

Robust Sparse Rank Learning for Non-Smooth Ranking Measures

Zhengya Sun
Institute of Automation,
Chinese Academy of Sciences
Beijing, 100190, P. R. China
zhengya.sun@ia.ac.cn

Tao Qin
Microsoft Research Asia,
No.49 Zhichun Road
Beijing, 100190, P. R. China
taoqin@microsoft.com

Qing Tao, Jue Wang
Institute of Automation,
Chinese Academy of Sciences
Beijing, 100190, P. R. China
{qing.tao,jue.wang}@ia.ac.cn

ABSTRACT

Recently increasing attention has been focused on directly optimizing ranking measures and inducing sparsity in learning models. However, few attempts have been made to relate them together in approaching the problem of learning to rank. In this paper, we consider the sparse algorithms to directly optimize the Normalized Discounted Cumulative Gain (NDCG) which is a widely-used ranking measure. We begin by establishing a reduction framework under which we reduce ranking, as measured by NDCG, to the importance weighted pairwise classification. Furthermore, we provide a sound theoretical guarantee for this reduction, bounding the realized NDCG regret in terms of a properly weighted pairwise classification regret, which implies that good performance can be robustly transferred from pairwise classification to ranking. Based on the converted pairwise loss function, it is conceivable to take into account sparsity in ranking models and to come up with a gradient possessing certain performance guarantee. For the sake of achieving sparsity, a novel algorithm named RSRank has also been devised, which performs L_1 regularization using truncated gradient descent. Finally, experimental results on benchmark collection confirm the significant advantage of RSRank in comparison with several baseline methods.

Categories and Subject Descriptors

H.3.3 [Information Storage And Retrieval]: Information Search and Retrieval; I.2.6 [Artificial Intelligence]: Learning

General Terms

Theory, Algorithms, Experimentation

Keywords

Information Retrieval, Learn to Rank, RSRank, truncated gradient

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '09, July 19–23, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-483-6/09/07 ...\$5.00.

1. INTRODUCTION

Rank learning, which aims to automatically learn a ranking function according to documents' relevance to a given query, plays a pivotal role in document retrieval. In order to credit ranking functions for their ability to retrieve highly relevant documents, evaluation measures from the viewpoint of information retrieval (IR) are developed, such as Winner Takes All (WTA)[17], Mean Reciprocal Rank (MRR)[17], Mean Average Precision (MAP)[2], and Normalized Discounted Cumulative Gain (NDCG)[11].

These widely-used IR measures characterize themselves by permutational structure dependent, which makes the task of ranking present a challenging research topic in supervised learning. Because permutational structure is only available via ordering, and ordering is intrinsically discontinuous. Thus, any evaluation measure used to examine the goodness of a ranking function up to a given sort is non-smooth. It seems difficult to learn an efficient ranking function through directly optimizing evaluation measures, which really makes sense in practice.

Accordingly, great efforts have been made to address the issue of direct optimization of ranking measures whose purpose is mainly to follow IR measures to push the more relevant document closer to the top of the ranked list. The algorithms, which have been recently developed and validated, can be divided into two categories. The methods in the first major category attempt to formulate an explicit smooth objective function which are then approached by various optimization strategies[8, 14, 16, 18]. The second category of methods, however, works with implicit objective functions which facilitate us to specify some rules about how to change rank orders for a given sorted instances, represented by LambdaRank[6] which is also our concern.

LambdaRank which characterizes itself by an implicit loss function and an explicitly defined gradient provides several edges in working with loss functions that are either flat or non-differentiable, such as easy to implement and good performance in practice. However, there still remain some issues in need of further investigation.

First, there isn't sufficient theoretical justification on the correlation between NDCG and the implicit loss function which is constructed by reducing ranking into importance weighted pairwise classification, thus lacking in a general guideline for generating gradients with desirable properties.

Second, few attempts have been made to touch on the issue of sparsity in ranking models, which means that some elements in model parameter vector are exactly zero. Along with the number of available ranking features mounts up,

ranking models become more complicated and worse interpretable, which inspires a ranking model that not only fits the data well but can also achieve sparsity.

To tackle the problems above, this paper attempts to introduce reduction analysis and sparsification to approach rank learning problems under NDCG criteria.

We first validate the approach that reduces ranking, as measured by NDCG, to the importance weighted pairwise classification. In terms of regret analysis, we demonstrate that for arbitrary probability distributions over instances, the importance weighted pairwise classifier with regret r points to an NDCG regret of at most r . This implies that ranking algorithms based on such a reduction are robust in a sense that a small pairwise classification regret cannot lead to a large NDCG regret. In other words, pairwise classification algorithm can be robustly translated into NDCG ranking algorithm, whereupon a general framework for generating gradients with desirable properties can be derived.

When sparsity is concerned, it is natural to fall back on 1-norm regularization which enjoys sound theoretical explanation[22, 23]. To the gradient approximation oriented problem, however, simply adding L_1 regularization to the gradient cannot lead to sparsity[13]. Therefore, we further devise a novel ranking algorithm, referred to as RSRank, which performs L_1 regularization using truncated gradient descent for the sake of achieving sparsity. The empirical results indicate that even with sparse models, the ranking performance is still comparable to that of the standard gradient descent ranking algorithm.

Finally, we evaluate the proposed method on LETOR 3.0 benchmark collections[1]. Experimental results manifest that RSRank not only achieves good sparsity in practice, but also exhibits a high level of performance in comparison with several proposed baseline algorithms.

