Inf Retrieval (2010) 13:216–235
DOI 10.1007/s10791-009-9110-3

Gradient descent optimization of smoothed information retrieval metrics

Olivier Chapelle · Mingrui Wu

Received: 27 March 2009 / Accepted: 18 August 2009 / Published online: 10 September 2009

Abstract Most ranking algorithms are based on the optimization of some loss functions, such as the pairwise loss. However, these loss functions are often different from the criteria that are adopted to measure the quality of the web page ranking results. To overcome this problem, we propose an algorithm which aims at directly optimizing popular measures such as the Normalized Discounted Cumulative Gain and the Average Precision. The basic idea is to minimize a smooth approximation of these measures with gradient descent. Crucial to this kind of approach is the choice of the smoothing factor. We provide various theoretical analysis on that choice and propose an *annealing* algorithm to iteratively minimize a less and less smoothed approximation of the measure of interest. Results on the Letor benchmark datasets show that the proposed algorithm achieves state-of-the-art performances.

Keywords Learning to rank · Gradient descent · Annealing

1 Introduction

Web page ranking has traditionally been based on a manually designed ranking function such as BM25 (Robertson and Walker 1994). However, ranking is currently considered as a supervised learning problem and several machine learning algorithms have been applied to it (Freund et al. 2003; Burges et al. 2005; Cossock and Zhang 2006; Cao et al. 2006; Burges et al. 2007; Xu and Li 2007; Tsai et al. 2007; Cao et al. 2007).

In these methods, the training data is composed of 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,

O. Chapelle (✉) · M. Wu
Yahoo! Labs, Santa Clara, CA, USA
e-mail: chap@yahoo-inc.com

M. Wu
e-mail: mingrui@yahoo-inc.com

$$\{\text{perfect, excellent, good, fair, bad}\} \quad (1)$$

and is assigned by human editors. The label can also simply be binary: relevant or irrelevant. Each query and each of its documents are paired together, and each query-document pair is represented by a feature vector. Thus the training data can be formally represented as: $\{(\mathbf{x}_j^q, l_j^q)\}$, where q goes from 1 to n, the number of queries, j goes from 1 to m_q, the number of documents for query q, $\mathbf{x}_j^q \in \mathbb{R}^d$ is the d-dimensional feature vector for the pair of query q and the j-th document for this query while l_j^q is the relevance label for $\mathbf{x}_j^q$; $l_j^q \in \{0, 1\}$ in the case of binary relevance or it can belong to an ordinal set such as (1).

In order to measure the quality of a search engine, we need some evaluation metrics. The *Discounted Cumulative Gain* (DCG) has been widely used to assess relevance in the context of search engines (Jarvelin and Kekalainen 2002) because it can handle multiple relevance grades such as (1). In the case of binary relevance, *Average Precision* is a very commonly used metric. Therefore, when constructing a learning to rank approach, it is important to consider how to optimize model parameters with respect to these metrics during the training procedure. Many machine learning algorithms apply the gradient based techniques for parameter optimization. Unfortunately IR measures are not continuous and it is not possible to directly optimize them with gradient based approaches.[1] Consequently many current ranking algorithms turn to optimize other objectives that can be divided into three categories:

Pointwise The objective function is of the form $\sum_{q,j} \ell(f(\mathbf{x}_j^q), l_j^q)$ where ℓ can for instance be a regression loss (Cossock and Zhang 2006) or a classification loss (Li et al. 2008).

Pairwise Methods in this category try to order correctly pairs of documents by minimizing

$$\sum_q \sum_{i,j, l_i^q > l_j^q} \ell(f(\mathbf{x}_i^q) - f(\mathbf{x}_j^q)).$$

RankSVM (Herbrich et al. 2000; Joachims 2002) uses $\ell(t) = \max(0, 1 - t)$, while RankNet (Burges et al. 2005) uses $\ell(t) = \log(1 + \exp(-t))$. GBRank (Zheng et al. 2008) is similar to RankSVM, but uses a quadratic penalization, $\ell(t) = \max(0, 1 - t)^2$ and is combined with functional gradient boosting. Finally in LambdaRank (Burges et al. 2007), the weight of each preference pair is the NDCG difference resulting from swapping that pair.

Listwise The loss function is defined over all the documents associated with the query, $\ell(\{f(\mathbf{x}_j^q)\}, \{l_j^q\})$ for $j = 1,\ldots,m_q$. This type of algorithms can further be divided into two different sub-categories. The first one ignores the IR measure during training: for instance ListNet (Cao et al. 2007) and ListMLE (Xia et al. 2008) belong to this category. The second sub-category tries to optimize the IR measure during training: examples include AdaRank (Xu and Li 2007), several algorithms based on structured estimation (Chapelle et al. 2007; Yue et al. 2007; Chakrabarti et al. 2008) and SoftRank (Taylor et al. 2008) which minimizes a smooth version of the NDCG measure.

The approach presented in this paper is similar to SoftRank (Taylor et al. 2008) as it also minimizes by gradient descent a smooth approximation of IR measures. There are however three major differences:

1. The smoothing approach is different and more general. Also the gradient of the smooth function can be computed in $O(m^2)$ versus $O(m^3)$ for SoftRank where m is the number of documents per query.

[1] When the number of queries is large, the average IR metric becomes approximately smooth and it is possible to compute an empirical gradient (Yue and Burges 2007).

2. We carefully discuss the choice of the smoothing parameter and propose an annealing schedule on that parameter to ease the optimization and alleviate the local minima problem.
3. We derive a generalization error bound as a function of the smoothing parameter which suggests that for better generalization, the smoothing should not be too small.

Regarding the last two points, we would like to stress that the choice of the smoothing parameter is critical for any learning algorithm which attempts at minimizing a smoothed approximation of a given loss/measure. It is indeed a trade-off between the following observations:

1. If the smoothing constant is too large, the objective function is easy to optimize but quite different from the one we want to optimize. We will discuss this approximation error in Sect. 5.1.
2. However, if it is too small, the function becomes very difficult to optimize since the gradient is close to zero almost everywhere.
3. In addition, if it is too small, the risk of overfitting becomes higher as we will see in Sect. 5.2.

To address points 1 and 2, we devise an annealing procedure: the smoothing parameter is initially set to a large value and then reduced iteratively.

Finally, the ranking problem is, in our opinion, more challenging when there are more than two relevance levels. For this reason, in the rest of the paper, we will mostly focus on NDCG as an evaluation metric (because AP can only handle binary relevance) and compare algorithms on the Ohsumed dataset from the Letor benchmark (because it is the only one with more than 2 levels of relevance).

2 Information retrieval measures

In this section, two of the most common information retrieval measures are introduced, namely NDCG and AP. For convenience, we just describe these measures based on a single query but in practice they are averaged over a set of queries.

2.1 Normalized discounted cumulative gain

The Discounted Cumulative Gain (DCG) score (Jarvelin and Kekalainen 2002) is a popular measure for multi-level relevance judgments. In its basic form it has a logarithmic position discount: the benefit of seeing a relevant document at position i is $1/\log_2(i+1)$. Following (Burges et al. 2005), it became usual to assign exponentially high weight 2^{l_i} to highly rated documents where l_i is the grade of the ith document going for instance from 0 (irrelevant) to 4 (perfect result). Thus, the DCG for a ranking function r of a query having m associated documents is defined as:

$$\mathrm{DCG} := \sum_{i=1}^{m} \frac{2^{l_i} - 1}{\log_2(1 + r(i))},$$

where $r(i)$ is the position (or rank) of the ith document in the ranking.

