Optimization of Ranking Measures

Quoc V. Le   QUOCLE@STANFORD.EDU
Department of Computer Science, Stanford University, USA

Alex Smola   ALEX@SMOLA.ORG
Olivier Chapelle   CHAP@YAHOO-INC.COM
Yahoo! Research, Mission College, Santa Clara, USA

Choon Hui Teo   CHOONHUI.TEO@ANU.EDU.AU
Statistical Machine Learning Program
NICTA and ANU, Canberra, 2600 ACT, Australia

Editor: Ralf Herbrich

Abstract

Web page ranking requires the optimization of sophisticated performance measures. Current approaches only minimize measures indirectly related to performance scores. We present a new approach which allows optimization of an upper bound of the appropriate loss function.

This is achieved via structured estimation, where in our case the input corresponds to a set of documents and the output is a ranking. Training is efficient since computing the loss function can be done via a linear assignment problem. At test time, a sorting operation suffices, as our algorithm assigns a relevance score to every (document, query) pair. Moreover, we provide a general method for finding tighter nonconvex relaxations of structured loss functions. Experiments show that the our algorithm yields improved accuracies on several public and commercial ranking datasets.

1. Introduction

Web page ranking has traditionally been based on a hand designed ranking function such as BM25 (Robertson et al., 1994). However ranking is now considered a supervised learning problem and several machine learning algorithms have been applied to it (Burges et al., 2005; Cao et al., 2006).

Traditional approaches in learning to rank optimize uniform ranking measures such as number of mis-ordered pairs (Herbrich et al., 2000). However, it is often the case that the users may be more interested in the most relevant items (first page) and ignore other items. Thus it is more appropriate for a ranker to spend effort and get the topmost items right. Performance measures have been developed in the information retrieval community which pay more attention to the top of the ranking. Examples of such measures are the *Normalized Discounted Cumulative Gain (NDCG)*, *Mean Reciprocal Rank (MRR)*, or *Precision@k*. They are used to address the issue of evaluating rankers, search engines or recommender sytems (Voorhees, 2001; Jarvelin and Kekalainen, 2002; Breese et al., 1998; Basilico and Hofmann, 2004).

Ranking methods have come a long way in the past years. Beginning with vector space models (Salton, 1971; Salton and McGill, 1983), various feature based methods have been proposed (Lee et al., 1997). Popular set of features include BM25 (Robertson et al., 1994) or its variants (Robertson and Hull, 2000; Craswell et al., 2005). Following the intent of Richardson et al. (2006) we show that when combining such methods with machine learning, the performance of the ranker can be increased significantly.

Over the past decade, many machine learning methods have been proposed. Ordinal regression (Herbrich et al., 2000; Joachims, 2002) using a SVM-like large margin method is one of the popular approach. Perceptrons and online methods have been introduced in (Crammer and Singer, 2002, 2005; Basilico and Hofmann, 2004) In the context of web search ranking, Neural Networks have been proposed by Burges et al. (2005). These methods aim at finding the best ordering function over the returned documents. However, as first noted in ?, none of them address the fact that ranking at the top is more important.

Only recently two theoretical papers (Rudin, 2006; Cossock and Zhang, 2006) discuss the issue of learning ranking with preference to top scoring documents. However, the cost function of (Rudin, 2006) is only vaguely related to the cost function used in evaluating the performance of the ranker. (Cossock and Zhang, 2006) argue that, in the limit of large samples, regression on the labels may be sufficient and, indeed, their method becomes competitive in the case of large numbers of observations.

Our work uses the framework of Support Vector Machines for Structured Outputs (Tsochantaridis et al., 2005; Joachims, 2005; Taskar et al., 2004) to deal with the inherent non-convexity of the performance measures in ranking. More precisely, we view ranking as trying to learn an ordering of the data: the input is a set of documents (associated with a given query) and the output is a ranking – or a permutation – of the documents. The optimization problem we propose is very general: it covers a broad range of existing criteria in a plug-and-play manner. It extends to position-dependent ranking and diversity-based scores. In addition to a ranking specific loss, we also show how structured loss bounds can be tightened in a general way.

Of particular relevance are three recent papers by Joachims (2005); Yue et al. (2007); ? which address the complication of the information retrieval loss functions. More specifically, they show that several ranking-related scores such as Precision@n, the Area under the ROC curve and the Mean Average Precision (MAP) can be optimized by using a variant of Support Vector Machines for Structured Outputs (SVMStruct). We use a similar strategy in our algorithm to obtain an Optimization of Ranking Measures (ORM) using the inequalities proposed by Tsochantaridis et al. (2005).

Finally, two other papers (Burges et al., 2007; Taylor et al., 2008) have attempted to optimize directly a smoothed version of nonconvex cost functions by *gradient descent.* [1]

2. Ranking

In the following we study the problem of ranking collections of objects d, such as webpages, products, movies, such that the most relevant (or popular) objects are retrieved at the beginning of the list. Typically we will do so given a query q. For instance, we might want to sort a collection of scientific papers such that those related to the query 'machine

---

1. The current paper is an extended version of ? which is available on arXiv.

Table 1: Summary of notation

Variable	Meaning
d_i	document
q^i	i-th query
l_i	number of documents for q^i
$D^i = \left\{ d_1^i, \ldots, d_{l_i}^i \right\}$	documents for q^i
(q, d)	document-query pair
m	number of queries for training
$y_j^i \in [0, \ldots, y_{\max}]$	relevance of document d_j^i
$g(q, d_i)$	scoring function
$\pi(D, q, g)$	permutation π obtained by sorting $g(d_i, q)$
$\Delta(y, \pi)$	loss incurred by permutation π for labels y

learning' are retrieved first.[2] The basic assumption we make is that such a ranking can be obtained by designing a scoring function $g(d, q)$ which tells us how relevant document d is for query q.

The requirement of being able to deal with each (document, query) pair independently arises from reasons of practicality. To search through a large collection of documents efficiently it is preferable to be able to score each document individually, in particular when performing such operations in a distributed setting.

Users are often only interested in the most relevant documents rather than the entire ranked list. For instance, for web search, it is likely that users will only want to look at the first 10 retrieved results. Similarly, when retrieving documents, a user may only be willing to consider viewing the best k documents. Alternatively, his satisfaction with the system may depend on how many documents he needs to sift through until he finds a relevant one. What is common to all those measures is that the *order* of the retrieved objects matters and that in many cases we will care primarily about the retrieval of the most relevant documents rather than all of them.

Our approach to ranking is centered around *learning* the scoring function $g(d, q)$ with desirable properties. This is in contrast to many past strategies in information retrieval which rely on ingenious engineering to obtain good scoring functions. Obviously, engineering still has its place in this framework — after all, we need to be able to obtain good features of a (document, query) pair which can be used to define such a scoring function. Hence, in this paper we take advantage of statistics and machine learning to guide us in finding a function that is optimized for this purpose. The design of good *features* remains the prerogative of an information retrieval expert.

2.1 The Problem

Let us formally define the problem in the language of machine learning. Assume that for a query q we are given a collection of documents $D := \{d_1, \ldots, d_l\}$ with associated labels

---

2. A corresponding problem arises in computational advertising where we aim to find a good ordering of ads for a given webpage such as to maximize revenue and click probability.

$y = \{y_1, \ldots, y_l\}$ where $y_i \in \{0, \ldots, y_{\max}\}$, which induce an order among the documents d_i. Here $y_{\max} \in \mathbb{N}$ is the maximum value of a label assigned to an object.

For instance, the label 0 may correspond to 'irrelevant' and $y_{\max}$ may correspond to 'highly relevant'. Often those labels are generated by (paid) experts. In other cases, they are obtained from clickthrough data. Often $y_{\max}$ is fairly small. For instance, the Netflix competition offers five different levels of appreciation of a movie. In fact, in some cases we only have two labels: $\{0, 1\}$, i.e. irrelevant and relevant. Using the label information, $y_i > y_j$ implies that document d_i is preferable to document d_j. It is our goal to find some permutation $\pi \in \Pi_l$ (here Π_l is the permutation group on l elements) of the documents such that the documents with a high label value y_i will show up at the beginning of the sorted list. This permutation is obtained by sorting the documents according to a score $g(d_i, q)$ which is obtained from the scoring function g in *descending* order. We will refer to this permutation as $\pi(D, q, g)$.