The remainder of the paper is organized as follows. In Section 2 the related work is presented. Section 3 introduces some basic definitions and provides theoretical analysis on the concerned reduction model. Section 4 discusses the design of RSRank algorithm, and empirical results are reported in Section 5. Section 6 concludes the paper and discusses the future work.

2. RELATED WORK

Direct optimization of IR measures has become one desirable direction for dealing with rank learning problems in recent years. The approaches that have been brought forward to handle this problem fall into two categories. For the approaches in the first category, [8, 9, 14] are under the large-margin structured learning framework and constructs an explicit objective function with constraints, [16, 10, 15] bypass an explicit sort and creates a smoothed objective with approximate ranking positions, and [18, 19] build an optimization objective after performing a sort and designs parameter update rules.

Instead of defining an explicit objective function, the second category of approaches resorts to a more straightforward way and simply focuses on a desirable change of each document's score for a given sorted list. In other words, the gradients are only required to be specified with respect to an implicit objective function. This category, characterized by LambdaRank[6], not only benefit from easiness to implement, but also from significant efficiency in applications of commercial interest. Given a pair of documents with differ-

ent relevance degrees, LambdaRank specifies the gradient with respect to an implicit objective function, for example the RankNet cost scaled by the NDCG gain found by swapping the two documents in question. When applied to commercial search engines, this method performs well and enjoys significant benefits on training speed and accuracy. To improve the efficiency of LambdaRank in high dimensional search space, Yisong Yue etc. [20] use Simultaneous Perturbation Stochastic Approximation (SPSA) as the gradient approximation method and also examine the empirical optimality of LambdaRank.

This paper attempts to conduct a study on the property when reducing ranking, as measured by NDCG, to importance weighted pairwise classification. In reality, analogous researches have been engaged in on reduction between classification tasks[4, 5] and reduction from ranking, as measured by the Area Under the Receiver Operating Characteristic Curve (AUC), to binary classification[3]. For the proposed reduction transformations, regret analysis receives intensive investigations. Regardless of the number of samples needed to achieve a certain level of performance, regret analysis focuses on relative error guarantees between general problems arising in practice and basic problems that are better understood, and apply when there are arbitrary high-order dependencies between samples. The derived bound have validated these reduction models which are robust in a sense that a small regret on the subproblem implies a small regret on the original problem. In addition, there are convincing empirical evidences that these reductions work well in practice, which offers further support to this style of analysis[12].

3. NDCG REDUCTION FRAMEWORK

The problem of learning to rank can be described as below. Based on a given query, there is a corpus of documents with labels specifying the degree of relevance. The input to the ranker consists of a sequence of feature vectors constructed from query-document pairs, and then the ranking function assigns a score to each item in the set, followed by producing a sorted list of items in descending order of scores. The quality of the sorted list is then evaluated by ranking measures used in the information retrieval community.

3.1 NDCG

In order to evaluate the quality of the retrieved documents, several ranking measures from the viewpoint of IR are developed, such as WTA, MRR, MAP and NDCG. WTA pays attention to whether the top ranked document is relevant. MRR is calculated according to the position of the top ranking relevant document. In the sight of MAP, the higher relevant documents are ranked, the better retrieved result gets, regardless of the fact that documents are not of equal relevance. NDCG, however, gives a high evaluation to the sorted list where highly relevant documents are ranked in front. More accurately, the NDCG value of a sorted list at position k is calculated as below:

$$N(k) = Z_k \cdot \sum_{j=1}^k \frac{2^{r(j)} - 1}{\log_2(1 + j)} \quad (1)$$

where Z_k is the normalization constant so that the perfect ranking is evaluated as 1. $r(j)$ is the label of the j th document in the sorted list. Let $a(j) = 2^{r(j)} - 1$ and $b(j) =$

$1/\log_2(1+j)$, equation (1) can be simplified as:

$$N(k) = Z_k \cdot \sum_{j=1}^k a(j) \cdot b(j) \quad (2)$$

Apparently, the former three ranking measures are based on binary relevance judgments, while NDCG, based on graded relevance judgments, is more suitable for the user requirements and is chosen as the ranking measure in this paper.

3.2 Problem Definition

We begin by presenting the basic notation descriptions to be used. Let $\mathcal{X}^{(n)} = \{x_1, x_2, \dots, x_n\}$ denote the set of input instances to be ordered, where each instance $x_i \in \mathbb{R}^d$ is a feature vector, and $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ denote the set of input instances and the corresponding relevance labels, with each label $y_i \in \{r_1, r_2, \dots, r_l\}$ where a total order relation exists, i.e. $r_l > r_{l-1} > \dots > r_1$. Let $\mathcal{X}^{(2)} = \{(x_i, x_j) \mid x_i, x_j \in \mathcal{X}^{(n)}, i < j\}$ denote the set of ordered pairs of different instances in $\mathcal{X}^{(n)}$, and the corresponding set with relevance labels is written as $\mathcal{S}^{(2)} = \{(x_i, x_j), \mathbf{1}(y_i, y_j) \mid (x_i, y_i), (x_j, y_j) \in \mathcal{S}, i < j\}$ where $\mathbf{1}(\cdot)$ is 1 when the argument is true and 0 otherwise. Let $f: \mathbb{R}^d \rightarrow \mathbb{R}$ denote the ranking function, and $f(x_i)$ indicate the score assigned to instance x_i by the ranking function f . Let π denote a permutation of $[1, 2, \dots, n]$, and π_i indicate the ranking position of instance x_i .

We consider the problem of ranking in a reduction manner, referred to as NDCG reduction in this paper. The NDCG reduction reduces ranking, as measured by NDCG, to importance weighted pairwise classification. We first give definitions of the problems involved.