One can also define the *normalized* DCG score by dividing the DCG score in such a way that the best ranking has a score of 1. Let us define the *gain* function $G(l) = 2^l - 1$ and the *discount* function $D(r) = 1/\log_2(1 + r)$. Then the DCG can be rewritten as:

$$\text{(N)DCG} := \sum_{i=1}^{m} G(l_i)D(r(i)), \tag{2}$$

The NDCG can be expressed in the same form as the last equation, while the discount function is redefined as $D(r) \leftarrow D(r)/Z$ with Z being the DCG obtained with the best ranking.

Finally, one can also define the (N)DCG at a given *truncation* level k, denoted by (N)DCG@k by ignoring the documents after rank k, that is setting $D(r) = 0$ for $r > k$.

2.2 Average precision

Average precision is a commonly used metric for binary relevance judgments, $l_i \in \{0, 1\}$. Before defining it, we first need to introduce precision at k which is the proportion of relevant documents in the first k positions:

$$\text{Prec@}k := \frac{1}{k}\sum_{i=1}^{m} l_i \mathbb{1}(r(i) \le k),$$

where $\mathbb{1}$ is the indicator function: $\mathbb{1}(A) = 1$ is A is true, 0 otherwise.

Average precision is then the average of the precisions at the positions where there is a relevant document:

$$\text{AP} := \frac{1}{\sum l_i}\sum_{i=1}^{m} l_i \text{Prec@}r(i). \tag{3}$$

As we will see later, it can be useful to combine the two equations above and rewrite AP in the following way:

$$\text{AP} = \frac{1}{\sum l_i}\sum_{i,j=1}^{m} l_i l_j \frac{\mathbb{1}(r(j) \le r(i))}{r(i)}. \tag{4}$$

3 Related work

As noted in (Burges et al. 2007), optimizing IR measures is difficult because they depend on the rank and are thus piecewise constant and non-differentiable. The solution proposed in (Burges et al. 2007) is to perform gradient descent on a smoothed version of the objective function. But in that work, the objective function is never computed explicitly, but only implicitly through its gradients.

However, it is possible to explicitly construct a smoothed version of an IR metric; in particular SoftRank (Taylor et al. 2008) has been the first paper to propose a soft version of the NDCG. The idea is based on the computation of a soft rank and is as follows. Suppose that the scores $f(\mathbf{x}_i)$ of the various documents have been corrupted with some Gaussian noise with standard deviation σ. Then one can compute the *expected* NDCG under this noise assumption. This amounts to replacing $D(r(i))$ in (2) by $\mathbf{E}D(r(i)) = \sum_j P(r(i) = j)D(j)$.

However, this quantity cannot be easily computed. The authors of SoftRank then proposed an approximate way of computing $P(r(i) = j)$ by introducing the so-called *binomial-rank* distribution. There are some cases however where this approximation can be very bad. Consider for instance the case where the m documents have the same score. Then it is clear that $P(r(i) = j) = 1/m$. But the binomial rank distribution yields $P(r(i) = j) = \binom{m-1}{j-1}\Big/2^{m-1}$, which is very different from the uniform distribution, especially if m is large.

Another drawback is that the computation of the of the rank-binomial distribution for each query is $O(m^3)$.

But another approximation could have been done: because $1/\log_2(1 + t)$ is convex for $t > 0$, we have through Jensen's inequality,

$$D(\mathbf{E}r(i)) \leq \mathbf{E}D(r(i)).$$

And $\mathbf{E}r(i)$ can be computed in closed form. Indeed, $r(i) = 1 + \sum_{j \neq i} \mathbb{1}(f(\mathbf{x}_i) < f(\mathbf{x}_j))$ and taking the expectation, we have:

$$\mathbf{E}r(i) = 1 + \sum_{j \neq i} \Phi(f(\mathbf{x}_j) - f(\mathbf{x}_i)) \quad \text{with} \quad \Phi(t) := \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{t} \exp\left(-\frac{t^2}{2\sigma^2}\right) \mathrm{d}t.$$

This formulation of the expected rank has already been derived in (Taylor et al. 2008). But the authors did not use it. Instead they computed the approximate rank distributions described above. Since the goal is to obtain a smooth version of the NDCG, both approximations are arguably equally valid.

It turns out that (Wu et al. 2009) followed that idea and proposed the following smooth approximation of the DCG:

$$\sum_{i=1}^{m} G(l_i) D(\hat{r}_i) \quad \text{with} \quad \hat{r}_i := 1 + \sum_{j \neq i} \frac{1}{1 + \exp((f(\mathbf{x}_i) - f(\mathbf{x}_j))/\sigma)}. \tag{5}$$

Experiments using this smoothing combined with gradient boosted decision trees have shown promising results on a real world ranking dataset (Wu et al. 2009). We have not yet tested this method in the linear case on the Letor benchmarks but will do so in future work.

4 Our approach

In this section, we describe our learning to rank algorithm based on the idea of optimizing the smooth approximations of IR metrics.

4.1 A general smoothing framework

The intuition for the smoothing technique presented in this paper stems from the *softmax activation function* used in neural networks (Bridle 1990). More specifically, given m numbers $f_1,\ldots,f_m$, the following quantities p_i can be interpreted as probabilities that f_i is the largest number:

$$p_i := \frac{\exp(f_i/\sigma)}{\sum_{j=1}^{m} \exp(f_j/\sigma)},$$

where σ is the *smoothing* parameter. When σ goes to 0 and the argmax is unique, the p_i converge to the argmax indicator function:

$$\lim_{\sigma \to 0} p_i = \mathbb{1}(f_i = \max_j f_j). \tag{6}$$

Based on this observation, it is easy to write the following smoothed version of NDCG@1:

$$D(1) \sum_{i=1}^{m} G(l_i) \frac{\exp(f(\mathbf{x}_i)/\sigma)}{\sum_j \exp(f(\mathbf{x}_j)/\sigma)},$$

Because of (6), this quantity will converge when σ goes to 0 to:

$$D(1)G(l_{\hat{\imath}}) \quad \text{with} \quad \hat{\imath} = \arg\max_i f(\mathbf{x}_i),$$

which is the NDCG@1.

We now extend this idea: instead of having of a soft argmax, we introduce a soft version of the indicator variable: "Is document i ranked at the jth position?". We define these soft indicator variable h_{ij} as:

$$h_{ij} := \exp\left(-\frac{(f(\mathbf{x}_i) - f(\mathbf{x}_{d(j)}))^2}{\sigma}\right) \Big/ \sum_{k=1}^{m} \exp\left(-\frac{(f(\mathbf{x}_k) - f(\mathbf{x}_{d(j)}))^2}{\sigma}\right), \tag{7}$$

where $d(j)$ is the index of that document which was ranked at position j by f. Clearly the function d is the inverse of the rank function r defined above: $d(j) = i \Leftrightarrow r(i) = j$.

Proposition 1 *The variable h_{ij} defined in (7) is a continuous function of f, $\forall i, j$.*

Proof As a function of f, all the quantities in (7) are continuous except the index function d. Let us thus consider an f and a position j such that the associated $d(j)$ is discontinuous at that point. That happens when there is a tie in the ranking for the jth position. In other words, there exists two documents i_1 and i_2 for which $f(\mathbf{x}_{i_1}) = f(\mathbf{x}_{i_2})$ and $d(j) \in \{i_1, i_2\}$ in a neighborhood of f. But even though d is discontinuous at that point, the fact that $f(\mathbf{x}_{i_1}) = f(\mathbf{x}_{i_2})$ implies that h_{ij} is continuous. □

Remark 1 h_{ij} is not differentiable with respect to f at the points where there is a tie in the ranking.