Definition 1 (Permutations) *In this paper, a permutation π maps from the set of documents to ranks. In other words, $\pi(i) = j$ means that the i-th documents has rank j. Whenever convenient we will also associate with π a* matrix *whose entries $\pi_{ij} \in \{0; 1\}$ are nonzero if and only if $j = \pi(i)$. Finally, for some vector a, $a[\pi]$ denotes the permuted vector πa, that is $a[\pi]_i = a_{\pi(i)}$*

The performance of the system is evaluated by comparing π to the labels y. Obviously an ideal case would be if for all i, j we have that $y_i > y_j \iff g(d_i, q) > g(d_j, q)$. In other words, the score $g(d, q)$ mimics the order imparted by the labels y. We denote the loss function measuring the discrepancy between y and π by $\Delta(y, \pi)$.

We are given a set of m instances of queries q^i and m document collections $D^i = \{d_1^i, \ldots, d_{l_i}^i\}$ of cardinality l_i respectively and corresponding labels $y^i = \{r_1^i, \ldots, r_{l_i}^i\}$. The *super*scripts denote sample indices whereas the *sub*scripts index elements within a sample. One strategy would be to minimize the empirical risk

$$R_{\text{emp}}[g] := \frac{1}{m} \sum_{i=1}^{m} \Delta(y^i, \pi(D^i, q^i, g)) \tag{1}$$

which is the sum over all empirical losses incurred on m instances.

2.2 Multivariate Ranking Losses

We now discuss permutation based losses $\Delta(y, \pi)$ in more detail. We consider losses which can be computed by comparing y, the permutation sorting y into descending order, and π, the proposed permutation of documents. It turns out that a large number of losses commonly used in the information retrieval community fit this template. Note that such scores are irreducibly multivariate, that is, they depend on the entire set of labels y and the corresponding permutation. Losses arising from 'Winner Takes All' (WTA), Mean Reciprocal Rank (MRR) (Voorhees, 2001), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen, 2002) all fulfill this property.

Definition 2 (Multivariate Ranking Loss) *Let $a \in \mathbb{R}^l$ and $y \in \{0, \ldots, y_{\max}\}^l$ and $\pi \in \Pi_l$. Assume that $b : \{0, \ldots, y_{\max}\}^l \to \mathbb{R}^l$ is some label weighting function. Then a, b induce a ranking loss via*

$$\Delta(y, \pi) := \max_{\pi' \in \Pi_l} \left[a[\pi'] - a[\pi] \right]^\top b(y). \tag{2}$$

Clearly, $\Delta(y, \pi) \geq 0$ for all y, π, since we have a maximization over π' in (2). In other words, (2) has the form of a regret, namely the difference between the scores for the best performing permutation π' and the proposed permutation π. Note that in general π' is not unique because there are often several relevant documents to a query. This loss formulation is useful, since many performance measures in Information Retrieval are expressed in terms of a retrieval score rather than a loss. This can be converted into a loss simply by considering the regret between the best possible answer and the answer proposed by suggesting the permutation π. Note that maximization with respect to π' is easily computed by virtue of the rearrangement inequality (Wayne, 1946):

Theorem 3 (Rearrangement Inequality) *Let $a, b \in \mathbb{R}^n$ and let $\pi \in \Pi$. Moreover, assume that a is sorted in decreasing order. Then $\langle a, b[\pi] \rangle$ is maximal for the permutation sorting b in decreasing order.*

We now show how most commonly used ranking losses can be cast into this form.

2.3 Examples

Winner-Takes-All (WTA): Assume that we are only interested in the first retrieved document. That is, we are only interested in $\pi^{-1}(1)$. If the document $d_{\pi^{-1}(1)}$ is relevant, we obtain a score of 1, otherwise we do not benefit from it, i.e. we have a score of 0.

To rewrite this in terms of (2) we set $y_i = 1$ for relevant documents and $y_i = 0$ otherwise. Moreover, we define a to be the first canonical vector, i.e. $a_i = \delta_{i,1}$. Finally, we let b be the identity map. In this case $a[\pi]^\top b(y) = y_{\pi^{-1}(1)}$, as required. We have

$$\text{WTA}(\pi, y) = a[\pi]^\top b(y) \text{ where } a_i = \delta_{i,1} \text{ and } b(y)_i = y_i \tag{3}$$

Mean Reciprocal Rank (MRR): Assuming that there only exists *one* relevant document we may want to measure the quality of π by how long it takes us to find this particular, relevant document. Assume that this document occurs at position n. In this case we would have had to look at n documents total until we found the relevant one, yielding an efficiency of $\frac{1}{n}$. If we need to perform many of such retrieval operations the average efficiency will be additive in $\frac{1}{n}$, hence the mean reciprocal rank is a good measure for single-item retrieval. It can be expressed in terms of an inner product by letting $y_i = 1$ for the relevant document and $y_i = 0$ otherwise (b is again the identity map). Moreover, we let $a_i = \frac{1}{i}$ to mimic the ratio decay. By construction we have that now $a[\pi]^\top b(y) = \frac{1}{n}$ as desired.

$$\text{MRR}(\pi, y) = a[\pi]^\top b(y) \text{ where } a_i = \frac{1}{i} \text{ and } b(y)_i = y_i. \tag{4}$$

In other words, the reciprocal rank is the inverse of the rank assigned to the most relevant document. Again, this can be converted into a loss by means of (2).

Discounted Cumulative Gain (DCG): WTA and MRR only care about a single term in π, namely the permutation mapping to the first and the relevant document respectively. This may prove to be a bit too aggressive. For instance, in web page ranking there may be quite a number of different yet all relevant webpages. For instance, when querying information about the weather in San Francisco, there may be a number of sites providing this information, all equally informative for a user. Hence all of them would share the same rating. It would be desirable if they were all ranked fairly highly according to π. The Discounted Cumulative Gains score (Jarvelin and Kekalainen, 2002) is one such measure which takes this explicitly into account.

In its basic form it has a logarithmic position dependency, that is, 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^{y_i} to highly rated documents. Thus the score associated with y and π is defined as $\sum_i (2^{y_i} - 1)/(\log_2 \pi(i) + 1)$. This can be rewritten in inner product form using the definitions

$$\mathrm{DCG}(\pi, y) = a[\pi]^\top b(y) \text{ where } a_i = \frac{1}{\log_2(i+1)} \text{ and } b(y)_i = 2^{y_i} - 1. \tag{5}$$

Clearly, if a relevant document is retrieved with a high rank the contribution to DCG is significant.

DCG Variants (DCG@k) Another variant of DCG truncates the sum over the positions after k documents. This is commonly abbreviated as DCG@k. In this case, instead of (5) one uses $a_i = \frac{1}{\log_2(i+1)}$ for all $i \leq k$ and $a_i = 0$ otherwise. In search engines the *truncation level* k is typically 10, as this constitutes the number of webpages returned in a query.

Normalized Discounted Cumulative Gain (NDCG): A downside of DCG is that its numerical range depends on y. For instance, a query with many associated relevant documents will yield a considerably larger value of DCG at optimality than one with only few relevant documents.

In order to normalize the effect of a single pair (D, q) one often resorts to normalizing the DCG or the DCG@k score by the relative maximum. This leads to the Normalized Discounted Cumulative Gains (NDCG) score and its truncated counterpart, defined as follows:

$$\mathrm{NDCG}(\pi, y) := \frac{\mathrm{DCG}(\pi, y)}{\max_{\pi'} \mathrm{DCG}(\pi', y)} \tag{6}$$

$$\text{and } \mathrm{NDCG}@k(\pi, y) := \frac{\mathrm{DCG}@k(\pi, y)}{\max_{\pi'} \mathrm{DCG}@k(\pi', y)}. \tag{7}$$

All we need to do is modify the definition of $b(y)$ in (5). We obtain

$$\mathrm{NDCG}(\pi, y) = a[\pi]^\top b(y) \text{ where } b(y)_i = \frac{2^{y_i} - 1}{\max_{\pi'} \mathrm{DCG}(\pi', y)} \tag{8}$$

$$\mathrm{NDCG@k}(\pi, y) = a[\pi]^\top b(y) \text{ where } b(y)_i = \frac{2^{y_i} - 1}{\max_{\pi'} \mathrm{DCG@k}(\pi', y)}. \tag{9}$$

It is the Normalized Discounted Gains Score at k which this paper focuses on, since it provides a good trade off between range-dependency and it adds normalization to ensure each query is given equal weight.

Precision@k: This measure allows one to deal with a precision/recall trade off for binary classification. It simply measures the number of elements with the desired class label among k retrieved terms. Joachims (2005) showed that this can be learned in linear models by using a suitable convex upper bound. It turns out that this measure, too, can be expressed in inner product form.

