approach can be straightforwardly applied to NDCG@k.

2.2 A Basic Smooth DCG Objective

In (2), the rank value r_i can be approximated as $r_i \approx 1 + \sum_{j \neq i} H(o_i - o_j)$, where o_i and o_j are the output values for document i and j of the ranking function, and $H(\cdot)$ is the step function,

$$H(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The sigmoid function $G_\alpha(\cdot)$ can be adopted as a smooth approximation of the step function $H(\cdot)$:

$$G_\alpha(x) = \frac{1}{1 + \exp(\alpha x)} \tag{5}$$

where $\alpha > 0$ is a parameter. Based on $G_\alpha(\cdot)$, we can build smooth approximations for r_i and $W_k(\cdot)$:

$$\hat{r}_i = 1 + \sum_{j \neq i} G_\alpha(o_i - o_j) \tag{6}$$

$$\hat{W}_k(x) = G_\beta(x - k) \tag{7}$$

Now we can define a smooth approximation of DCG@k:

$$\text{SDCG@k} = \sum_{i=1}^{n} \frac{S(y_i)}{log_2(1 + \hat{r}_i)} \hat{W}_k(\hat{r}_i) \tag{8}$$

where $\hat{r}_i$ and $\hat{W}_k$ are defined in (6) and (7) respectively. Clearly SDCG@k is smooth and can be readily optimized by gradient based algorithms. In particular, we adopt the *Gradient Boosting Tree* [1] GBT approach to optimize it.

3. SHF-SDCG AND REG-SHF-SDCG

In this section, We will carry out a careful analysis of the gradient of the SDCG objective to pinpoint some of its problems, which result in our proposed ranking algorithms.

3.1 SHF-SDCG

To facilitate the discussion, we consider a simple situation where there are only two documents $\mathbf{x}_1$ and $\mathbf{x}_2$ for a query, with relevance grades y_1 and y_2 respectively.

Let o_i denote the output value of the ranking function for document $\mathbf{x}_i$ $(i = 1, 2)$. First we investigate the gradient of the SDCG objective with respect to o_i:

$$\frac{\partial \text{SDCG@k}}{\partial o_i} = \frac{\partial \text{SDCG@k}}{\partial \hat{r}_1} \frac{\partial \hat{r}_1}{\partial o_i} + \frac{\partial \text{SDCG@k}}{\partial \hat{r}_2} \frac{\partial \hat{r}_2}{\partial o_i} \tag{9}$$

where $\frac{\partial \hat{r}_j}{\partial o_i}$ is calculated as

$$\frac{\partial \hat{r}_j}{\partial o_i} = \frac{\partial G_\alpha(\eta(o_1 - o_2))}{\partial o_i}, \quad 1 \leq i, j \leq 2 \tag{10}$$

where $\eta = 1$ if $j = 1$, and -1 otherwise.

From Figure 1, we can see that the curve of $G_\alpha(x)$ is flat, i.e. the gradient value of close to 0, when the absolute value of x is large, and it has relatively larger values when x is close to 0. Suppose $y_2 < y_1$ and at a certain *Gradient Descent* GD iteration, the ranking function gives the wrong order for this two documents, i.e. $o_2 > o_1$. Based on (10) and Figure 1, we can see that the a larger value of $o_2 - o_1$ can lead to gradient values $\frac{\partial \hat{r}_j}{\partial o_i}$ close to 0. Therefore correcting the order of these two documents is difficult since $o_2 - o_1$ will be changed very little by the current GD iteration. This is not desirable since we hope in each iteration we can focus more on those incorrectly ranked pairs and correct them quickly.