The smoothing is controlled by σ and assuming that there is no tie, it is clear that

$$\lim_{\sigma \to 0} h_{ij} = \mathbb{1}(d(j) = i).$$

Thus the smoothed version of the NDCG metric (2) for query q can be defined as:

$$A_q(f, \sigma) := \sum_{i,j=1}^{m} G(l_i) D(j) h_{ij}, \tag{8}$$

Because $A_q(f, \sigma)$ is a linear combination of the h_{ij} functions, Proposition 1 and Remark 1 imply that it is continuous but non-differentiable at points where there is a tie in the ranking. However, it is still differentiable almost everywhere and in practice this non-differentiability did not cause any problem to the gradient descent optimizer.

It is noteworthy that the smoothing (8) not only applies to NDCG, but to any IR metric which can be written of the form

$$\sum_{i,j} a_i b_j \mathbb{1}(r(i) = j).$$

As discussed in (Le et al. (2009), Sect. 2.3), this encompasses other IR metrics such as precision at k or mean reciprocal rank. Therefore, the use of the smooth indicator variables h_{ij} is a *general* and powerful smoothing technique.

4.2 AP smoothing

For the AP metric (3), we can also make use of the smooth variables h_{ij} (7) and obtain the following smooth approximations:

$$\text{Prec@}k \approx \frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{m} l_i h_{ij}$$

$$\text{AP} \approx \frac{1}{\sum l_i}\sum_{j=1}^{m}\text{Prec@}j\sum_{i=1}^{m} l_i h_{ij}.$$

Combining these two approximations gives a smoothed approximation of AP.

But based on Eq. (4), a more direct approximation can be constructed. Using the smooth rank $\hat{r}_i$ defined in (5) along with a sigmoid smoothing for the indicator function $\mathbb{1}(r(j) \leq r(i))$, we can obtain the following smoothed version of AP:

$$A_q(f,\sigma) := \frac{1}{\sum l_i}\sum_{i,j} l_i l_j \frac{(1+\exp((f(\mathbf{x}_i)-f(\mathbf{x}_j))/\sigma))^{-1}}{\hat{r}_i}. \tag{9}$$

We have not tested empirically how this smooth AP performs. As noted in the introduction, the focus of this paper is the NDCG approximation. But it can be seen that the techniques developed here can be easily applied to other IR metrics.

4.3 Gradients

In order to perform gradient descent to optimize the smoothed IR metric, we need to compute the gradient of the smoothed indicator variables h_{ij} defined in Eq. (7). It turns out that,

$$\frac{\partial h_{ij}}{\partial f(\mathbf{x}_p)} = \frac{2}{\sigma(\sum_k e_{kj})^2}\left[e_{pj}(f(\mathbf{x}_p)-f(\mathbf{x}_{d(j)}))(e_{ij}-\mathbb{1}(i=p)\sum_k e_{kj}) + \mathbb{1}(d(j)=p)e_{ij}\left(f(\mathbf{x}_i)\sum_k e_{kj}-\sum_k e_{kj}f(\mathbf{x}_k)\right)\right], \tag{10}$$

with

$$e_{ij} := \exp\left(-\frac{(f(\mathbf{x}_i)-f(\mathbf{x}_{d(j)}))^2}{\sigma}\right). \tag{11}$$

One can check that, as expected, $\sum_p \frac{\partial h_{ij}}{\partial f(\mathbf{x}_p)} = 0$: it is indeed clear from the definition of h_{ij} that it is invariant by adding a constant to f, so the derivative along the direction (1,...,1) should be 0.

When the function f is linear, $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}$, the gradient of the smooth NDCG with respect to $\mathbf{w}$ is:

$$\frac{\partial A_q(\mathbf{w},\sigma)}{\partial \mathbf{w}} = \sum_{i,j,p} G(l_i)D(j)\frac{\partial h_{ij}}{\partial f(\mathbf{x}_p)}\mathbf{x}_p. \tag{12}$$

Note that when there are m documents for a query, the above sum can be rewritten such that its computation takes $O(m^2)$ operations instead of $O(m^3)$ as it first seems. In fact, for a given j, each term in (10) is of the form $a_i b_p$ and computing $\sum_{i,p} a_i b_p = (\sum_i a_i)(\sum_p b_p)$ just takes $O(m)$, rather than $O(m^2)$ operations. The final expression for the gradient is quite lengthy and is given in Appendix 2.

4.4 Optimization

For the optimization, we use nonlinear conjugate gradient (Shewchuk 1994) with the Polak-Ribiere update. This method only requires the objective function and its gradient, computed in Sect. 4.3.

The difficulty comes from the fact that the objective function is *non-convex* and the gradient descent algorithm tends to get trapped in a local optimum. We alleviate this difficulty by a combination of two key ingredients: first, a good starting point and regularizer; and second, an annealing procedure.

4.4.1 Starting point

The starting point is an important component while minimizing a non-convex objective function. We set it to be the solution found by regression on the gain function values:

$$\mathbf{w}_0 = \arg\min_{\mathbf{w}} \sum_q \sum_i (\mathbf{w}^\top \mathbf{x}_i^q - G(l_i^q))^2.$$

We chose this $\mathbf{w}_0$ because it is expected to be a reasonable solution. Alternatively, the solution of RankSVM (Joachims 2002) or any other ranking algorithms can also be used.

We also use this $\mathbf{w}_0$ as a *regularizer*: we add to the smooth objective function $\lambda||\mathbf{w} - \mathbf{w}_0||^2$, where λ is an hyperparameter to be tuned on a validation set. The reason for using $||\mathbf{w} - \mathbf{w}_0||^2$ as a regularizer is that since $\mathbf{w}_0$ is expected to be already a good solution, we do not want to deviate too much from it.

4.4.2 Annealing

As mentioned in the introduction, the choice of the smoothing parameter σ is critical: on the one hand, if it is too small the objective function approaches the original piecewise constant IR metric and is therefore highly non-smooth and difficult to optimize; on the other hand, if it is too large, the objective function is smooth and easy to optimize, but substantially different from what we are interested in. The SoftRank paper (Taylor et al. 2008) simply selects this parameter on a validation set. We propose here a different approach to solve this problem: annealing on σ. This procedure can be seen as a *homotopy* method (Dunlavy and O'Leary 2005) where a series of functions is constructed: the first one is easy to optimize, the last one is the function of interest and each function in the middle can be seen as a deformed function of the previous one; a homotopy method iteratively minimizes each of these functions starting from the minimizer of the previous function.

In our case, the deformation is controlled by σ: we start with a large σ, find the minimizer and iteratively reduce σ by steps. In our experiments, we start with $\sigma = 2^6$, divid σ by 2 at each step and stop when $\sigma = 2^{-6}$.

An illustration of the smoothness of the objective function as a function of σ can be found in Fig. 1. It is clear that when σ is small, the objective function is highly non-smooth and almost impossible to optimize. On the other hand, when σ is large, the objective function is smooth but rather different from the target one.

The evolution of the NDCG values for test and training as a function σ during the annealing procedure is shown in Fig. 2. The training NDCG improves while σ decreases (which is expected) but the test NDCG saturates (and even slightly deteriorates) for small σ. This might due to overfitting issues that will be analyzed in Sect. 5.2.