Assume we have two classes $\{0, 1\}$ and that we want to retrieve objects of class 1. In this case, we set y_i to be the class label of object d_i. To obtain the dot product form we define

$$\mathrm{Prec@k}(\pi, y) = a[\pi]^\top b(y) \text{ where } a_i = \begin{cases} \frac{1}{k} & \text{if } i \leq k \\ 0 & \text{else} \end{cases} \text{ and } b(y)_i = y_i \tag{10}$$

The main difference to NDCG@k is that Prec@k has no decay factor, weighing the top k answers equally.

Expected rank utility (ERU): In recommender systems, e.g. for movie ratings, it is often desirable to retrieve only terms which have a certain minimum degree of popularity, say d. To retrieve movies below a minimum rating offers no value to the user. Moreover, the position dependent decay is considered exponential. This choice of a loss function is manifest in the Expected Rank Utility, which can be expressed in dot product form, too:

$$\mathrm{ERU}(\pi, y) = a[\pi]^\top b(y) \text{ where } a_i = 2^{\frac{1-i}{\alpha - 1}} \text{ and } b(y)_i = \max(y_i - d, 0). \tag{11}$$

Here d is a neutral vote and α is the viewing halflife. The normalized ERU can also be defined in a similar manner to NDCG. The (normalized) ERU is often used in collaborative filtering for recommender systems (Breese et al., 1998; Basilico and Hofmann, 2004) where the lists of items tend to be very short.

The algorithm we are going to propose is able to deal with *all* losses presented here in a systematic fashion without the need of designing customized solutions for each of them. Unfortunately it seems that Mean Average Precision (MAP) can not be written in the form (2). But for this particular measure, another formulation of SVMs with structured output has been proposed Yue et al. (2007).

2.4 A Linear Model

For reasons of computational convenience we choose a linear model to describe the score function $g(d, q)$:

$$g(d, q) = \langle \phi(d, q), w \rangle, \tag{12}$$

where w denotes the parameters of the linear model. The vector $\phi(d, q)$ may be consist of explicit features, such as BM25 (Robertson et al., 1994; Robertson and Hull, 2000; Craswell et al., 2005), date information, page rank like information (Kleinberg, 1999; Page et al., 1998), or click-through logs (Joachims, 2002). But we can also think of (12) as a *generalized* linear model where ϕ is constructed using neural networks, decision trees, Reproducing Kernel Hilbert Spaces, or other elementary estimators.

Note that there are quite different approaches than linear modeling. For instance, Matveeva et al. (2006) use a cascade of rankers not unlike Viola and Jones (2004) and Romdhani et al. (2000) in the context of face detection. In practice, successive stages for eliminating irrelevant pages are clearly useful. Using techniques such as functional gradient descent (Mason et al., 2000; Friedman et al., 1998) could be used to achieve this goal. In practice, their deployment is quite application dependent and hence we omit a more detailed discussion.

3. Learning

Recent developments in structured estimation by Tsochantaridis et al. (2005); Taskar et al. (2004) have led to a large number of new algorithms which deal with what is essentially an inverse optimization problem. We will be following this strategy for the purpose of finding a good score function.

3.1 From Ranking to Regularized Risk Minimization

In (1) we already outlined that given a set of m observations, minimizing the empirical risk may be a desirable goal. However, it is clear that if we do so at all cost, we are likely to encounter overfitting. For instance, for a sufficiently rich enough function class we could simply try finding some g such that $g(d_i, q) = y_i$ for all documents and queries. Hence we need to impose additional regularity requirements in the form of a regularizer $\Omega : \mathcal{G} \to \mathbb{R}_0^+$. This means that instead of $R_{\text{emp}}[g]$ we aim to minimize the regularized risk functional

$$R_{\text{reg}}[g] := \frac{1}{m} \sum_{i=1}^{m} \Delta(y^i, \pi(D^i, q^i, g)) + \lambda \Omega[g]. \tag{13}$$

Here $\lambda > 0$ is a regularization constant. A popular choice for Ω in the case of linear models is $\Omega[g] = \frac{1}{2} \|w\|_2^2$. This is the regularizer we will be using in this paper (hence we omit the subscript in the remaining of this paper). Note, however, that different norms and exponents are equally popular. For instance, for sparse expansions one typically uses $\Omega[g] = \|w\|_1$.

The problem with (13) is that the regularized risk function is not convex. Indeed, it is not even continuous in g, since $\pi(D, q, g)$ may only take on a finite number of values.

Optimization problems of this kind have been addressed in recent years by structured estimation (Taskar et al., 2004; Tsochantaridis et al., 2005). The basic idea is to upper bound (13) by convex function in g and to minimize this upper bound.

3.2 Structured Estimation

Large margin structured estimation is a general strategy to solve estimation problems of mapping $\mathfrak{X} \to \mathfrak{Z}$ by solving related optimization problems. More concretely it solves the estimation problem of finding a matching $z \in \mathfrak{Z}$ from a set of (structured) estimates, given patterns $x \in \mathfrak{X}$, by finding a function $f(x, z)$ such that

$$z^*(x) := \underset{z \in \mathfrak{Z}}{\operatorname{argmax}} f(x, z). \tag{14}$$

This means that instead of finding a mapping $\mathfrak{X} \to \mathfrak{Z}$ directly, the problem is reduced to finding a real valued function on $\mathfrak{X} \times \mathfrak{Z}$.

In the ranking case, x corresponds to the collection of documents D with a matching query q, whereas z denotes a permutation π. In order to take advantage of the optimization framework offered by structured estimation we need to rewrite $\pi(D, q, w)$[3] in the form of $\pi = \operatorname{argmax}_{\pi'} f(D, q, \pi')$. This is easily achieved by defining:

$$f_w(D, q, \pi) := \sum_{i=1}^{l} c_{\pi(i)} g(d_i, q) = c[\pi]^\top \phi(D, q) w \tag{15}$$

for a monotonically decreasing c_i, and where $\phi(D, q)$ is a shorthand for the matrix

$$\phi(D, q) := (\phi(d_1, q), \ldots, \phi(d_l, q))^\top. \tag{16}$$

From the rearrangement inequality (Theorem 3), it is clear that the permutation maximizing $f_w(D, q, \pi)$ is obtained by sorting in decreasing order the scores $g(d_i, q)$. The choice of c depends on the application and we will discuss it in Section 6. While this reformulation does not immediately offer computational benefits, note that f is linear in w. This means that any convex upper bound using f will also result in a convex upper bound with respect to w and therefore be amenable to efficient optimization.

3.3 A Convex Upper Bound

We now describe the convex upper bounding technique of (Tsochantaridis et al., 2005; Taskar et al., 2004). The basic strategy is to obtain a convex upper bound on the empirical loss $\Delta(y, \pi(D, q, w))$ of (13)

Definition 4 *For a query q, associated documents D and labels y, define the loss for a given function f_w as*

$$l_{\text{convex}}(f_w, D, q, y) := \max_{\pi} f_w(D, q, \pi) - f_w(D, q, \bar{\pi}) + a[\bar{\pi}]^\top b(y) - a[\pi]^\top b(y), \tag{17}$$

where $\bar{\pi}$ is any maximizer of $a[\pi]^\top b(y)$.

---

3. We use interchangeably g and w as g is the linear function with parameters w.

Convexity of (17) with respect to f_w follows from the fact that a pointwise maximum of convex functions is convex. For the upper bound, we substitute $\pi^* := \pi(D, q, w)$, a maximizer of $f_w(D, q, \pi)$ into the loss term to obtain

$$l_{\text{convex}}(f_w, D, q, y) \geq f_w(D, q, \pi^*) - f_w(D, q, \bar{\pi}) + a[\bar{\pi}]^\top b(y) - a[\pi^*]^\top b(y) \tag{18}$$

$$\geq a[\bar{\pi}]^\top b(y) - a[\pi^*]^\top b(y) = \Delta(y, \pi^*). \tag{19}$$

Combining equations (15) and (17), we obtain the following convex upper bound on the regret incurred by ranking documents according to the score function $g(d, q) = \langle \phi(d, q), w \rangle$:

$$l_{\text{convex}}(w, D, q, y) = \max_\pi \left[c[\pi]^\top \phi(D, q)w - a[\pi]^\top b(y) \right] - c[\bar{\pi}]^\top \phi(D, q)w + a[\bar{\pi}]^\top b(y) \tag{20}$$

We will discuss methods for minimizing (20) in Section 4. However, before we do so, let us review an alternative tighter bound on the empirical risk.