Definition 1. An importance weighted pairwise classification problem is defined by a distribution P over the set \mathcal{S} . The goal is to find a pairwise classifier $c: \mathcal{X}^{(2)} \rightarrow \{0, 1\}$ which minimizes the expected importance weighted loss, $E_{\mathcal{S} \sim P} \sum_{i < j} w(y_i, y_j, c) \cdot [\mathbf{1}(y_i > y_j) \cdot c(x_i, x_j)]$, where $w(y_i, y_j, c)$ denotes the importance weight for the ordered pair (x_i, x_j) , and $c(x_i, x_j) = 1$ if c prefers x_j to x_i and 0 otherwise.

Definition 2. A NDCG ranking problem is defined by a distribution P over the set \mathcal{S} . The goal is to find a ranker $h: \mathcal{S} \rightarrow \pi$ which maximizes the expected NDCG gain, $E_{\mathcal{S} \sim P} Z_n \cdot \sum_{j=1}^n a(j) \cdot b(j)$, where j denotes the index of input instances, $a(j) = 2^{y_j} - 1$ and $b(j) = 1/\log_2(1 + \pi_j)$.

Note that the number of relevance levels is usually determined in advance, while there is no need to fix the size of input instance sequence in our analysis. In addition, the query variables which are not explicitly delivered, may still have underlying effect on the joint distribution P .

The procedure of NDCG reduction is composed of two components. During the training phase, NDCG-Train (Procedure 1), takes as input a set of instances to be ordered in \mathcal{S} . On the basis of a ranking function, any linear ordering can be expressed as a collection of preference relations of type $\mathcal{S}^{(2)}$ on this issue. The induced learning problem when encoded with the information about the loss of different pairs is to predict, given an arbitrary pair of instances, whether one instance should be ranked above or below the other. Thus, the induced distribution over $\mathcal{S}^{(2)}$ can be defined by drawing an importance weighted pair from \mathcal{S} which

Procedure 1 NDCG-Train (labeled set \mathcal{S} , importance weighted pairwise learning algorithm \mathcal{A})

Initialize a pairwise classifier c_0 .

for $t = 1, 2, \dots, T$:

Set $\mathcal{S}' = \emptyset$.

for all pairs $(x_i, x_j) \in \mathcal{X}^{(2)}$ with $1 \leq i < j \leq n$:

Add an importance weighted ordered pair $((x_i, x_j), \mathbf{1}(y_i > y_j), w(y_i, y_j, c_{t-1}))$ to \mathcal{S}' .

end for

Let $c_t = \mathcal{A}(\mathcal{S}')$.

end for

Return c_T .

Procedure 2 NDCG-Test (unlabeled set \mathcal{U} , pairwise classifier c_T)

For $x_i \in \mathcal{U}$, let

$pos(x_i) = |\{x_j : c_T(x_i, x_j) = 1, x_j \in \mathcal{U}\}| + 1$.

Sort \mathcal{U} in ascending order of $pos(x_i)$.

is randomly drawn from an underlying distribution P . We can represent the induced distribution as NDCG-Train(P).

In fact, given a pairwise classifier, the importance weight is specified as

$$w(y_i, y_j, c) = Z_n \cdot |(a(i) - a(j)) \cdot (b(i) - b(j))| \quad (3)$$

where $a(i) = 2^{y_i} - 1$, $b(i) = 1/\log_2(1 + pos(x_i))$, $pos(x_i) = |\{x_j : c(x_i, x_j) = 1, x_j \in \mathcal{X}^{(n)}\}| + 1$ and Z_n is the normalization factor.

During the test phase, NDCG-Test (Procedure 2), applies the pairwise classifier c_T constructed in Procedure 1 to the unordered set \mathcal{U} . The ranking position of one instance can be obtained by accumulating the times that each of the other instances in the set would beat it under the pairwise contest.

3.3 The Main Theorem

Before presenting the theorem in point, we will give the definition of the regret at first. *Regret* is the difference between the incurred loss(gain) and the lowest(highest) achievable loss(gain) on the problem, which is commonly used to analyze the prediction accuracy of algorithms involving online learning [13] and reduction algorithm [3]. Unlike the sample complexity analysis, regret analysis for a reduction algorithm does not depend on any assumptions about the way that samples are generated, but focuses on the relative error guarantee between the subproblem and the original problem. When the error transformation is bounded independently of the number of instances, the reduction algorithm can be regarded as *robust* [3]. Regret analysis is applicable, especially when there are arbitrary high-order dependencies between instances, such as ranking to be discussed below.

The loss of the importance weighted pairwise classifier c on a set $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ is defined as

$$l(c, \mathcal{S}) = \sum_{i < j} w(y_i, y_j, c) \cdot [\mathbf{1}(y_i > y_j) \cdot c(x_i, x_j)] \quad (4)$$

Given any distribution P on \mathcal{S} , the expected loss of the importance weighted pairwise classifier c is given by

$$L(c, P) = E_{\mathcal{S} \sim P} l(c, \mathcal{S}) \quad (5)$$

Thus, the regret of c on P is defined as

$$r(c, P) = L(c, P) - \min_{c^*} L(c^*, P) \quad (6)$$

The importance weighted classifier with minimum expected loss is referred to as the "Bayes optimal pairwise classifier" in this paper. Similarly, taking \mathcal{S} as input, the gain of the ranker h measured by NDCG is defined as

$$g(h, \mathcal{S}) = Z_n \cdot \sum_{j=1}^n a(j) \cdot b(j) \quad (7)$$

The notations of the right-hand in equation (6) are defined as before. Given any distribution P on \mathcal{S} , the expected gain of the ranker h measured by NDCG is given by

$$G(h, P) = E_{\mathcal{S} \sim P} g(h, \mathcal{S}) \quad (8)$$

Thus, the regret of h on P is defined as

$$r(h, P) = \max_{h^*} G(h^*, P) - G(h, P) \quad (9)$$

The ranker with maximum NDCG expected gain is described as the "Bayes optimal ranker" in this paper. As a pairwise classifier acts as the ranker, we are interested in whether a small importance weighted pairwise regret leads to a small NDCG regret.