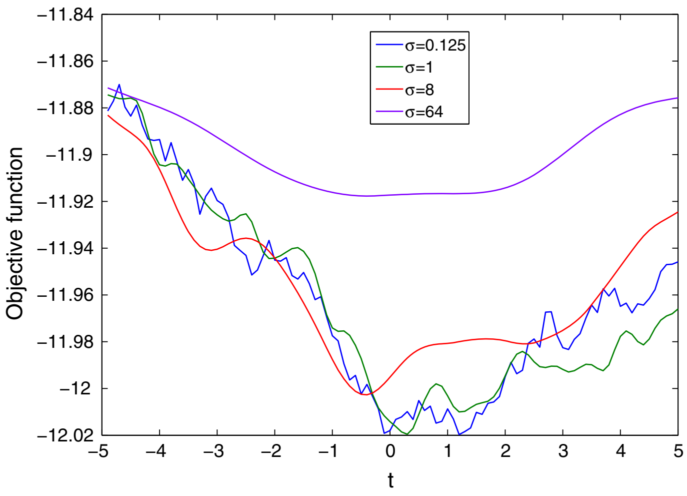

Fig. 1 Objective function $A_q(\mathbf{w}, \sigma)$ for different values of the smoothing parameter σ. The function is plotted at $\mathbf{w}^* + t\mathbf{w}_{rand}$, where $\mathbf{w}^*$ is the optimal solution and $\mathbf{w}_{rand}$ is a random Gaussian vector of unit variance on each component

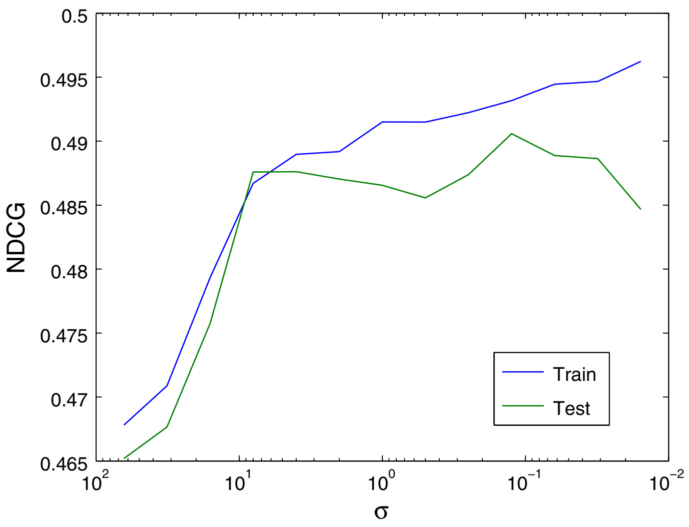

Fig. 2 Evolution of the average train and test NDCG on Ohsumed as a function of σ during the annealing procedure

4.4.3 Algorithm

Our algorithm in its generic version is called `SmoothRank` and is described in Algorithm 1. Two of its instantiations are `SmoothNDCG` and `SmoothAP` that minimize respectively the smooth NDCG (8) and the smooth AP (9).

Algorithm 1 `SmoothRank`: Minimization of a smooth IR metric by gradient descent and annealing

```
1:  Find an initial solution $\mathbf{w}_0$ (by regression for instance).
2:  Set $\mathbf{w} = \mathbf{w}_0$ and $\sigma$ to a large value.
3:  Starting from $\mathbf{w}$, minimize by (conjugate) gradient descent
        $\lambda||\mathbf{w} - \mathbf{w}_0||^2 - \sum_q A_q(\mathbf{w}, \sigma).$    (13)
4:  Divide $\sigma$ by 2 and go back to 3 (or stop if a stopping condition is satisfied).
```

5 Theoretical analysis

The effect of smoothing is theoretically analyzed in this section. Two antagonistic conclusions are drawn: the smaller the value of σ is, the smaller the approximation error is, but the higher the chances of overfitting are. While we have not made use of these theoretical results in practice in our algorithm, we believe that they can provide in future work valuable guidance on the final value of σ at which the annealing should be stopped.

5.1 Approximation error

As discussed in Sect. 4.1, we know that the smooth version of the NDCG (8) approaches the true NDCG as σ goes to 0 (assuming that there is no tie in the ranking). In this section, we further analyze the rate at which this approximation error goes to 0.

Let us first define μ as the minimum squared difference between two predictions:

$$\mu := \min_{i \neq j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2.$$

The denominator of (7) is always larger than 1 and so if $i \neq d(j)$,

$$h_{ij} \leq \exp\left(-\frac{(f(\mathbf{x}_i) - f(\mathbf{x}_{d(j)}))^2}{\sigma}\right) \leq \exp(-\mu/\sigma).$$

For $i = d(j)$, we use that $1 - \frac{1}{1+t} \leq t$ for $t \geq 0$ and obtain

$$1 - h_{ij} \leq \sum_{k \neq d(j)} \exp\left(-\frac{(f(\mathbf{x}_k) - f(\mathbf{x}_{d(j)}))^2}{\sigma}\right) \leq (m-1)\exp(-\mu/\sigma).$$

From the two inequalities above, we conclude that the difference between the smoothed NDCG and the exact NDCG as σ goes to 0 is:

$$O(\exp(-\mu/\sigma)).$$

5.2 Complexity

In this section, we study the generalization error as a function of the smoothing parameter σ. First, the following theorem is a generalization error bound for a Lipschitz loss function. In the one dimensional case this type of results is well known: see for instance (Bartlett and Mendelson 2003, Theorem 7) for classification. We extend it to the multidimensional case. For simplicity, we assume that all queries return m documents.

Theorem 1 *Let $\mathcal{X} = \mathbb{R}^{m \times d}$, the input space consisting of m documents represented by a d dimensional feature vector, and let $\mathcal{Y}$ be the output space consisting of permutations of length m. Let us assume that each feature vector has norm bounded by R. Let $\phi : \mathbb{R}^m \times \mathcal{Y} \rightarrow \mathbb{R}$ be a loss function such that $\forall y \in \mathcal{Y}, \phi(\cdot, y)$ is a Lipschitz function with constant L:*

$$\forall y \in \mathcal{Y}, \forall f, f' \in \mathbb{R}^m, |\phi(f, y) - \phi(f', y)| \leq L||f - f'||_2.$$

Then, for any training set of n queries $((x^1, y^1), \ldots, (x^n, y^n))$ drawn independently according to a probability distribution on $\mathcal{X} \times \mathcal{Y}$, with probability at least $1-\delta$, the following inequality holds for all $\mathbf{w} \in \mathbb{R}^d$ such that $||\mathbf{w}|| \leq B$,

$$\mathbf{E}\phi(\mathbf{w}^\top \mathbf{x}_1, \ldots, \mathbf{w}^\top \mathbf{x}_m, y) \leq \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{w}^\top \mathbf{x}_1^i, \ldots, \mathbf{w}^\top \mathbf{x}_m^i, y^i) + 3LBR\sqrt{\frac{m}{n}} + \sqrt{\frac{8 \log(2/\delta)}{n}}, \tag{14}$$

where the expectation on the left hand size is taken over the underlying data distribution on $\mathcal{X} \times Y$.

The proof is in Appendix 1.

Theorem 1 states that the difference between the generalization error and the training error can be controlled by the Lipschitz constant of the loss function. This result can be useful beyond ranking and can apply to any learning problem with a multivariate Lipschitz loss function.