Even worse, from (10) we can see that,

$$\frac{\partial \hat{r}_1}{\partial o_i} = -\frac{\partial \hat{r}_2}{\partial o_i}, \quad i = 1, 2 \tag{11}$$

Therefore the two terms on the right hand side of (9) give opposite updating directions for o_i, $i = 1, 2$, which means that the optimization procedure is quite inefficient. By investigating the above simple case, we can see that *even though the SDCG objective is smooth, directly optimizing this objective is not effective since correcting wrongly ranked pairs requires many optimization iterations.* To overcome this problem, we propose to modify the calculation of $\hat{r}_1$ and $\hat{r}_2$ as follows:

$$\hat{r}_1 = 1 + A_\alpha(o_1 - o_2) \tag{12}$$

$$\hat{r}_2 = 1 + B_\alpha(o_2 - o_1) \tag{13}$$

where

$$A_\alpha(x) = \begin{cases} G_\alpha(x) & \text{if } x \geq 0 \\ 0.5 + \frac{\alpha}{4} x & \text{otherwise} \end{cases} \tag{14}$$

and

$$B_\alpha(x) = 1 + A_\alpha(-x) \tag{15}$$

It is interesting to note that $A_\alpha(x)$ and $B_\alpha(x)$ can be considered as a smoothed version of the hinge functions.

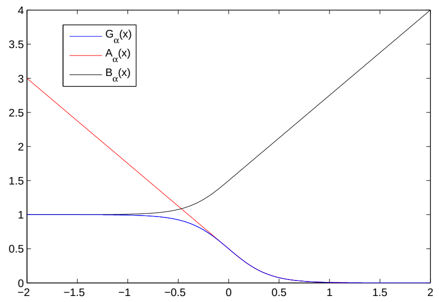

Figure 1: $G_\alpha(x)$, $A_\alpha(x)$ and $B_\alpha(x)$

The figure of the three functions $G_\alpha(\cdot)$ with $A_\alpha(\cdot)$ and $B_\alpha(\cdot)$ are displayed in Figure 1. Based on (14) and (15), we can see that when $o_2 > o_1$, we have:

$$\left|\frac{\partial \hat{r}_j}{\partial o_i}\right| = \frac{\alpha}{4} \quad \text{and} \quad \frac{\partial \hat{r}_1}{\partial o_i} = \frac{\partial \hat{r}_2}{\partial o_i} \tag{16}$$

where $1 \le i, j \le 2$. This way, whenever the ranking order of these two documents is wrong, the GD iterations can effectively update them towards the right direction.

Thus, by replacing $G_\alpha(\cdot)$ with $A_\alpha(\cdot)$ and $B_\alpha(\cdot)$ for $\hat{r}_1$ in (12) and $\hat{r}_2$ in (13), we can overcome the problems described above. By replacing the sigmoid function with SHFs, we obtain anthoer objective, which will be called SHF-SDCG in the following.

3.2 Weight Function

In (8), there is a weight function $\hat{W}_k(\cdot)$ calculated as (7). Suppose at a certain GD iteration, a perfectly relevant document for a query is not well ranked at position, say, $k+n$. The weight value of this document is $G_\beta(n)$, which decreases very fast as n increases. This means this wrongly ranked documents tend to be ignored and hence improving their ranks is difficult. Let s_m denote the largest rank value of the top k documents, i.e. the k most relevant documents, for a query at the m-th GD iteration, we modify the weight function in (7) as,

$$\hat{W}_m(x) = G_\beta(x - s_m) \tag{17}$$

This way, we can make sure that all the top k documents of each query can get enough weights. The width s_m becomes smaller and smaller as GD iterations keep improving the ranking result. When s_m is close to k, $\hat{W}_m(\cdot)$ is also close to the original weight function $W_k(\cdot)$ in DCG metric (2). This is similar as the idea of the *Deterministic Annealing* (DA) algorithm. In order to solve a difficult optimization problem, DA optimizes a series of objective functions, which are easier to optimize and gradually approach the original objective.