THEOREM 1. *For all pairwise learning algorithms \mathcal{A} and importance weighted datasets \mathcal{S}' , let $c = \mathcal{A}(\mathcal{S}')$. Then for all distributions P ,*

$$r(\text{NDCG} - \text{Test}(\cdot, c), P) < r(c, \text{NDCG} - \text{Train}(P)).$$

The proof of the theorem is given in the appendix.

This theorem states that the NDCG regret is bounded by the importance weighted pairwise regret, which implies that good performance on the induced importance weighted learning problem can be robustly transferred to that on the ranking problem measured by NDCG.

With the error transformation guarantee, our goal in solving ranking problem is to find a predictor $f(x)$ minimizing the expected loss below.

$$L(f(\cdot)) = E_{\mathcal{S} \sim P} l(f(\cdot), \mathcal{S}) \quad (10)$$

Given a set of training examples S_1, S_2, \dots, S_m , independently drawn from P , we generally consider the minimization of f with respect to the following empirical loss

$$\sum_{i=1}^m l(f(\cdot), \mathcal{S}_i) \quad (11)$$

Observe that without the importance weight terms, various developed pairwise classification methods can serve the purpose with sound theoretical basis.

In order to obtain the values of importance weights, the ordering information is encoded, which challenges the common convex optimization strategies. We approach this issue by simply specifying the sub-gradients of the loss which combines importance weights with the pairwise loss functions of general interest after the sort, and then updating.

4. RSRANK

As can be seen above, the NDCG reduction gives us a simple and quite general way to cope with rank learning problems evaluated from the user point of view. To learn an

effective ranking function, in this section, we follow such a reduction manner and further devise a novel ranking algorithm referred to as RSRank.

4.1 Objective Function

Reducing ranking to importance weighted pairwise classification, each ordered pair is assigned with an importance value which measures how important it might be in comparison with the other ones generated from a permutation. Hence, given a set of instances with cardinality n , written as \mathcal{S} , we measure the loss as follows,

$$l(f(\cdot), \mathcal{S}) = \sum_{y_i > y_j} w(y_i, y_j, f(\cdot)) \cdot \mathbf{1}(f(x_i) \leq f(x_j)) \quad (12)$$

where $w(y_i, y_j, f(\cdot))$ is the importance weight as defined in equation (3) except for a tiny difference in notations.

$$w(y_i, y_j, f(\cdot)) = Z_n \cdot (2^{y_i} - 2^{y_j}) \cdot \left(\frac{1}{\log_2(1 + \pi_j)} - \frac{1}{\log_2(1 + \pi_i)} \right)$$

Note that Z_n is defined as before and a permutation π can be obtained in descending order of $f(\cdot)$.

We then follow the idea of classification by substituting the indicator function $\mathbf{1}(\cdot)$ with a well-behaved loss function. Intuitively, for the ordered pair $\langle (x_i, x_j), \mathbf{1}(y_i > y_j) \rangle$, the ranking function f should receive no punishment when matching the given preference well. Otherwise, there is no need to exert too much punishment for the rare cases when f is quite inconsistent with the preference. To achieve these ends, we take the modified Huber loss function[21] as our choice, assuming that $y_i > y_j, v = f(x_i) - f(x_j)$, we have

$$\phi(v) = \begin{cases} -4v, & v < -1, \\ (v-1)^2, & v \in [-1, 1] \\ 0, & v > 1 \end{cases} \quad (13)$$

By replacing the indicator function in equation (12) with the above modified Huber loss, we get the loss function

$$l(f(\cdot), \mathcal{S}) = \sum_{y_i > y_j} w(y_i, y_j, f(\cdot)) \cdot \phi(f(x_i) - f(x_j)) \quad (14)$$

In this paper, we assume that f is a linear combination of model parameters, i.e., $f(x_i) = \langle \beta, x_i \rangle$, where $\beta \in \mathbb{R}^d$ is a model parameter vector, and $\langle \cdot, \cdot \rangle$ denotes the inner product.

Statistically, the solution of β tends to overfit the training examples without a regularization condition which imposes constraints on the parameter space. One important constraint is sparsity with desirable properties in practice. 1-norm regularization which often leads to sparse solutions is applied in our model. Given a set of training examples S_1, S_2, \dots, S_m , the objective function is formulated as

$$\min_{\beta} \sum_{i=1}^m l(\beta, \mathcal{S}_i) + g \|\beta\|_1 = \min_{\beta} L(\beta) + g \|\beta\|_1 \quad (15)$$

where g is a non-negative regularization parameter making a tradeoff between the loss on the training set and the penalty on model parameters.

4.2 Algorithm

In order to solve (14), the sub-gradients of the loss, expressed as $\nabla L(\beta)$, are calculated after each sort, yet simply adding L_1 regularization to the gradients cannot lead to sparsity. Therefore, we further devise the following ranking

algorithm, referred to as RSRank, which performs L_1 regularization using truncated gradient descent for the sake of achieving sparsity.