The next step is to find the Lipschitz constant of the smooth NDCG (8). By applying the mean value theorem, one can see that this Lipschitz constant can be bounded by the maximun gradient norm:

$$L^2 \leq \max_f \sum_p \left(\sum_{i,j=1}^{m} G(l_i) D(j) \frac{\partial h_{ij}}{\partial f(\mathbf{x}_p)} \right)^2.$$

From the expression (10), we obtain the following bound using $|f(\mathbf{x}_i)| = |\mathbf{w}^\top \mathbf{x}_i| \leq BR$,

$$\frac{\partial h_{ij}}{\partial f(\mathbf{x}_p)} \leq \frac{8BR}{\sigma}.$$

And using Chebyshev's sum inequality,

$$\sum_{i,j=1}^{m} G(l_i) D(j) \leq m \max_r \sum_{i=1}^{m} G(l_i) D(r(i)) = m,$$

where the equality comes from the fact that by definition the maximum NDCG is 1.

Combining these equations, we finally obtain

$$L \leq \frac{8BR}{\sigma} m^{3/2}.$$

The complexity term in Eq. (14) (the middle term in the right hand side) is thus bounded by:

$$\frac{24(BRm)^2}{\sigma\sqrt{n}}.$$

It is likely that the above bound can be tightened, in particular the m^2 dependence. But the quantity of interest in this analysis is the $1/\sigma$ dependence. The conclusion to be drawn is that the smaller the σ, the larger the overfitting danger is.

This suggests that in our annealing algorithm we should do early stopping based on the validation set. We have tried such an approach but because of the small size of the validation set, the results were too unstable. One can not indeed select reliably *two* hyperparameters (the regularization parameter λ and the smoothing factor σ) on a validation set of around 20 queries (Ohsumed dataset) because of the large variance of the validation error.

5.3 Special case $\sigma \to +\infty$

When σ goes to infinity, it is clear that the h_{ij} in Eq. (7) converge to $1/m$ where m is the number of documents associated with the query. In that case, the smooth NDCG score (8) is constant. In other words, the objective function (13) becomes:

$$\lambda||\mathbf{w} - \mathbf{w}_0||^2 + \text{constant}.$$

Thus the minimizer is trivially found as $\mathbf{w}_0$. We said in Sect. 4.4.1 that we chose $\mathbf{w} = \mathbf{w}_0$ as a starting point, but it turns out that the starting point is irrelevant as long as the first value of σ in the annealing procedure is large enough. Indeed, after the first iteration, $\mathbf{w}$ will be very near from $\mathbf{w}_0$ regardless of the starting point.

5.4 Special case $\mathbf{w}_0 = 0$

When $\mathbf{w}_0 = 0$, λ and σ are redundant parameters and one of them can be eliminated. Indeed the following lemma holds:

Lemma 1 *Let* $\mathbf{w}^*_{\lambda,\sigma}$ *be the* $\mathbf{w}$ *found by minimizing* (13) *for a given* λ, σ, *and* $\mathbf{w}_0 = 0$. *Then,*

$$\forall c > 0, \quad \mathbf{w}^*_{\lambda,\sigma} = \sqrt{c}\,\mathbf{w}^*_{\lambda c,\sigma/c}$$

And since the ranking is independent of the scale of $\mathbf{w}$, this lemma shows that one can always fix either of the variables λ or σ.

Proof $\mathbf{w}^*_{\lambda c,\sigma/c}$ is the minimizer of

$$\lambda c||\mathbf{w}||^2 + \sum_q A_q(\mathbf{w}, \sigma/c). \tag{15}$$

With change of variable $\tilde{\mathbf{w}} = \sqrt{c}\mathbf{w}$, the above objective function becomes:

$$\lambda||\tilde{\mathbf{w}}||^2 + \sum_q A_q(\tilde{\mathbf{w}}/\sqrt{c}, \sigma/c).$$

But since $A_q(\tilde{\mathbf{w}}/\sqrt{c}, \sigma/c) = A_q(\tilde{\mathbf{w}}, \sigma)$ (that is clear from the definition of the h_{ij} variables (7)), the minimizer is obtained for $\tilde{\mathbf{w}} = \mathbf{w}^*_{\lambda,\sigma}$. Going back to the original variable, the minimizer of (15) is $\mathbf{w}^*_{\lambda,\sigma}/\sqrt{c}$. □

6 Non-linear extension

In this paper, we focus on the linear case, that is we learn a function f of the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. There are several ways of learning a non-linear extension:

Kernel trick Simply replace all the $\mathbf{x}_i$ by $\Phi(\mathbf{x}_i)$ where Φ is the so-called *empirical kernel PCA map*. See for instance (Chapelle et al. 2006, Sect. 2.3).
Non-linear architecture One can replace the dot products $\mathbf{w}^\top \mathbf{x}_i$ by $f_\mathbf{w}(\mathbf{x}_i)$ where $f_\mathbf{w}$ could for instance be a neural network with weights $\mathbf{w}$. For a gradient optimization, we simply need that $f_\mathbf{w}(\mathbf{x}_i)$ is differentiable with respect to $\mathbf{w}$. Ranking with neural networks has for instance been explored by Burges et al. (2005) and Taylor et al. (2008).
Functional gradient boosting When the class of functions cannot be optimized by gradient descent (that is the case of decision trees for instance), one can still apply the gradient boosting framework of Friedman (2001). Indeed in this framework one only needs to compute the gradient of the loss with respect to the function values. This is possible if we ignore the regularization term. The regularization would have to be imposed in the boosting algorithm, for instance by limiting the number of iterations and/or the number of leaves in a decision tree. More details about functional gradient boosting on general loss functions can be found in (Zheng et al. 2008).

7 Experiments

We tested our algorithm on the datasets of the Letor 3.0 benchmark[2] (Liu et al. 2007). The characteristics of the various datasets included in this distribution are listed in Table 1. We use the same five splits training/validation/test as the ones provided in the distribution.

As explained in the introduction, we are mostly interested in the Ohsumed dataset because it is the only one containing more than two levels of relevance. We believe that in this case the difference between various learning algorithms on the NDCG metric can be more pronounced.

The only parameter to select for our algorithm is the regularization parameter λ. We chose it on the validation set in the set $\{10^{-6}, 10^{-5},...,10^{2}, 10^{3}\}$. Even though we report values for the NDCG at various truncation levels, we always optimize for NDCG@50, both in our training criterion and during the model selection. This seems to give better results as we will see later.