3.4 A Tighter Nonconvex Upper Bound

The problem with the loss function (17) is that it is not tight enough in some cases. For instance, for large values of f_w the convex upper bound may keep on increasing while the actual regret of performing suboptimal ranking is obviously bounded. **?** establish that, indeed, the structured max-margin setting is not statistically consistent (although it usually works well in practice).[4]

Problems with the convex bound have first been observed in (Chapelle et al., 2007) where the optimal solution using the convex loss function is $w = 0$ (using the Ohsumed dataset of the Letor package). This is clearly a bad solution. Intuitively the problem stems from the dependence on $\bar{\pi}$. There might be indeed several permutations maximizing $a[\pi]^\top b(y)$ and it is not satisfactory to pick an arbitrary one. A better choice is to take the minimum over all such maximizers $\tilde{\pi}$ via:

$$\min_{\tilde{\pi} \in \tilde{\Pi}} \max_\pi f_w(D, q, \pi) - f_w(D, q, \tilde{\pi}) + a[\bar{\pi}]^\top b(y) - a[\pi]^\top b(y), \tag{21}$$

with $\tilde{\Pi} = \{\tilde{\pi}, \ a[\tilde{\pi}]^\top b(y) = \max_\pi a[\pi]^\top b(y)\}$. Pushing this idea further, note that

$$\max_\pi f_w(D, q, \pi) - f_w(D, q, \tilde{\pi}) + a[\bar{\pi}]^\top b(y) \geq a[\operatorname*{argmax}_\pi f_w(D, q, \pi)]^\top b(y)$$

holds for *any* choice of $\tilde{\pi}$. This can be seen by plugging $\operatorname{argmax}_\pi f(w(D, q, \pi)$ into the left hand side of the inequality. Hence we may take the minimum with respect to $\tilde{\pi}$ to obtain a variational upper bound on the ranking regret. As a result, the "reference" permutation $\tilde{\pi}$ is still almost as good as any permutation from Π. The resulting loss is:

$$l_{\text{nonconvex}}(f, D, q, y) := \min_{\tilde{\pi}} \max_\pi \left[f_w(D, q, \pi) - f_w(D, q, \tilde{\pi}) + \max_{\bar{\pi}} a[\bar{\pi}]^\top b(y) - a[\pi]^\top b(y) \right]$$

$$= \max_\pi \left[f_w(D, q, \pi) - a[\pi]^\top b(y) \right] + \max_{\bar{\pi}} a[\bar{\pi}]^\top b(y) - \max_{\tilde{\pi}} f_w(D, q, \tilde{\pi})$$

---

4. An exponential families version of the objective function, which is consistent, that is $p(\pi|f) \propto e^{f(\pi)}$ leads to an NP hard problem when computing the normalization, since it involves computing the permanent of a matrix.

Combining this with (15) we can rewrite the non-convex loss as a function of w:

$$l_{\text{nonconvex}}(w, D, q, y) = \max_{\pi} \left[c[\pi]^{\top} \phi(D, q) w - a[\pi]^{\top} b(y) \right]$$
$$+ \max_{\bar{\pi}} a[\bar{\pi}]^{\top} b(y) - \max_{\tilde{\pi}} c[\tilde{\pi}]^{\top} \phi(D, q) w \qquad (22)$$

We can see this nonconvex loss is very similar to the original convex one (20). The only difference is in the third term: instead of considering the optimal ranking $\bar{\pi}$, it involves the *current* best ranking $\tilde{\pi}(q, D, w) = \arg\max_{\tilde{\pi}} c[\tilde{\pi}]^{\top} \phi(D, q) w$. Several comments about this new loss function:

1. It is smaller than the original loss. This can be seen by replacing $\tilde{\pi}$ with $\bar{\pi}$.

2. It still an upper bound on the loss.

3. It is non-convex: the first two terms are convex, but the third one is concave.

In summary, we have a tighter upper bound on the loss, but the convexity is lost. Optimization over this nonconvex upper bound is more difficult. However, as we shall see, DC programming (An and Tao, 2005), also called the Concave-Convex Procedure (Yuille and Rangarajan, 2003), or the MM algorithm (Hunter and Lange, 2004), can be used to find approximate solutions to the optimization problem. The basic idea of all those methods is to use successive linear upper bounds on the concave part of (22) for solving a sequence of convex optimization problems.

4. Optimization

4.1 Objective Functions

Let us briefly review the objective functions arising from the convex and the nonconvex upper bound of the regularized risk, since it is those which we need to minimize. With the regularizer $\Omega[g] = \frac{1}{2} \|w\|^2$, and discarding constant terms, we have the following optimization problem arising from the convex upper bound of Section 3.3.

$$\underset{w}{\text{minimize}} \ \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{m} \max_{\pi} \left[c[\pi]^{\top} \phi(D^i, q^i) w - a[\pi]^{\top} b(y^i) \right] - c[\bar{\pi}_i]^{\top} \phi(D^i, q^i) w \qquad (23)$$
$$\text{where } \bar{\pi}_i = \underset{\pi}{\arg\max} \, a[\pi]^{\top} b(y^i).$$

This is a convex minimization problem, being the combination of squared norm and the maximum over a set of linear functions, hence it has a unique minimum value.

A similar optimization problem can be obtained by replacing the convex upper bound by its nonconvex counterpart which was introduced in Section 3.4. We arrive at the following:

$$\underset{w}{\text{minimize}} \ \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{m} \max_{\pi} \left[c[\pi]^{\top} \phi(D^i, q^i) w - a[\pi]^{\top} b(y^i) \right] - \max_{\pi} c[\pi]^{\top} \phi(D^i, q^i) w \qquad (24)$$

Clearly, this problem is nonconvex. However, it is possible to obtain a succession of convex

upper bounds which can be minimized in the same fashion as when minimizing (23). This is the idea behind DC programming (An and Tao, 2005) and we now give a template of the basic idea.

For a given function $h(x)$ assume that $u(x|x')$ satisfies the conditions $u(x|x) = h(x)$, u is convex in x for all x', and $u(x|x') \geq h(x)$ for all x. In this case $h(x)$ can be minimized (for a local minimum) by using the iterative algorithm 1.

Algorithm 1 DC programming

Require: $u(x|x')$ such that $u(x|x) = h(x)$, u is convex in x, and $u(x|x') \geq h(x)$.
Require: Initial x
  repeat
    Update $x' \longleftarrow x$
    Improve estimate via $x = \operatorname{argmin}_x u(x|x')$
  until converged, e.g. by measuring $u(x|x') - h(x)$

An upper bound in (24) can be obtained by a linear approximation of the concave term. Denote by

$$\pi_i^*(w) := \operatorname*{argmax}_\pi c[\pi]^\top \phi(D^i, q^i) w \tag{25}$$

the best ranking for document collection D^i and query q^i given the current weight vector. In this case the following bound holds:

$$\max_\pi c[\pi]^\top \phi(D^i, q^i) w \geq c[\pi_i^*(w')]^\top \phi(D^i, q^i) w \tag{26}$$

The equality holds for $w = w'$. Hence we may use (26) to successively improve the solution of the nonconvex optimization problem (24). Plugging (26) into (24) and using the above algorithmic template we obtain Algorithm 2 for solving the nonconvex optimization problem.

The resulting optimization problem is very similar to (23) with the only difference that the reference permutation $\bar{\pi}_i$ has been replaced by $\pi_i^*(w)$, the currently best estimate of the permutation ranking the documents. Given the similarity it turns out that both the convex upper bound and a given step in Algorithm 2 can be solved by the same optimization approach. Key is the computation of subgradients of the objective function with respect to the parameter vector w.