Algorithm 1 RSRank Algorithm

Inputs:

- training examples: $\mathcal{S}_i = \{(x_1^{(i)}, y_1^{(i)}), \dots, (x_n^{(i)}, y_n^{(i)})\}$, ($i = 1, 2, \dots, m$)
- regularization parameter $g \geq 0$
- truncation step interval K
- learning rate $\eta \in (0, 1)$

Initialize model parameter vector $\beta = [\beta_1, \dots, \beta_d] \in \mathbb{R}^d$

for $t = 1, 2, \dots$

 for $i = 1, 2, \dots, m$

1. Sort instances in \mathcal{S}_i in descending order according to β
2. Compute the importance weights for each ordered pair in \mathcal{S}_i

 end for

 Update the model parameter vector as

$$\beta = \beta - \eta \cdot \nabla L(\beta)$$

 if t/K is an integer, then

 for $j = 1, \dots, d$

- if $\beta_j > 0$, then $\beta_j = \max(\beta_j - \eta g, 0)$
- else if $\beta_j < 0$, then $\beta_j = \min(\beta_j + \eta g, 0)$

 end for

 end if

end for

The main idea that motivates RSRank algorithm is derived from the work [13] in which truncated gradient is initially proposed to induce sparsity for online learning algorithms. When applied into our algorithm focusing on a desirable gradient for a given sorted list, it is also effective. We will demonstrate its validity in the following experiments.

5. EXPERIMENTS

In this section, we evaluate the proposed algorithm on OHSUMED and TD2003 data in Leter3.0 benchmark collection released lately[1]. The goals of our experiments mainly include two aspects where one is to compare our algorithm without sparsification to the existing baseline results and the other is to examine the efficiency of our algorithm with sparsity option.

5.1 Datasets and Baselines

The OHSUMED dataset collected from medical publications contains 106 queries and 16,140 query-document pairs in total. For each query-document pair, there are 45 ranking features extracted and a 3-level relevance judgement provided, i.e. definitely relevant, possibly relevant or not relevant. In the TD2003 dataset constructed from web entry pages in topic distillation tasks, altogether 50 queries and 49,171 query-document pairs are included. Each query-document pair comprises 64 ranking features and a binary relevance judgement, i.e. relevant or not relevant.

For each dataset, we measured the performance of our algorithm averaged over five folds off-the-shelf, each of which consists of training, validation, and test set. The validation set was used to identify the best set of parameters, including the initial model parameter vector β , the regularization parameter g , and the number of iterations t , which were then

verified on the test set. We fixed the learning rate $\eta = 0.001$ and the truncation step interval $K = 1$ in these experiments.

Several common baseline algorithms were chosen as comparisons to our algorithm without sparsification, including LambdaRank[6], AdaRank.NDCG[18] and ListNet[7]. [6] and [18] allow direct optimization of NDCG criteria, while [7] attempts to optimize a listwise loss function. In addition, the ranking functions that these baseline methods target at are of linear relationship with the model parameters, the same as our method.

5.2 Results

5.2.1 comparison with baselines

In the first set of experiments, we are interested in how our ranking algorithm performs, which is based on a robust reduction mechanism in theory, and no sparsification is considered, i.e. the parameter g is set to zero. The experimental results on OHSUMED and TD2003 dataset are illustrated in Figure 1-2.

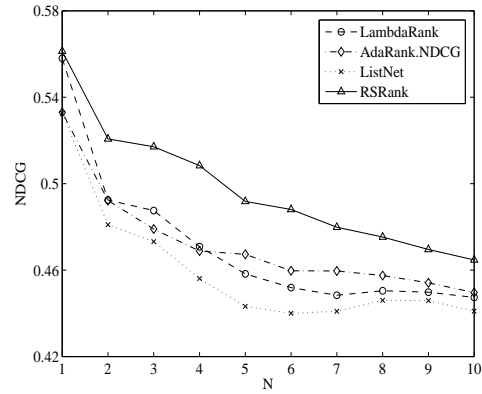


Figure 1: NDCG at top 10 on OHSUMED dataset

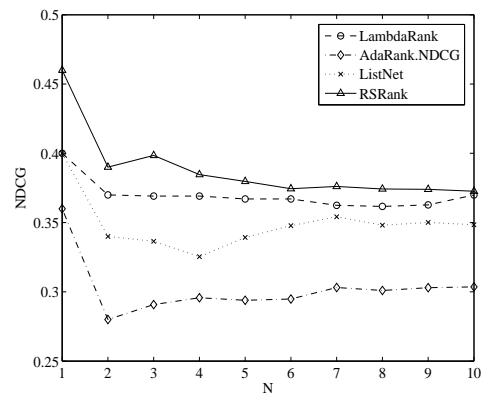


Figure 2: NDCG at top 10 on TD2003 dataset

The above two figures show that RSRank consistently outperforms the reference baseline algorithms in terms of NDCG at top 10. For OHSUMED dataset, the improvement of RSRank over LambdaRank is also statistically significant for NDCG@3 to NDCG@10 with a p-value of less than 0.032

in t-test, which indicates that the excellent effectiveness of our ranking approach following the NDCG reduction practice. However, for TD2003 dataset, RSRank behaves a bit differently. By comparison with LambdaRank, the performance of RSRank is not superior statistically in terms of t-test, in spite of rising around 6 percent on NDCG@1. It is yet noteworthy that our algorithm still keep obvious advantages over AdaRank.NDCG and ListNet.

5.2.2 feature sparsification

The next set of experiments is designed to demonstrate the behavior of RSRank when working with sparse ranking models. First and foremost, we are concerned about how much degree of sparsity can be achieved without affecting ranking performance significantly. To this end, we performed RSRank from the same initial model parameters in one fold simply changing the setting of regularization parameter. Table 1 gives the results on the fraction of features left after sparsification. It is interesting to observe that the number of features is reduced by a fraction of more than 60% on average, while the accuracy loss in NDCG at top 10 decreases by no more than 3% on average, as shown in Figure 3.