We reimplemented two of the baselines given on the Letor webpage:

RankSVM Our own implementation of RankSVM achieves better results than the Letor baseline. After investigation, it turns out that this baseline was computed using SVMLight with a maximum number iterations set to 5,000 and the optimal solution was not found.
Regression We added a regularizer—i.e., we implemented ridge regression. We also overweighted the relevant documents such that the overall weight of relevant and non-relevant documents is the same.
Comparison to Letor baselines The NDCG achieved by various algorithms on the Ohsumed dataset is presented in Fig. 3. `SmoothNDCG` turns out to be the best performing method on this dataset. It is also noteworthy that RankSVM performs better than all the otherwise "listwise" approaches (SVMMAP, ListNet, AdaRank-NDCG,

[2] Available at http://www.research.microsoft.com/en-us/um/beijing/projects/letor/

Table 1 Statistics of the datasets included in the LETOR 3.0 benchmark

	Queries	Rel. levels	Features	Av. doc. per query	% Rel. doc.
Ohsumed	106	3	45	152	30
HP 2003	150	2	64	984	0.12
HP 2004	75	2	64	992	0.11
NP 2003	150	2	64	991	0.1
NP 2004	75	2	64	984	0.1
TD 2003	50	2	64	981	0.83
TD 2004	75	2	64	989	1.5

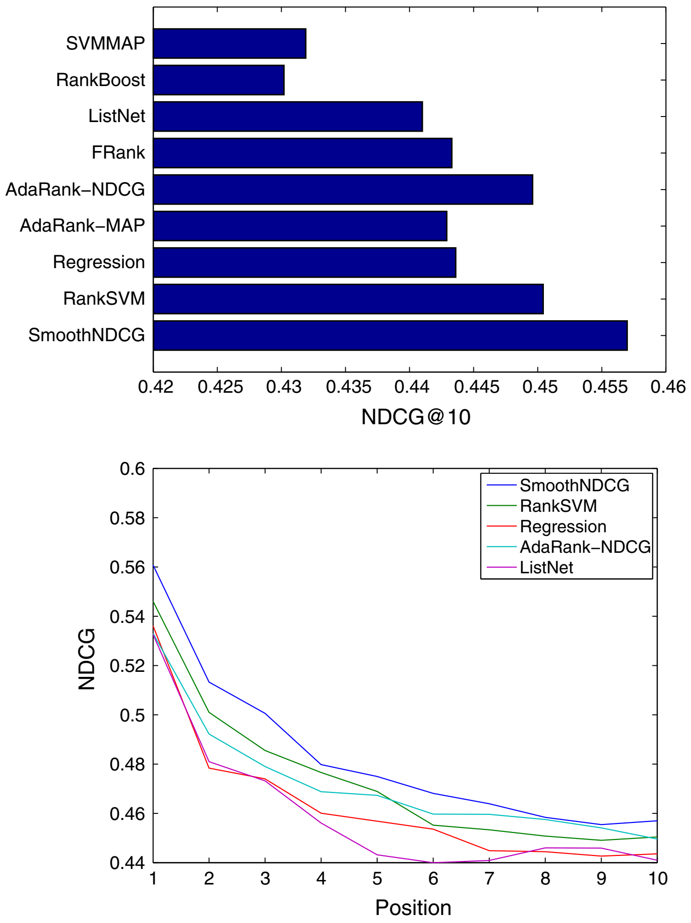

Fig. 3 Results on the Ohsumed dataset of SmoothNDCG compared to the baselines provided with the Letor distribution. Note that we reimplemented RankSVM and regression; see text for details. *Top* NDCG@10; *Bottom* NDCG at various truncation levels for a subset of the algorithms

AdaRank-MAP). There seems to be a common belief that listwise approaches are necessarily better than pairwise or pointwise approaches. But as one can see from Fig. 3, that is not the case, at least on this dataset.

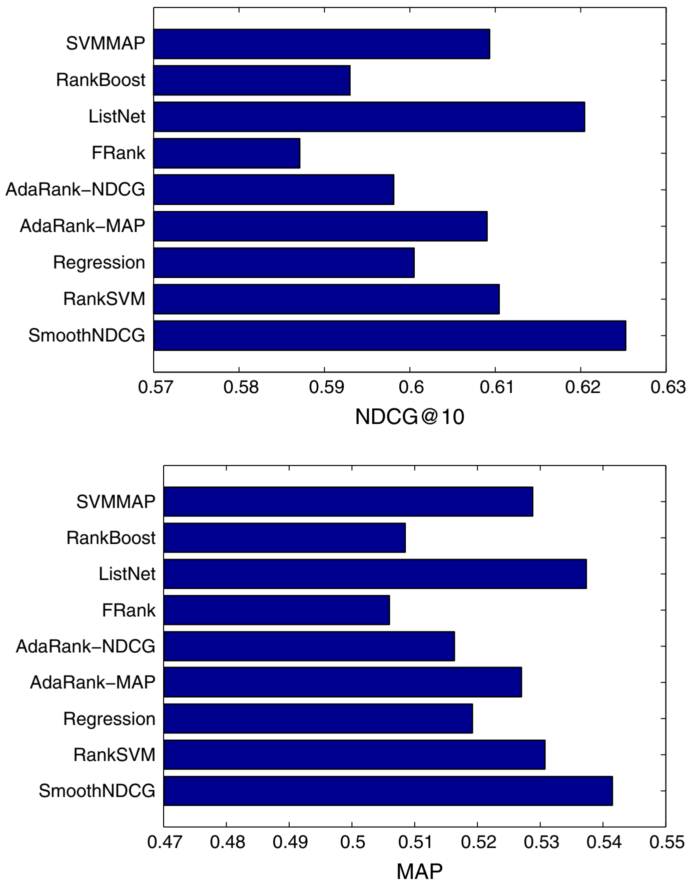

Fig. 4 NDCG@10 and MAP results averaged over all the datasets

Even though we are primarily interested in NDCG performances on the Ohsumed dataset, we also report in Fig. 4 the NDCG and MAP performances averaged over the seven datasets of the Letor distribution. Finally Table 2 reports the NDCG@10 and MAP performances of all the algorithms on all the Letor datasets.

Comparison to SoftRank We compared `SmoothNDCG` to SoftRank because both methods are closely related in the sense that they try to minimize a smoothed NDCG. Unfortunately the results for SoftRank are not available on the Letor baselines page. However results of the Gaussian Process (GP) extension of SoftRank (Guiver and Snelson 2008)—in which the smoothing factor is automatically found as the predictive variance of a GP—are reported in (Guiver and Snelson 2008; Fig. 3b) for the Ohsumed dataset from Letor 2.0. We have also run `SmoothNDCG` on that older version of the dataset and results are compared in Fig. 5. Accuracy of `SmoothNDCG` is a bit better, maybe due to the fact the annealing component enables the discovery of a better local minimum.

Influence of the truncation level in learning At a first sight, it seems that if one is interested in getting the best performances in terms of NDCG@k, the same measure should also be optimized during training. But as shown in Table 3, that is not the case.

Table 2 NDCG@10 and MAP results on all datasets

	HP2003	HP2004	NP2003	NP2004	TD2003	TD2004	Ohsumed
SVMMAP	0.742	0.718	0.687	0.662	0.244	0.205	0.445
RankBoost	0.733	0.625	**0.707**	0.564	0.227	**0.261**	0.441
ListNet	0.766	0.690	0.690	0.672	**0.275**	0.223	0.446
FRank	0.710	0.682	0.664	0.601	0.203	0.239	0.444
AdaRank-NDCG	0.748	0.691	0.668	0.627	0.237	0.194	**0.450**
AdaRank-MAP	**0.771**	0.722	0.678	0.622	0.228	0.219	0.449
Regression	0.749	0.630	0.682	**0.687**	0.243	0.199	0.444
RankSVM	0.764	0.671	0.688	0.675	0.265	0.206	0.445
SmoothDCG	0.764	**0.724**	0.676	0.676	0.270	0.233	0.448
	HP2003	HP2004	NP2003	NP2004	TD2003	TD2004	Ohsumed
SVMMAP	0.799	0.806	0.797	0.808	0.328	0.291	0.432
RankBoost	0.817	0.743	**0.807**	0.691	0.312	**0.350**	0.430
ListNet	0.837	0.784	0.802	**0.813**	0.348	0.318	0.441
FRank	0.797	0.761	0.776	0.730	0.269	0.333	0.443
AdaRank-NDCG	0.806	0.806	0.767	0.738	0.304	0.316	0.450
AdaRank-MAP	**0.838**	**0.833**	0.764	0.750	0.307	0.329	0.443
Regression	0.822	0.719	0.802	0.804	0.330	0.283	0.444
RankSVM	0.818	0.772	0.789	0.795	**0.357**	0.291	0.450
SmoothDCG	0.833	0.824	0.784	0.807	0.336	0.335	**0.457**

In each column, the result of the best performing algorithm is bolded; but because of the small size of the datasets, none of the differences is statistically significant

Indeed the larger k is during the optimization, the better the results tend to be, independent of the truncation level used as test measure. That is the reason why in the experimental results reported above, we used $k = 50$ in the training (and model selection) criterion.