4.2 Linear Assignment Problem

To compute the objective functions (23) or (24), we need to compute a maximum over the set of permutations. We are not only interested in the maximum, but also in the maximizer since this will be needed for the subgradient computation. The maximizer of interest is:

$$\tilde{\pi} := \operatorname*{argmax}_\pi c[\pi]^\top \phi(D, q) w - a[\pi]^\top b(y) \tag{27}$$

$$= \operatorname*{argmax}_\pi \operatorname{tr} \pi E \text{ where } E_{ij} = c_i \langle \phi(d_j, q), w \rangle - a_i b(y)_j. \tag{28}$$

Algorithm 2 Successive convex approximation

Find an initial solution, by standard regression for instance:

$$w^* := \underset{w}{\operatorname{argmin}} \ \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{m} ||\phi(D^i, q^i)w - b(y^i)||_2^2.$$

repeat

   Compute $\pi_i^* := \operatorname{argmax}_\pi c[\pi]^\top \phi(D^i, q^i)w^*$ for all $i \in \{1, \ldots, m\}$.
   Solve the optimization problem

$$w^* := \underset{w}{\operatorname{argmin}} \ \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{m} \max_\pi \left[c[\pi]^\top \phi(D^i, q^i)w - a[\pi]^\top b(y^i) \right] - c[\pi_i^*]^\top \phi(D^i, q^i)w$$

until converged

As explained in definition 1, we have identified the permutation π with its representation as a permutation matrix.

The optimization problem of (28) is a so-called *Linear Assignment Problem* which can be solved efficiently by a number of algorithms, including the Hungarian Marriage algorithm of Kuhn (1955) and Munkres (1957). These papers implied an algorithm with $O(l^3)$ cost in the number of terms. Later, Karp (1980) suggested an algorithm with expected quadratic time in the size of the assignment problem (ignoring log-factors). Finally, Orlin and Lee (1993) propose a linear time algorithm for large problems.

Once we find $\tilde{\pi}$ we may compute the gradient of (23). It is given by the sum over the gradients of the individual upper bounds plus the derivative of the regularizer.

$$\lambda w + \sum_{i=1}^{m} \left[c[\tilde{\pi}_i] - c[\bar{\pi}_i] \right]^\top \phi(D^i, q^i) \tag{29}$$

The gradient arising from Algorithm 2 is given by

$$\lambda w + \sum_{i=1}^{m} \left[c[\tilde{\pi}_i] - c[\pi_i^*] \right]^\top \phi(D^i, q^i) \tag{30}$$

Note that objective is differentiable everywhere except at points where one of the maximizers $\tilde{\pi}_i$ is not unique. In this case, we select such a maximizer at random, which amounts to say that we consider an arbitrary *subgradient*.

The only difference between the two problems above is that in the second case we use π_i^* instead of $\bar{\pi}_i$. Now that we have the derivatives we can substitute them directly into algorithms which use gradient information for optimization.

4.3 Nonsmooth optimization

Because of the max operation in objective functions (23) and (24), we can not use standard optimization techniques[5], but need to resort to *nonsmooth optimization* algorithms (**?**, Part II). One of the most popular is the so-called *bundle* method (**?**) which combines the ideas of cutting planes – by building iteratively a piecewise linear lower bound of the function – and trust-region for stability. In the case of machine learning, Smola et al. (2007) proposed a simplified bundle method in which the regularizer of the objective function acts as a stabiliizer. It is described in Algorithm 3.

Algorithm 3 Bundle method for regularized empirical risk minimzation

Initialize $w_1 = 0$
for $t = 1$ **to** maxiter **do**
  Let $R_t[w] := \max_{i \leq t} \langle \partial_w R_{\text{emp}}[w_t], w - w_t \rangle + R_{\text{emp}}[w_t]$
  $w_{t+1} \leftarrow \underset{w}{\text{argmin}} \dfrac{\lambda}{2} \|w\|^2 + R_t[w]$
  break if $R_{\text{emp}}[w_{t+1}] - R_t[w_{i+1}] < \epsilon$
end for

4.4 Online Algorithms

An alternative approach, in particular with large amounts of data, is to use a stochastic gradient descent procedure, such as those proposed by Nesterov and Vial (2000); Bottou and LeCun (2005) or more recently in the context of structured output learning by Ratliff et al. (2007). Let us rewrite the optimization problems (23) and (24) as follows:

$$\underset{w}{\text{minimize}} \ \sum_{i=1}^{m} \underbrace{\frac{\lambda}{2m} \|w\|^2 + \max_{\pi} \left[c[\pi]^\top \phi(D^i, q^i) w - a[\pi]^\top b(y^i) \right] - c[\bar{\pi}_i]^\top \phi(D^i, q^i) w}_{:= r_i^c(w)} \tag{31}$$

$$\underset{w}{\text{minimize}} \ \sum_{i=1}^{m} \underbrace{\frac{\lambda}{2m} \|w\|^2 + \max_{\pi} \left[c[\pi]^\top \phi(D^i, q^i) w - a[\pi]^\top b(y^i) \right] - \max_{\pi} c[\pi]^\top \phi(D^i, q^i) w}_{:= r_i^n(w)} \tag{32}$$

Hence we may write the minimization problem as one of averaging over m risk terms $r_i^c(w)$ and $r_i^n(w)$ respectively. For convex problems, that is, for $r_i^c(w)$ the following update scheme

$$w \longleftarrow w - \eta_t \partial_w r_t(w) \tag{33}$$

converges for a variety of schedules in choosing η. For instance, for $\eta_t = (t + c)^{-1}$ we obtain $O(t^{-1})$ convergence after t steps due to the added regularizer $\|w\|^2$ which ensures strong convexity. For the nonconvex part, the same type of converge holds, but of course only to a local minimum.

  This procedure is also preferable to a full batch update since we only need to solve one linear assignment problem and one sorting operation at a given time. In the batch setting,

---

5. It is noteworthy that, on nonsmooth problems, a smooth optimization algorithm such as BFGS has been shown to find very good solutions (**?**). There is however no theoretical convergence guarantee in general.

we obtain an upper bound on *all* the losses whereas in the online setting we will effectively have an upper bound for the improved estimate of the parameter vector. This allows for faster convergence.

4.5 Constrained Quadratic Programming

A third option for solving the minimization problems (23) and (24) is to rewrite the problem as a constrained quadratic program along the lines of Taskar et al. (2004); Tsochantaridis et al. (2005). Rewriting the maximum operations in (23) as constraints we have the following equivalent optimization problem:

$$\underset{w,\xi}{\text{minimize}} \quad \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{m} \xi_i \tag{34a}$$

$$\text{subject to} \quad [c[\pi] - c[\bar{\pi}_i]]^\top \phi(D^i, q^i)w + [a[\bar{\pi}_i] - a[\pi]]^\top b(y^i) \leq \xi_i \text{ for all } \pi \tag{34b}$$

$$\text{and } \bar{\pi}_i = \underset{\pi}{\text{argmax}} \, a[\pi]^\top b(y^i).$$

This optimization problem is identical in form to those given by Taskar et al. (2004) and Tsochantaridis et al. (2005). This allows us to read off the dual problem directly without the need for further proof:

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2} \sum_{i,j,\pi,\pi'} \alpha_{i\pi} \alpha_{j\pi'} \bar{k}((D^i, q^i, \bar{\pi}_i, \pi), (D^j, q^j, \bar{\pi}_j, \pi')) - \sum_{i,\pi} [a[\bar{\pi}_i] - a[\pi]]^\top b(y^i)\alpha_{i\pi}$$

$$\text{subject to} \quad \sum_{\pi} \alpha_{i\pi} \leq \lambda^{-1} \text{ and } \alpha_{i\pi} \geq 0 \text{ for all } i \text{ and } \pi.$$

Here the kernel $\bar{k}$ is given by the inner product of the compound feature maps of document collections D, queries q and permutations π into a joint feature space via

$$\bar{k}((D, q, \bar{\pi}, \pi), (D', q', \bar{\pi}', \pi')) := [c[\pi] - c[\bar{\pi}]]^\top \phi(D, q)\phi(D', q')^\top [c[\pi'] - c[\bar{\pi}']] \tag{35}$$

$$\tag{36}$$

Finally, the optimal weight parameter w^* is given by

$$w^* = \sum_{i,\pi} \alpha_{i\pi} \phi(D^i, q^i)^\top [c[\pi] - c[\bar{\pi}_i]]. \tag{37}$$

Solving and evaluating such an optimization problem presents a formidable challenge since it contains an exponential number of variables $\alpha_{i\pi}$. It turns out that by using column generation (Tsochantaridis et al., 2005) one may find an approximate solution in polynomial time. The key idea in this is to check the constraints (34b) to find out which of them are violated for the current set of parameters and to use this information to improve the value of the optimization problem. We omit details: suffice it to say that the same linear assignment procedure which is used for computing the subgradients can again be used in this context.

5. The `SmoothGrad` method

We now describe an alternative approach for optimizing ranking measures. The rationale is to compare our upper bound with an *approximation* of the ranking score to assess whether the approximation provides any benefit. The basic idea is to perform a direct gradient descent optimization on a smoothed version of the NDCG measure. For this reason, we call this method `SmoothGrad`; its performance will be evaluated in the experimental section. A similar idea has been independently explored by Taylor et al. (2008).