Table 1: Fraction of features left after sparsification for OHSUMED and TD2003 datasets when NDCG at top 10 is changed by less than 3% on average.

Dataset	OHSUMED	TD2003
fraction of features left	0.36888	0.34062

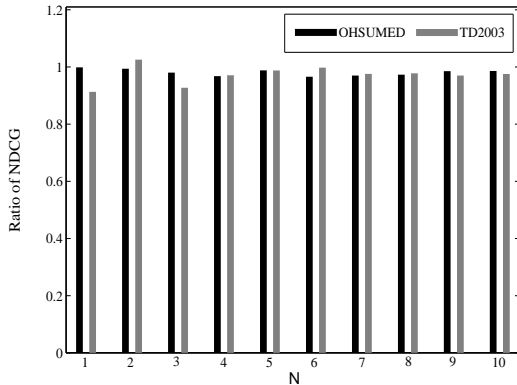


Figure 3: The ratio of the NDCG at top 10 when sparsification is used over the case when no sparsification is used on OHSUMED and TD2003 datasets.

Specifically, for OHSUMED dataset, the NDCG metrics on the test set drop at most 3.5% with more than 63% of features removed, indicating that only a fraction of features are enough for ranking at a slight cost of prediction accuracy loss. Especially, the NDCG value with sparsification hits its peak value of 0.5605 at the first rank position in comparison with 0.5613 without sparsity option, which is still comparable to the reference baseline algorithms. Meanwhile, for TD2003 dataset with more ranking features added, a qualitatively similar observation can be derived. Besides, we found that the NDCG criteria at the second rank position

was actually improved from 0.39 to 0.40 due to removal of some redundant features. Better results can be expected if more extensive parameters are tried in cross validation.

The other aspect we are concerned about is the influence of regularization parameter g when performing truncated gradient. Given a regularization parameter g , the initial weight parameter vector that performed best on the validation set was evaluated on the test set. Figure 4 and 5 plot the curves on NDCG criteria at the tenth rank position and the number of features selected. Based on the experimental results, we can make the following observations.

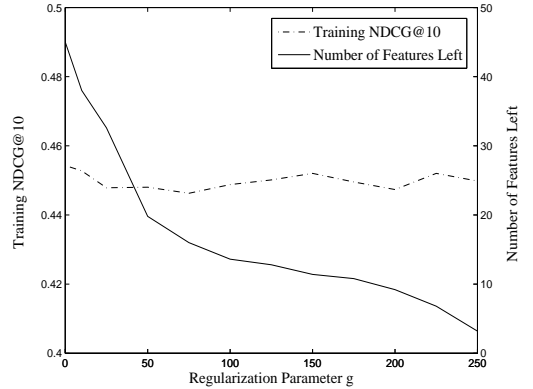


Figure 4: The trajectories of training NDCG@10 and number of nonzero coefficients versus regularization parameter g on OHSUMED training dataset

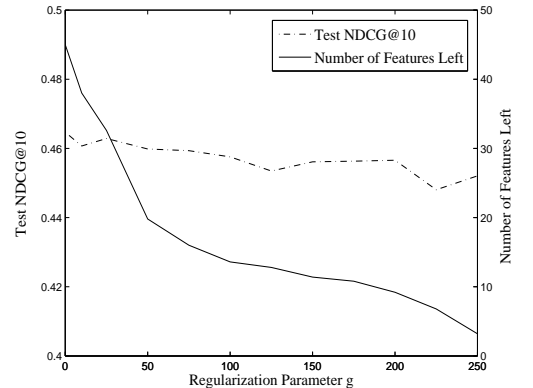


Figure 5: The trajectories of test NDCG@10 and number of nonzero coefficients versus regularization parameter g on OHSUMED dataset

First, more large regularization parameter g is, more degree of sparsity we can achieve. When g is set to zero, the linear composition of all the ranking features is used for ranking prediction. When g is set to 250, only 3 features or so exercise the power of ranking prediction.

Second, as the regularization parameter g increases, the variations of the evaluation measures in terms of NDCG@10 hold within a range of less than 0.75% for training set and less than 2% for test set. In other words, better sparsity leads to a approximately equal level of performance

maintained on OHSUMED dataset, which shed light on the strength of our ranking algorithm in representing sparse models for ranking.

6. CONCLUSIONS

In this paper, we introduced a robust and sparse mechanism to deal with the rank learning problem under the NDCG criteria. After describing an efficient reduction of ranking to importance weighted pairwise classification, we bound the NDCG performance in terms of the induced pairwise classification performance. With the aim of achieving sparsity in ranking models, we then constructed the objective function composed of the converted pairwise loss function and an additional 1-norm regularization penalty. We also developed a novel ranking algorithm via truncated gradient, called RSRank. Empirical studies on benchmark collection justified the effectiveness of the proposed algorithm.

In future work, we will continue to study the reduction-based model for ranking measured by other non-smooth ranking measures besides NDCG, also try other sparse learning methods in rank learning problems, and pursue further theoretical investigation on the ability of algorithms to produce sparse ranking models.

7. ACKNOWLEDGMENTS

This work was supported partially by 973 Program (No. 2004CB318103), NSFC Projects (No. 60835002 and 60621001) and by Microsoft Research Asia Funding.