This observation was also made in (Taylor et al. 2008): not only the authors of that paper did not use any truncation during learning, but they have also observed that using a discount function which does not decay as quickly as 1/log gave better results. And we concord with their explanation: in the extreme case, optimizing NDCG@1 amounts to ignoring training documents which are not in the first position; throwing this data away is harmful from a generalization point of view and is likely to lead to overfitting.

Experimental evidence of the inadequacy of optimizing certain test metrics has also been reported by Chakrabarti et al.(2008),Donmez et al.(2009), and Yilmaz and Robertson (2009).

8 Conclusion

In this paper, we have shown how to optimize complex information retrieval metrics such as NDCG by performing gradient descent optimization on a smooth approximation of these metrics. We have mostly focused on NDCG because it is the only common metric that is

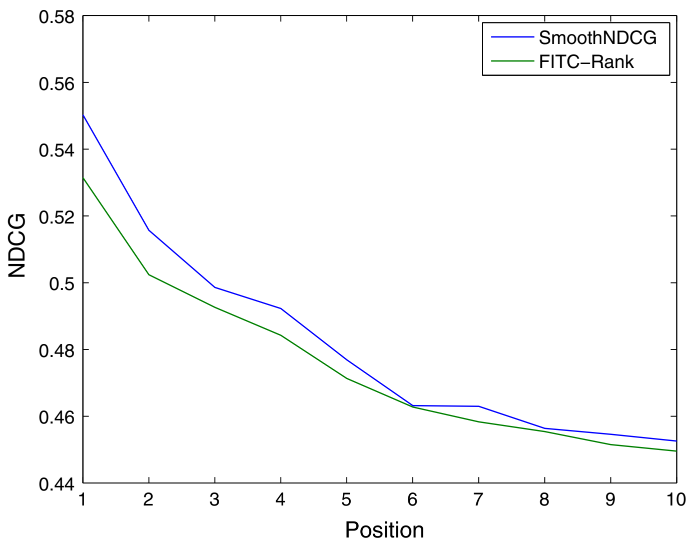

Fig. 5 Comparison of `SmoothNDCG` and FTIC-Rank, the sparse GP extension of SoftRank (Guiver and Snelson 2008), on the older version of the Ohsumed dataset from Letor 2.0

Table 3 Effect of k when optimizing NDCG@k

	NDCG@1	NDCG@5	NDCG@10
$k = 1$	0.5365	0.4622	0.4369
$k = 5$	0.5397	0.4691	0.4470
$k = 10$	**0.5834**	0.4723	0.4496
$k = 50$	0.5608	**0.4750**	**0.4570**

Various rows (resp. columns) correspond to different truncation levels used during training (resp. testing). It turns out that the larger k, the better

able to handle multi-level relevance—a case where ranking is more difficult. But as discussed in Sect. 4.1, this framework can be easily extended to any other metrics because we have been able to smooth the indicator function: "Is document i at the jth position in the ranking?"

A critical component of this framework is the choice of the smoothing factor. Related to our work is SoftRank (Taylor et al. 2008). In the original paper, the smoothing parameter is found on the validation set while in (Guiver and Snelson 2008) it is found automatically as the predictive variance of a sparse Gaussian Process (and the smoothing is thus different for each document). But these papers do not address the problem of local minima. The annealing technique we have presented here can alleviate this difficulty and better local minimum can be found as suggested by the superior results obtained by our algorithm compared to SoftRank.

Finally, we have presented a theoretical analysis and in particular we have studied the influence of the smoothness of the loss function on the generalization error. Even though we have not directly used these results in our algorithm, we plan, in future work, to leverage them as a tool for deciding when to stop the annealing.

Appendix 1: Proof of Theorem 1

The proof of the theorem makes uses of the results and techniques presented in (Bartlett and Mendelson 2003).

Key to the proof is to upper bound the Gaussian complexity of the class of functions

$$F := \{(x,y) \in (\mathcal{X} \times \mathcal{Y}) \mapsto \phi(\mathbf{w}^\top \mathbf{x}_1, \ldots, \mathbf{w}^\top \mathbf{x}_m, y), \text{with } \mathbf{w} \text{ such that } ||\mathbf{w}|| \leq B\}$$

Given $(x^1, y^1) \ldots, (x^n, y^n)$, the empirical Gaussian complexity of F is defined as

$$\hat{G}_n(F) := \mathbf{E} \sup_{f \in \mathcal{F}} \frac{2}{n} \left| \sum_{i=1}^n g_i f(x^i, y^i) \right|,$$

where g_i are independent Gaussian random variables with zero mean and unit variance.

For a given $\mathbf{w}$, define the random variables

$$X_\mathbf{w} = \sum_{i=1}^n \phi(\mathbf{w}^\top \mathbf{x}_1^i, \ldots, \mathbf{w}^\top \mathbf{x}_m^i, y^i) g_i$$

and

$$Y_\mathbf{w} = L \sum_{i=1}^n \sum_{j=1}^m \mathbf{w}^\top \mathbf{x}_j^i h_{ij},$$

where g_i and h_{ij} are Gaussian random variables.

Then, $\forall \mathbf{w}, \mathbf{w}'$

$$\begin{aligned}
\mathbf{E}(X_\mathbf{w} - X'_\mathbf{w})^2 &= \sum_{i=1}^n (\phi(\mathbf{w}^\top \mathbf{x}_1^i, \ldots, \mathbf{w}^\top \mathbf{x}_m^i, y^i) - \phi(\mathbf{w}'^\top \mathbf{x}_1^i, \ldots, \mathbf{w}'^\top \mathbf{x}_m^i, y^i))^2 \\
&\leq L^2 \sum_{i=1}^n \sum_{j=1}^m (\mathbf{w}^\top \mathbf{x}_j^i - \mathbf{w}'^\top \mathbf{x}_j^i)^2 \\
&= \mathbf{E}(Y_\mathbf{w} - Y'_\mathbf{w})^2
\end{aligned}$$

By Slepian's Lemma (Ledoux and Talagrand (1991), Corollary 3.14), we have,

$$\mathbf{E} \sup_\mathbf{w} X_\mathbf{w} \leq 2\mathbf{E} \sup_\mathbf{w} Y_\mathbf{w}.$$