Let us for rewrite the ranking score formula (9) as:

$$\sum_{i,j} a_j b(y)_i h_{ij},$$

where h_{ij} is the indicator variable: "Is document i at the j-th position in the ranking?".

Having written the ranking score in this equivalent form, the only thing we have to do now is to "soften" the indicator variable h_{ij}. Several choices are possible. We consider the following:

$$\tilde{h}_{ij} = \exp\left(-\frac{(\phi(d_i, q)^\top w - \phi(d_{\pi(j)}, q)^\top w)^2}{2\sigma^2}\right),$$

where $\pi(j)$ is the index of j-th document in the ranking induced by w, and σ is a parameter controlling the amount of smoothing: when σ goes to 0, $\tilde{h}_{ij}$ converges to h_{ij} (because $h_{ij} = 1$ if and only if $\pi(j) = i$).

For a given query q, let us define the smooth score of this query as A_q,

$$A_q(w, \sigma) := \sum_j \frac{\sum_i b(y)_i \tilde{h}_{ij}}{\sum_i \tilde{h}_{ij}}.$$

Note that $A_q(w, \sigma)$ is continuous but non-differentiable at points where there is a tie in the ranking. But since it is differentiable almost everywhere, we did not encounter any problem with a conjugate gradient descent optimizer.

Finally we added an annealing procedure on σ in an outer loop in order to alleviate the local minimum problem inherent to the minimization of a non-convex function. The algorithm is described in Algorithm 4.

Algorithm 4 `SmoothGrad`: Minimization of a smooth score by gradient descent

---

1: Find an initial solution w_0 (by regression for instance).
2: Set $w = w_0$ and σ to a large value.
3: Starting from w, minimize by (conjugate) gradient descent

$$\lambda \|w - w_0\|^2 + \sum_q A_q(w, \sigma).$$

4: Divide σ by 2 and go back to 3 (or stop if converged).

---

6. Experiments

We compare our proposed algorithm on three different datasets: the Ohsumed dataset, part of the Letor package[6] and two datasets from different web search engines. The reason for choosing Ohsumed over Trec (the other dataset part of Letor) is that Trec contains fewer queries and it has only 2 relevance levels and is thus less interesting from a ranking perspective.

In both cases, we compare the NDCG at different truncation levels with some baseline algorithms. Note that we only consider linear algorithms.

Even though there is no clear state-of-the-art algorithm, we have identified based on the literature and our own experience two popular methods in web search ranking that we will compare against: least-squared regression (Cossock and Zhang, 2006) on the output relevance levels $0, 1, 2, 3, 4$ and RankSVM (Herbrich et al., 2000; Joachims, 2002). Note that even though regression is a very simple method, it can sometimes achieve better performance than a more sophisticated method such as LambdaRank (Li et al., 2008, Figure 1). This might be due in part to the fact that regression is provably *consistent* (Cossock and Zhang, 2006).

We also compared our method to `SmoothGrad` which attempts to directly minimize by gradient descent a smooth version of the NDCG. This method, described in Section 5, is similar in spirit to the one proposed by Taylor et al. (2008).

In our experiments, λ is chosen by the standard process of cross validation in a validation set. The choice of c_i is explained in details in each experiment.

6.1 Web search dataset 1

In this first experiment, we present results on a ranking dataset coming from a commercial search engine. It consists of about 1500 queries and 50k (query,url) pairs. For each of these pairs, a relevance grade has been assigned out of 5 possible relevance levels. Each (query,url) pair is made of several hundred features. A fifth of the data is held out as a test set and a fifth as a validation set. The regularization parameter is tuned on this validation set. For each k from 1 to 10, the NDCG@k is independently optimized. The parameters c_i in (15) are chosen to be $c_i = \max(k + 1 - i, 0)$. We also tried to take $c_i = a_i$ but the results were not as good.

Results are shown in Figure 1. The gain in performance is about 2%, which is considered very satisfactory in the web search ranking community. For each query, we add the NDCGs from k = 1 to 10 and perform a Wilcoxon signed-rank test. The p-value corresponding to the difference between our method and regression (resp RankSVM) is 3% (resp 7%). From a computational point of view, the training is relatively fast. For all the different methods described in Section 4, the training time is of the order of 15 minutes.

6.2 Web search dataset 2

In the second experiment, we use a different dataset from another search engine which has 1000 queries for training, 1000 queries for validation and 1000 for testing. The dataset has 5 levels of relevance and the number of documents per query is in the order of 50.
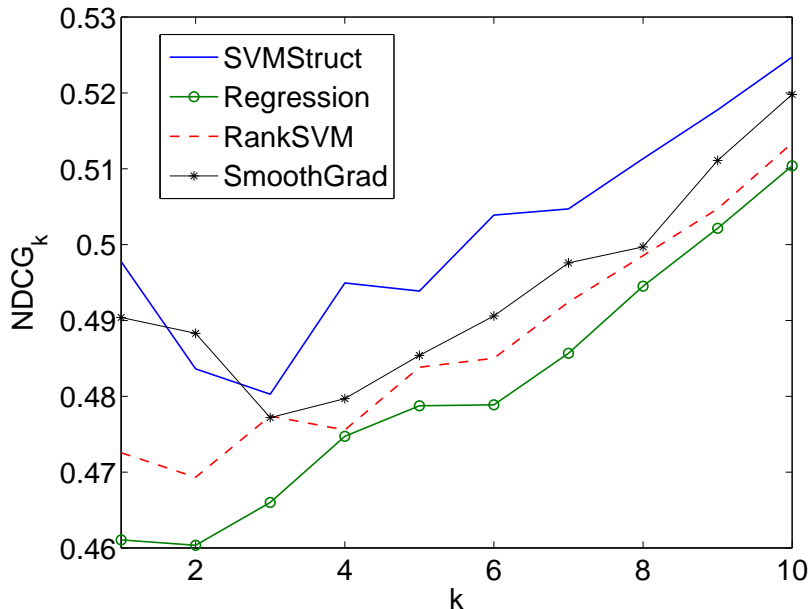
---

6. Available at `http://research.microsoft.com/users/tyliu/LETOR/`

Figure 1: NDCG performance results on web search dataset 1.

The parameters c_i in (15) are chosen to be $c_i = i^{-d}$, where d is an hyperparameter controlling the decay of the coefficients. We train our model on the training set, cross validate our λ, d using the validation set and predict on the test set. We report the results of the experiments and compare against RankSVM, standard least square regression and `SmoothGrad`. The results show that our methods perform very well on this dataset. The improvement is statistically significant (p-value $< 1\%$).

Choice of c Execpt the regularization parameter, the only design choice in our algorithm are the ceofficients c_i of equation (15). Since we have little theoretical guidance with regard to this choice, we investigated experimentally on this dataset the effect of different c_i. Clearly c_i needs to be a monotonically decreasing function. We chose $c_i = i^{-d}$ for $d \in \left\{ \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1, 2, 3 \right\}$ and $c_i = 1/\log_2(i+1)$ and $c_i = 1/\log_2(\log_2(i+1)+1)$.