8. REFERENCES

- [1] M. R. Asia. Letor3.0: benchmark datasets for learning to rank. Microsoft Corporation, December 2008.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
- [3] M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. Sorkin. Robust reductions from ranking to classification. *Machine Learning*, 72(1-2):139–153, August 2008.
- [4] A. Beygelzimer, V. Dani, T. Hayes, and J. Langford. Reductions between classification tasks. *Electronic Colloquium on Computational Complexity*, 11, 2004.
- [5] A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the 22th International Conference on Machine Learning*, pages 49–56. ACM, 2005.
- [6] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19*, pages 193–200. MIT Press, 2007.
- [7] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136. ACM, 2007.
- [8] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 88–96. ACM, 2008.
- [9] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS Workshop on Machine Learning for Web Search*, 2007.
- [10] J. Guiver and E. Snelson. Learning to rank with softrank and gaussian processes. In *Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in IR*, pages 259–266. ACM, 2008.
- [11] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002.
- [12] J. Langford and A. Beygelzimer. Sensitive error correcting output codes. *Lecture Notes in Computer Science*, 3559:158–172, June 2005.
- [13] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *CoRR*, abs/0806.4686, 2008.
- [14] Q. Le and A. Smola. Direct optimization of ranking measures. *CoRR*, abs/0704.3359, 2007.
- [15] T. Qin, T.-Y. Liu, and H. Li. A general approximation framework for direct optimization of information retrieval measures. MSR-TR-2008-164, Microsoft Research, 2008.
- [16] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the International Conference on Web Search and Web Data Mining*, pages 77–86. ACM, 2008.
- [17] E. Voorhees. Overview of the trec 2002 question answering track. In *Proceedings of the Eleventh Text Retrieval Conference (TREC)*, pages 115–123, 2002.
- [18] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in IR*, pages 391–398. ACM, 2007.
- [19] J.-Y. Yeh, J.-Y. Lin, H.-R. Ke, and W.-P. Yang. Learning to rank for information retrieval using genetic programming. In *LR4IR*, 2007.
- [20] Y. Yue and C. Burges. On using simultaneous perturbation stochastic approximation for learning to rank, and the empirical optimality of lambdarank. MSR-TR-2007-115, Microsoft Research, 2007.
- [21] T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.
- [22] T. Zhang. Some sharp performance bounds for least squares regression with l_1 regularization. Technical Report TR-2007-005, Rutgers Statistics Department, 2007.
- [23] P. Zhao and B. Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, December 2006.

APPENDIX

Proof of Theorem 1

Given an unlabeled test set \mathcal{U} , the joint distribution P induces a conditional distribution $P(y_1, y_2, \dots, y_n | \mathcal{U})$ over the set of relevance label sequences $\{r_1, r_2, \dots, r_n\}^n$, denoted as D . In the following, we identify \mathcal{U} with $\{1, 2, \dots, n\}$. Due

to the existence of a ranking function, any pairwise classifier can equally induce a permutation. The following lemma is the essential step in our analysis.

Lemma 1. For all conditional distributions D , suppose that the pairwise classifier c induces an imperfect permutation π , while the pairwise classifier c' induces a perfect permutation π' , then the following relationship holds,

$$G(\pi', D) - G(\pi, D) < L(c, NDCG - Train(D)) - L(c', NDCG - Train(D))$$

Note that there is generally more than one perfect permutation, and we make no differential treatment of notations for simplicity.

PROOF. For any imperfect permutation π with inversion pairs of $k \in \mathbb{N}$, there exists at least one inversion pair (π_i, π_j) , through swapping which the number of inversion pairs decreases only by one. Thus, π can be transposed into a perfect permutation π' via k such operations denoted as \circ .

This allows us to decompose the left hand of the above inequality in the following way. Let $\pi^{(0)} = \pi$, and $\pi^{(1)} = \pi^{(0)} \circ (\pi_{i_0} \pi_{j_0})$, $i_0 < j_0$, then

$$\begin{aligned} G(\pi^{(1)}, D) - G(\pi^{(0)}, D) &= E_{y^n \sim D} [g(\pi^{(1)}, \mathcal{U})] - E_{y^n \sim D} [g(\pi^{(0)}, \mathcal{U})] \\ &= E_{y^n \sim D} Z_n \cdot \left[\left(\frac{2^{r(i_0)} - 1}{\log_2(1 + \pi_{j_0}^{(0)})} + \frac{2^{r(j_0)} - 1}{\log_2(1 + \pi_{i_0}^{(0)})} \right) \right. \\ &\quad \left. - \left(\frac{2^{r(i_0)} - 1}{\log_2(1 + \pi_{i_0}^{(0)})} + \frac{2^{r(j_0)} - 1}{\log_2(1 + \pi_{j_0}^{(0)})} \right) \right] \\ &= E_{y^n \sim D} Z_n \cdot \left[\left(2^{r(j_0)} - 2^{r(i_0)} \right) \right. \\ &\quad \left. \cdot \left(\frac{1}{\log_2(1 + \pi_{i_0}^{(0)})} - \frac{1}{\log_2(1 + \pi_{j_0}^{(0)})} \right) \right] \end{aligned}$$