3.3 REG-SHF-SDCG

Our objective function is not convex. Therefore a good starting point is helpful for obtaining a good result. We propose to smoothly mix SHF-SDCG with regression by minimizing the following objective function:

$$R \times \tau_m - \text{SHF-SDCG} \times (1 - \tau_m) \tag{18}$$

where R is the regression error, $0 \le \tau_m \le 1$ decreases when the number of iterations m increases, giving SHF-SDCG more and more weights. In our algorithm, τ_m is calculated as:

$$\tau_m = \frac{1}{1 + \exp(\gamma(m - M/2))} \tag{19}$$

where M is the total number of the iterations, and γ is chosen such that $\tau_1 = 0.999999$. According to (19), $\tau_m = 0.5$ when $m = M/2$. Namely, regression error and SHF-SDCG have the same weight when half of the iterations have been finished. This is just a heuristic choice, and in practice we found it usually leads to good results.

In the following, we use REG-SHF-SDCG to denote objective defined based on (18) and (19).

3.4 Transition from Pointwise and Pairwise to Listwise Learning

In REG-SHF-SDCG, at the beginning, minimizing the regression error, which is pointwise, is the focus of optimization such that a reasonably good ranking solution can be reached. As the number of iterations increases, the second term in (18), i.e. SHF-SDCG has more and more importance. At this time, for a query, when there are many incorrectly ranked pairs, SHF-SDCG is not an accurate approximation of the true DCG value. For this query, the optimization procedure is mainly to correct those wrongly ranked pairs, which is more like a pairwise approach. As the number of iterations increases more and the ranking result is further improved, for the queries whose most documents are correctly ranked, and SHF-SDCG is a good smooth approximation of DCG metric, optimizing SHF-SDCG can lead to a good DCG value. Namely, at this stage, a listwise objective is being optimized. Hence we can see that in REG-SHF-SDCG and SHF-SDCG, we actually transit from pointwise and pairwise to listwise learning.

4. EXPERIMENTAL RESULTS

4.1 Experiments on a Commercial Search Engine Data Collection

For this search engine data set, we extracted about 400 features. The queries are sampled from the search engine query logs and a certain number of query-document pairs are labeled according to their relevance judged by human editors. The five levels of grades shown in (1) is adopted. For experiment, we used a training, a validation and a test set, which contain 8179, 3755 and 916 queries respectively. The number of query-url pairs constrained in these three data sets are 322763, 111561 and 32008.

In [5], a pairwise ranking algorithm called GBRank, is presented which shows the state of art ranking results. It also adopts the GBT framework for training and optimization. We will include GBRank for comparison. As a baseline, we also show the results of a simple regression approach, denoted by Regression in the following, which uses the GBT method to train a regression model.

In the experiment, 600 trees were used for all these five algorithms. Figure 2 gives the results. It can be seen that REG-SHF-SDCG compares favorably to all the other algorithms. In particular, REG-SHF-SDCG outperforms SHF-SDCG, which indicates that the mixture with regression (18) can improve the ranking results, since a better staring point can be reached. And it can be observed that both REG-SHF-SDCG and SHF-SDCG can beat the SDCG algorithm.

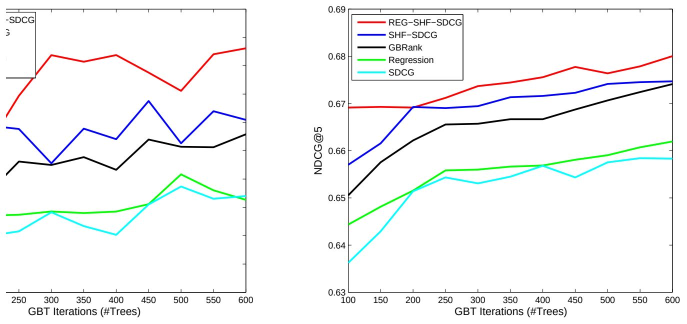

Figure 2: NDCG@1 and NDCG@5 of REG-SHF-SDCG, SHF-SDCG, SDCG, GBRank, and Regression.