We found experimentally that the differences between the various schemes are not as dramatic as the improvement obtained by using SVMStruct instead of other algorithms. To summarize the results we show the difference in performance in Figure 3 for NDCG@10. Note that the difference in terms of NDCG accuracy resulted by taking different c_i will decrease when the sample size increases. The rate of convergence is suspected to be $1/\sqrt{m}$. An possible interpretation is that the choice of c_i can be considered prior knowledge. Thus with increasing sample size, we will need to rely less on this prior knowledge and a reasonable choice of c_i will suffice.
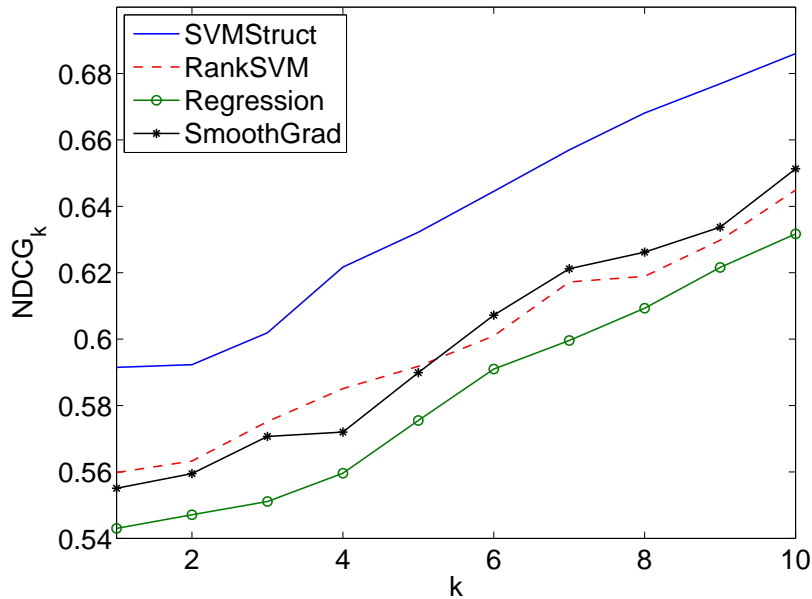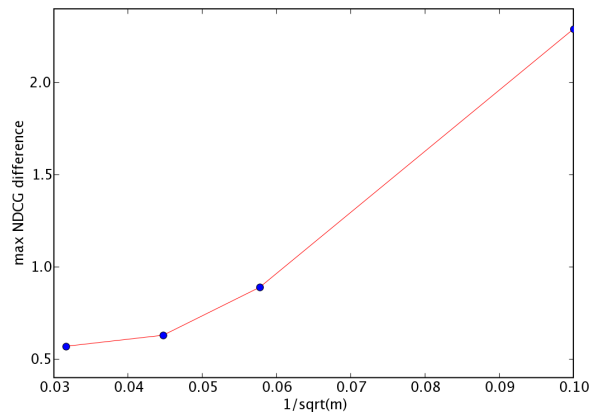
Figure 2: NDCG performance results on web search dataset 2.

Figure 3: Maximum difference (in percent point) in NDCG@10 for different choices of c with respect to $1/\sqrt{m}$.

6.3 Ohsumed

The Ohsumed dataset has 106 queries and we used the same 5 splits training / validation / test as provided in the Letor distribution. Each (query,document) pair has 25 features and 3 possible relevance scores.

On this dataset, the convex formulation gives the solution $w = 0$ even with very small values of λ. As explained in Section 3.4, the problem is likely to be related to underfitting. We then use the nonconvex formulation (24). Since this optimization problem is non-convex,

the starting point is important. We set it to w_0, where w_0 is found by regression on the grades. Also the regularizer is changed from $\|w\|^2$ to $\|w - w_0\|^2$.

Finally we observed that optimizing the NDCG@k, as done above, gave unstable results in particular during the model selection phase. This is probably because the dataset if very small. Instead we optimize the NDCG@10 which gives a more stable indicator of the quality of a solution.

We compare our nonconvex method against RankSVM and RankBoost in Figure 4. The standard results for RankSVM and RankBoost are included in the LETOR baselines. Even though there is 2% improvement for NDCG, the difference is not statistically significant. This may be because of the fact that the dataset is small (only 106 queries). We also note that for NDCG@k for small k the performance of `SmoothGrad` is better than SVMStruct. This might be because both methods are non-convex[7], but `SmoothGrad` has a smooth objective function which might be easier to optimize. Nevertheless, both methods are comparable with respect to NDCG@k for large k, which is anyway a more stable of performance when the dataset is small as it is the case here.
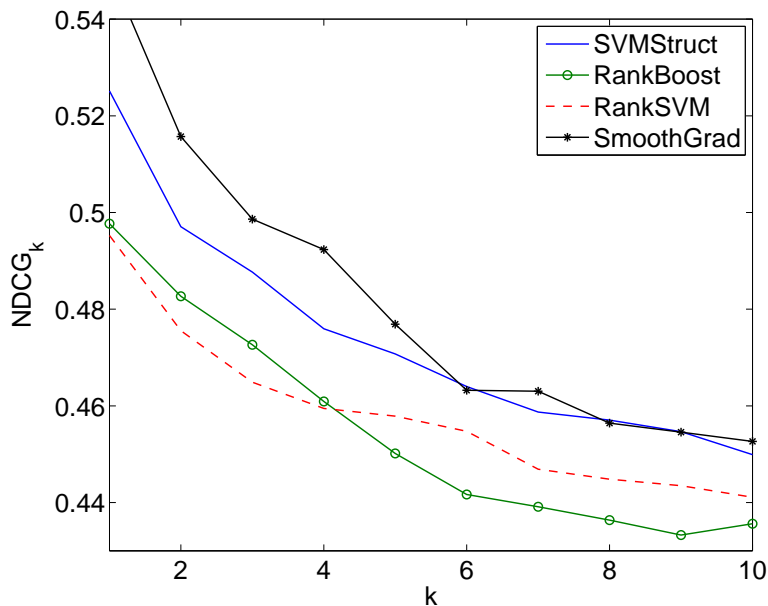

Figure 4: NDCG performance results on OHSUMED dataset.

7. Extensions

So far our discussion was primarily concerned with ranking without any further restrictions. In practice, this is an overly idealized situation and we need to take redundancy constraints and the like into account. We now present three extensions which can be used to adapt ranking to more realistic settings.

---

7. Remember that for this dataset we use the non-convex formulation (24).

7.1 Diversity Constraints

Imagine the following scenario: when searching for 'Jordan', we will find many relevant webpages containing information on this subject. They will cover a large range of topics, such as a basketball player (Mike Jordan), a country (the kingdom of Jordan), a river (in the Middle East), a TV show (Crossing Jordan), a scientist (Mike Jordan), a city (both in Minnesota and in Utah), and many more. Clearly, it is desirable to provide the user with a diverse mix of references, rather than exclusively many pages from the same site or domain or topic range.

One way to achieve this goal is to include an interaction term between the items to be ranked. This leads to optimization problems of the form

$$\underset{\pi \in \Pi}{\text{minimize}} \sum_{ijkl} \pi_{ij} \pi_{kl} c_{ij,kl} \tag{38}$$

where $c_{ij,kl}$ would encode the interaction between items. This is clearly not desirable, since problems of the above type cannot be solved in polynomial time. This would render the algorithm impractical for swift ranking and retrieval purposes.

However, we may take a more pedestrian approach, which will yield equally good performance in practice, without incurring exponential cost. This approach is heavily tailored towards ranking scores which only take the top n documents into account. We will require that among the top n retrieved documents no more than one of them may come from the same source (e.g. topic, domain, subdomain, personal homepage). Nonetheless, we would like to minimize the ranking scores subject to this condition. For this to work we need to restrict the set of possible ranking such that they are compatible with the diversity constraints.

Definition 5 *Denote by $\mathcal{B} := \{B_1, \ldots, B_n\}$ a partition of the set $\{1, \ldots, l\}$ into n subsets, that is $\cup_i B_i = \{1, \ldots, l\}$ and $B_i \cap B_j = \emptyset$ for all $i \neq j$. Then we define $\Pi(\mathcal{B}, k)$ to be the set of retrieval matrices where all $\pi \in \Pi(\mathcal{B}, k)$ satisfy*

$$\pi \in \{0,1\}^{k \times l} \text{ and } \sum_j \pi_{ij} = 1 \text{ and } \sum_i \sum_{j \in B_s} \pi_{ij} \leq 1 \text{ for all } i, j, s. \tag{39}$$

The set $\Pi(\mathcal{B}, k)$ represents all rankings of k out of l objects which have no redundancy within each of the sets B_s. For instance, assume that 6 documents were retrieved from 3 domains, e.g. via a split into $\{1, 2, 3\}, \{4, 5\}$ and $\{6\}$. Moreover, assume that we wanted to retrieve only two documents. In this case, pairs such as $(1, 5)$ or $(6, 4)$ are legitimate, whereas pairs of the form $(1, 3), (5, 4)$ are clearly not.

Retrieving the best k documents subject to the constraints imposed by $\Pi(\mathcal{B}, k)$ at deployment time is easy since all we need to do is compute the scores $g(d_i, q)$ and sort the list. Subsequently we pick documents in descending order of their scores, choosing at most one from each set B_i. The latter can be achieved in linear time given that we already have $\mathcal{B}$ at our disposition. This simple procedure maximizes an objective function of the form

$$f(D, q, \pi) := \sum_{i=1}^{k} c_i g(d_{\pi(i)}, q) = c^\top \pi g(D, q) \tag{40}$$

where the coefficients c_i are chosen in descending order just as in (15). With some abuse of notation we identified $\pi(i)$ with the single nonzero element in row i of $\pi \in \Pi(\mathcal{B}, k)$. Note that (40) is identical to (15) with the exception that we only consider the first k terms whereas (15) considers a ranking order of *all* documents.