We also have $\mathbf{E} \sup_\mathbf{w} X_\mathbf{w} = n\hat{G}_n(F)$ and

$$\begin{aligned}
\mathbf{E} \sup_\mathbf{w} Y_\mathbf{w} &= L\mathbf{E} \sup_\mathbf{w} \mathbf{w}^\top \left(\sum_{i,j} \mathbf{x}_j^i h_{ij} \right) \\
&\leq LB\mathbf{E}|| \sum_{i,j} \mathbf{x}_j^i h_{ij} || \\
&\leq LB \left(\mathbf{E}|| \sum_{i,j} \mathbf{x}_j^i h_{ij} ||^2 \right)^{1/2} \\
&= LB \left(\sum_{i,j} ||\mathbf{x}_i^j||^2 \right)^{1/2} \\
&\leq LBR\sqrt{nm}
\end{aligned}$$

The Gaussian complexity $G_n(F)$ is defined as the expectation over the choice of the sample $(x^1, y^1) \ldots, (x^n, y^n)$

$$\begin{aligned} G_n(F) &= \mathbf{E}\hat{G}_n(F) \\ &\leq 2LBR\sqrt{\frac{m}{n}} \end{aligned}$$

We now apply theorem Theorem 8 of (Bartlett and Mendelson 2003):

$$\begin{aligned} \mathbf{E}\phi(\mathbf{w}^\top \mathbf{x}_1, \ldots, \mathbf{w}^\top \mathbf{x}_m, y) \leq \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{w}^\top \mathbf{x}_1^i, \ldots, \mathbf{w}^\top \mathbf{x}_m^i, y^i) \\ + R_n(F) + \sqrt{\frac{8\log(2/\delta)}{n}}, \end{aligned}$$

where $R_n(f)$ is the *Rademacher* complexity of F.

Combining the above two inequalities with the fact that the Rademacher complexity can be bounded by the Gaussian complexity (Ledoux and Talagrand (1991), Eq. 4.8) $R_n(f) \leq \sqrt{\pi/2}G_n(f)$, proves the theorem.

Appendix 2: Gradient of the smooth NDCG

Combining Eqs. (10) and (12), the final expression of the gradient with respect of $\mathbf{w}$ of the smooth NDCG $A_q(\mathbf{w}, \sigma)$ is:

$$\begin{aligned} \frac{\partial A_q(\mathbf{w}, \sigma)}{\partial \mathbf{w}} = \frac{2}{\sigma}\sum_j \frac{D(j)}{E_j^2} \Bigg[\left(\sum_i G(l_i)e_i\right)\left(\sum_p e_{pj}(f(\mathbf{x}_p) - f(\mathbf{x}_{d(j)}))\mathbf{x}_p\right) \\ - E_j \sum_p e_{pj}(f(\mathbf{x}_p) - f(\mathbf{x}_{d(j)}))G(l_p)\mathbf{x}_p \\ + E_j \sum_i G(l_i)e_{ij}f(\mathbf{x}_i)\mathbf{x}_{d(j)} - \left(\sum_i G(l_i)e_{ij}\right)\left(\sum_i f(\mathbf{x}_i)e_{ij}\right)\mathbf{x}_{d(j)} \Bigg], \end{aligned}$$

where e_{ij} is defined in (11) and $E_j := \sum_k e_{kj}$.

References

Bartlett, P., & Mendelson, S. (2003). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research, 3*, 463–482.

Bridle, J. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. F. Soulie & J. Herault (Eds.), *Neurocomputing: Algorithms architectures and applications* (pp. 227–236). Berlin: Springer.

Burges, C. J., Le, Q. V., & Ragno, R. (2007). Learning to rank with nonsmooth cost functions. In Schölkopf, J. Platt, & T. Hofmann (Eds.), *Advances in neural information processing systems* (Vol. 19).

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. In *Proceedings of the international conference on machine learning*.

Cao, Y., Xu, J., Liu, T. Y., Li, H,. Huang, Y., & Hon, H. W. (2006). Adapting ranking SVM to document retrieval. In *SIGIR*.

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007) Learning to rank: From pairwise approach to listwise approach. In *International conference on machine learning*.

Chakrabarti, S., Khanna, R., Sawant, U., & Bhattacharyya, C. (2008). Structured learning for non-smooth ranking losses. In *International conference on knowledge discovery and data mining (KDD)*.

Chapelle, O., Chi, M., & Zien, A. (2006) A continuation method for semi-supervised SVMs. In *International conference on machine learning*.

Chapelle, O., Le, Q., & Smola, A. (2007) Large margin optimization of ranking measures. In *NIPS workshop on machine learning for web search*.

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. In *19th Annual conference on learning theory*, Lecture Notes in Computer Science (Vol. 4005, pp. 605–619). Springer.

Donmez, P., Svore, K., & Burges, C. (2009). On the local optimality of LambdaRank. In *SIGIR'09: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, ACM* (pp. 460–467).

Dunlavy, D., & O'Leary, D. (2005). *Homotopy optimization methods for global optimization*. Tech. Rep. SAND2005-7495, Sandia National Laboratories.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003) An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research, 4*, 933–969.

Friedman, J. (2001) Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*, 1189–1232.

Guiver, J., & Snelson, E. (2008). Learning to rank with softrank and gaussian processes. In: *SIGIR'08: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, ACM* (pp. 259–266).

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In A. Smola, P. Bartlett, B. Schölkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers*. Cambridge, MA: MIT Press.

Jarvelin, K., & Kekalainen, J. (2002) Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems, 20*(4), 422–446.

Joachims, T. (2002) Optimizing search engines using clickthrough data. In *Proceedings of the ACM conference on knowledge discovery and data mining (KDD), ACM*.

Le, Q. V., Smola, A., Chapelle, O., & Teo, C. H. (2009). Optimization of ranking measures (submitted).

Ledoux, M., & Talagrand, M. (1991) *Probability in banach spaces: Isoperimetry and processes*. Berlin: Springer.

Li, P., Burges, C., & Wu, Q. (2008) Mcrank Learning to rank using multiple classification and gradient boosting. In J. Platt, D. Koller, Y. Singe, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20, pp. 897–904). Cambridge, MA: MIT Press.

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

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval*.

Shewchuk, J. R. (1994). *An introduction to the conjugate gradient method without the agonizing pain*. Tech. Rep. CMU-CS-94-125, School of Computer Science, Carnegie Mellon University.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: Optimizing non-smooth rank metrics. In *WSDM'08: Proceedings of the international conference on web search and web data mining, ACM* (pp. 77–86).

Tsai, M. F., Liu, T. Y., Qin, T., Chen, H. H., & Ma, W. Y. (2007). Frank: A ranking method with fidelity loss. In *International ACM SIGIR conference on research and development in information retrieval*.

Wu, M., Chang, Y., Zheng, Z., & Zha, H. (2009). Smoothing DCG for learning to rank: A novel approach using smoothed hinge functions. In *Proceedings of the 17th ACM conference on information and knowledge management, CIKM*.

Xia, F., Liu, T., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th international conference on machine learning* (pp. 1192–1199).

Xu, J., & Li, H. (2007). Adarank: A boosting algorithm for information retrieval. In *International ACM SIGIR conference on research and development in information retrieval*.

Yilmaz, E., & Robertson, S. (2009). On the choice of effectiveness measures for learning to rank. Information Retrieval Journal To appear in the special issue of Learning to Rank for Information Retrieval. doi:10.1007/s10791-009-9116-x.

Yue, Y., & Burges, C. (2007). *On using simultaneous perturbation stochastic approximation for learning to rank, and the empirical optimality of LambdaRank*. Tech. Rep. MSR-TR-2007-115, Microsoft Reseach.

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007) A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM* (pp. 271–278).

Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., & Sun, G. (2008). A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems* (Vol. 20, pp 1697–1704). MIT Press.