Table 1: NDCG, MAP and Precision at position k on OHSUMED data (average over 5 folds)

Algorithms	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	P@1	P@2	P@3	P@4	P@5	MAP
RankBoost	0.4632	0.4504	0.4555	0.4543	0.4494	0.5576	0.5481	0.5609	0.5580	0.5447	0.4411
RankSVM	0.4958	0.4331	0.4207	0.4240	0.4164	0.5974	0.5494	0.5427	0.5443	0.5319	0.4334
FRank	0.5300	0.5008	0.4812	0.4694	0.4588	0.6429	0.6195	0.5925	0.5840	0.5638	0.4439
ListNet	0.5326	0.4810	0.4732	0.4561	0.4432	0.6524	0.6093	0.6016	0.5745	0.5502	0.4457
AdaRank.MAP	0.5388	0.4789	0.4682	0.4721	0.4613	0.6338	0.5959	0.5895	0.5887	0.5674	0.4487
AdaRank.NDCG	0.5330	0.4922	0.4790	0.4688	0.4673	0.6719	0.6236	0.5984	0.5838	0.5767	0.4498
SDCG	0.5012	0.4750	0.4714	0.4533	0.4420	0.6152	0.5816	0.5802	0.5461	0.5313	0.4501
SHF-SDCG	0.5409	0.5022	0.4743	0.4701	0.4598	0.6333	0.6141	0.5922	0.5790	0.5593	0.4503
REG-SHF-SDCG	0.5517	0.5110	0.4802	0.4716	0.4634	0.6333	0.6152	0.5893	0.5716	0.5574	0.4506

In fact, SDCG has even lower NDCG values than Regression. All these support our analysis in section 3.

4.2 Experiments on OHSUMED data

The OHSUMED data set we used is contained in the LETOR 3.0 package [3], which is derived from the existing data sets widely used in IR. The OHSUMED data set is a subset of the MEDLINE database, which is popular in IR community. This data set contains 106 queries. The documents are manually labeled with three levels of relevance grades: definitely relevant, possibly relevant and not relevant. Three metrics are adopted here: NDCG, Precision and Mean Average Precision (MAP), which have been used in literature of ranking.

For OHSUMED, 250 trees were used for SDCG, SHF-SDCG and REG-SHF-SDCG. These three algorithms are compared with other six state of art learning to rank algorithms reported in the LETOR package. The results are listed in Table 2.

It can be observed that REG-SHF-SDCG has the highest NDCG@1, NDCG@2 values, and it has the second best NDCG@3, NDCG@4 and NDCG@5 values. This indicates that REG-SHF-SDCG is an effective algorithm for optimizing the DCG metric. The precision values of REG-SHF-SDCG are not strong compared with other algorithms, since it aims to optimize DCG metric. However, it is interesting to see that it has the highest MAP value, also SDCG and SHF-SDCG have better MAP values than the six algorithms. Also, comparing REG-SHF-SDCG, SHF-SDCG and SDCG, we can see the similar results as in the last subsection. All these again support our proposed approaches.

5. CONCLUSIONS

We have proposed a smooth DCG approach for learning ranking functions for web search. Starting from a "faithful" smooth approximation of the DCG metric, we have elucidated some of its problems and pointed out that optimizing an accurate approximation of DCG metric is not effective. Then we have provided solutions to these problems and formulated REG-SHF-SDCG and SHF-SDCG, which can be readily optimized by the gradient descent. The proposed approaches illustrate properties of pointwise, pairwise and listwise ranking approaches in different optimization stages, and the objective functions being optimized gradually approach an accurate smooth approximation of the true DCG metric. Experimental results on both a commercial search engine data set and a publicly available benchmark data set have shown encouraging results.

6. REFERENCES

[1] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist*, 29, 2001.

[2] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002.

[3] T.-Y. Liu, T. Qin, J. Xu, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007, in conjunction with SIGIR 2007*, 2007.

[4] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimising non-smooth rank metrics. In *International ACM International Conference on Web Search and Data Mining*, 2008.

[5] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgements. In *International ACM SIGIR Conference on Resarch and Development in Information Retrieval*, 2007.