For permutation $\pi^{(1)}$, suppose that swapping inversion pair $(\pi_{i_1}^{(1)} \pi_{j_1}^{(1)})$ incurs the number of inversions decrease by one. Let $\pi^{(2)} = \pi^{(1)} \circ (\pi_{i_1} \pi_{j_1})$, $i_1 < j_1$, then

$$\begin{aligned} G(\pi^{(2)}, D) - G(\pi^{(1)}, D) &= E_{y^n \sim D} [g(\pi^{(2)}, \mathcal{U})] - E_{y^n \sim D} [g(\pi^{(1)}, \mathcal{U})] \\ &= E_{y^n \sim D} Z_n \cdot \left[\left(\frac{2^{r(i_1)} - 1}{\log_2(1 + \pi_{j_1}^{(1)})} + \frac{2^{r(j_1)} - 1}{\log_2(1 + \pi_{i_1}^{(1)})} \right) \right. \\ &\quad \left. - \left(\frac{2^{r(i_1)} - 1}{\log_2(1 + \pi_{i_1}^{(1)})} + \frac{2^{r(j_1)} - 1}{\log_2(1 + \pi_{j_1}^{(1)})} \right) \right] \\ &= E_{y^n \sim D} Z_n \cdot \left[\left(2^{r(j_1)} - 2^{r(i_1)} \right) \right. \\ &\quad \left. \cdot \left(\frac{1}{\log_2(1 + \pi_{i_1}^{(1)})} - \frac{1}{\log_2(1 + \pi_{j_1}^{(1)})} \right) \right] \end{aligned}$$

Similarly, the rest may be deduced by analogy.

$$\begin{aligned} &\vdots \\ G(\pi', D) - G(\pi^{(k-1)}, D) &= E_{y^n \sim D} Z_n \cdot \left[\left(2^{r(j_{k-1})} - 2^{r(i_{k-1})} \right) \right. \\ &\quad \left. \cdot \left(\frac{1}{\log_2(1 + \pi_{i_{k-1}}^{(k-1)})} - \frac{1}{\log_2(1 + \pi_{j_{k-1}}^{(k-1)})} \right) \right] \end{aligned}$$

Whence, telescoping the sum of the above equations, we obtain

$$\begin{aligned} G(\pi', D) - G(\pi, D) &= E_{y^n \sim D} Z_n \cdot \sum_{m=0}^{k-1} \left[\left(2^{r(j_m)} - 2^{r(i_m)} \right) \right. \\ &\quad \left. \cdot \left(\frac{1}{\log_2(1 + \pi_{i_m}^{(m)})} - \frac{1}{\log_2(1 + \pi_{j_m}^{(m)})} \right) \right] \end{aligned}$$

As to the right hand of the inequality, it can also be expressed in terms of inversion pairs.

$$\begin{aligned} L(c, NDCG - Train(D)) - L(c', NDCG - Train(D)) &= E_{y^n \sim D} \sum_{i < j} w(y_i, y_j, c) \cdot [\mathbf{1}(y_i > y_j) \cdot c(x_i, x_j)] \\ &\quad - E_{y^n \sim D} \sum_{i < j} w(y_i, y_j, c') \cdot [\mathbf{1}(y_i > y_j) \cdot c'(x_i, x_j)] \\ &= E_{y^n \sim D} \sum_{i < j} w(y_i, y_j, c) \cdot [\mathbf{1}(y_i > y_j) \cdot c(x_i, x_j)] \\ &= E_{y^n \sim D} Z_n \cdot \sum_{m=0}^{k-1} \left[\left(2^{r(j_m)} - 2^{r(i_m)} \right) \right. \\ &\quad \left. \cdot \left(\frac{1}{\log_2(1 + \pi_{i_m})} - \frac{1}{\log_2(1 + \pi_{j_m})} \right) \right] \end{aligned}$$

Observe that although the both sides of the inequality share the same inversion pairs $(i_0, j_0), (i_1, j_1), \dots, (i_{k-1}, j_{k-1})$, the position deviation between elements in each inversion is different. It is easy to see that the following inequalities hold, $\pi_{j_m}^{(m)} = \pi_{i_m}^{(m)} + 1$, $\pi_{i_m} \leq \pi_{i_m}^{(m)}$, and $\pi_{j_m} \geq \pi_{i_m} + 1$. Hence,

$$\begin{aligned} &\frac{1}{\log_2(1 + \pi_{i_m}^{(m)})} - \frac{1}{\log_2(1 + \pi_{j_m}^{(m)})} \\ &= \frac{1}{\log_2(1 + \pi_{i_m}^{(m)})} - \frac{1}{\log_2(1 + \pi_{i_m}^{(m)} + 1)} \\ &< \frac{1}{\log_2(1 + \pi_{i_m})} - \frac{1}{\log_2(1 + \pi_{i_m} + 1)} \\ &\leq \frac{1}{\log_2(1 + \pi_{i_m})} - \frac{1}{\log_2(1 + \pi_{j_m})} \end{aligned}$$

The first inequality follows from the monotone of the function $1/\log_2(1+x) - 1/\log_2(2+x)$. Summing over $m = 1, 2, \dots, k-1$, we get the lemma. \square

PROOF. (of theorem 1) Note that we represent NDCG-Train(D) as N-T(D) in short. In terms of the definition of regret with respect to the ranker h and the pairwise classifier c aforementioned, we have

$$\begin{aligned} r(h, D) &= \max_{h^*} G(h^*, D) - G(h, D) \\ &= [G(h', D) - G(h, D)] - [G(h', D) - \max_{h^*} G(h^*, D)] \\ &= [G(\pi', D) - G(\pi, D)] - [G(\pi', D) - \max_{\pi^*} G(\pi^*, D)] \\ &< [L(c, N - T(D)) - L(c', N - T(D))] \\ &\quad - [\min_{c^*} L(c^*, N - T(D)) - L(c', N - T(D))] \\ &= L(c, N - T(D)) - \min_{c^*} L(c^*, N - T(D)) \\ &= r(c, N - T(D)) \end{aligned}$$

The first inequality follows by applying Lemma 1 to obtain the desired bound. \square