The slightly more tricky part is how to deal with the modified set of retrieval matrices for training purposes. One may check that (20) and (22) apply without modification, with the only exception that now the maximization over the entire set of permutation matrices is replaced by a maximization over $\pi \in \Pi(\mathcal{B}, k)$. It is clear that imposing additional constraints will lead to a permutation retrieving less informative documents. It is arguably more useful, though, to obtain a *diverse* set of potentially less relevant documents covering a number of topics rather than many relevant documents on a single topic.

Key in solving this new optimization problem is the ability to perform maximization over all $\pi \in \Pi(\mathcal{B}, k)$.

Optimization problem (28) now becomes:

$$\underset{\pi}{\text{maximize}} \ \ \text{tr} \ \pi E \text{ subject to } \pi \in \Pi(\mathcal{B}, k) \tag{41}$$

The above integer linear program can be relaxed by replacing the integrality constraint $\pi_{ij} \in \{0, 1\}$ of (39) by $\pi_{ij} \geq 0$ leading to a linear program which can be solved efficiently

The following theorem shows that this relaxation will lead to integral solutions:

Theorem 6 (Heller and Tompkins (1956)) *An integer matrix A with $A_{ij} \in \{0, \pm 1\}$ is totally unimodular if no more than two nonzero entries appear in any column, and if its rows can be partitioned into two sets such that:*

1. *If a column has two entries of the same sign, their rows are in different sets;*

2. *If a column has two entries of different signs, their rows are in the same set.*

Corollary 7 *The linear programming relaxation of $\Pi(\mathcal{B}, k)$ has an integral solution.*

Proof All we need to show is that each term π_{ij} in the constraints of $\Pi(\mathcal{B}, k)$ only shows up exactly twice with coefficient 1. This is clearly the case since $\Pi(\mathcal{B}, k)$ is a partition of $\{1, \ldots, l\}$, which accounts for one occurrence, and the assignment constraints which account for the other occurrence. Hence Theorem 6 applies. ∎

An experimental evaluation of this setting is the subject of future research. In particular, being able to obtain publicly accessible sets of type $\mathcal{B}$ is a major challenge.

7.2 Ranking Matrix Factorization

An obvious application of our framework is matrix factorization for collaborative filtering. The work of Srebro and Shraibman (2005); Rennie and Srebro (2005); Srebro et al. (2005b) suggests that regularized matrix factorizations are a good tool for modeling collaborative filtering applications. More to the point, Srebro and coworkers assume that they are given a sparse matrix X arising from collaborative filtering, which they would like to factorize.

More specifically, the entries X_{ij} denote ratings by user i on document/movie/object j. The matrix $X \in \mathbb{R}^{m \times n}$ is assumed to be sparse, where zero entries correspond to

(user,object) pairs which have not been ranked yet. The goal is to find a pair of matrices U, V such that $UV^\top$ is close to X for all nonzero entries. Or more specifically, such that the entries $[UV^\top]_{ij}$ can be used to recommend additional objects.

However, this may not be a desirable approach, since it is, for instance, completely irrelevant how accurate our ratings are for undesirable objects (small X_{ij}), as long as we are able to capture the users preferences for desirable objects (large X_{ij}) accurately. In other words, we want to model the user's *likes* well, rather than his *dislikes*. In this sense, any indiscriminate minimization, e.g. of a mean squared error, or a large margin error for X_{ij} is inappropriate.

Instead, we may use a ranking score such as those proposed in Section 2.2 to evaluate an entire *row* of X at a time. That is, we want to ensure that X_{ij} is well reflected as a whole in the estimates for *all objects j for a fixed user i*. This means that we should be minimizing

$$R_{\mathrm{emp}}[U, V, X] := \sum_i l_{\mathrm{convex}}([UV^\top]_{i\cdot}, X_{i\cdot}) \tag{42}$$

where l_{convex} is defined as a ranking score. It is understood that the loss is evaluated over the nonzero terms of X_{ij} only. Weimer et al. (2008) use this formulation to perform collaborative filtering on large scale datasets.

Note that by linearity this upper bound is convex in U and V respectively, whenever the other argument remains fixed. Moreover, note that $R_{\mathrm{emp}}[U, V, X]$ decomposes into m independent problems in terms of the users $U_{i\cdot}$, whenever V is fixed, whereas no such decomposition holds in terms of V.

In order to deal with overfitting, regularization of the matrices U and V is recommended. The trace norm $\|U\|_F^2 + \|V\|_F^2$ can be shown to have desirable properties in terms of generalization (Srebro et al., 2005a). This suggests an iterative procedure for collaborative filtering:

- For fixed V solve m independent optimization problems in terms of $U_{i\cdot}$, using the Frobenius norm regularization on U.

- For fixed U solve one large-scale convex optimization problem in terms of V.

Since the number of users is typically considerably higher than the number of objects, it is possible to deal with the optimization problem in V efficiently by means of subgradient methods. Details can be found in (Weimer et al., 2008).

7.3 General Position Dependent Loss for Matching

The linear assignment problem that we used to match documents with positions in a list can also be used for general position dependent loss functions. For instance, suppose that we want to have a given topic at a given place in a list (e.g. domestic news, international news, entertainment). In this case we might have overall score functions $g(d_i, q, j)$ which tell us the benefit of placing document d_i at position j, where j denotes the category.

More generally, we can encode matching problems in this fashion where we try finding a function $g(i, j)$ which determines the match quality between i and j. One may check that this also leads to a linear assignment problem which can be solved efficiently in much

the same way as what we discussed in this paper. Caetano et al. (2007) use this idea for matching graphs arising from computer vision problems.

8. Summary and Discussion

In this paper we proposed a general scheme to deal with a large range of criteria commonly used in the context of web page ranking and collaborative filtering. Unlike previous work, which mainly focuses on pairwise comparisons we aim to minimize the multivariate performance measures, or rather a number of upper bounds of them. This has both computational savings, leading to a faster algorithm and practical ones, leading to better performance. This is a practical approximation of the mantra of (Vapnik, 1982) of estimating *directly* the desired quantities rather than optimizing a surrogate function. There are clear extensions of the current work:

- The key point of our paper was to construct a well-designed loss function for optimization. In this form it is completely generic and can be used as a drop-in replacement in many settings. We completely ignored language models (Ponte and Croft, 1998) to parse the queries in any sophisticated fashion.

- Although the content of the paper is directed towards ranking, the method can be generalized for optimizing many other complicated multivariate loss functions.

- We could use our method directly for information retrieval tasks or authorship identification queries. In the latter case, the query q^i would consist of a collection of documents written by one author.

- We may add personalization to queries. This is no major problem, as we can simply add some personal data u_i to $\phi(q^i, d_i, u_i)$ and obtained personalized ranking.

Note that the choice of a Hilbert space for the scoring functions is done for reasons of convenience. If the applications demand Neural Networks or similar (harder to deal with) function classes instead of kernels, we can still apply the large margin formulation. That said, we find that the kernel approach is well suited to the problem.

References

L. T. H. An and P. D. Tao. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133(1–4):23–46, 2005.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proc. Intl. Conf. Machine Learning*, pages 65–72, New York, NY, 2004. ACM Press.

J. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. Sagastizàbal. *Numerical Optimization: Theoretical and Practical Aspects.* Springer, 2nd edition, 2006.

Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

J. S. Breese, D. Heckerman, and C. Kardie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

C. J. Burges, Quoc V. Le, and R. Ragno. Learning to rank with nonsmooth cost functions. In Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, 2007.

C. J. C. Burges. Ranking as learning structured outputs. In S. Agarwal, C. Cortes, and R. Herbrich, editors, *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, 2005.

C.J.C Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hulldender. Learning to rank using gradient descent. In *Proc. Intl. Conf. Machine Learning*, 2005.

T. S. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *Proceedings of the 11th International Conference On Computer Vision (ICCV-07)*, pages 1–8, Los Alamitos, CA, 2007. IEEE Computer Society.

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

O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS workshop on Machine Learning for Web Search*, 2007.

D. Cossock and T. Zhang. Subset ranking using regression. In *Proceedings of Conference on Learning Theory (COLT)*, 2006.

K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

K. Crammer and Y. Singer. Loss bounds for online category ranking. In P. Auer and R. Meir, editors, *Proc. Annual Conf. Computational Learning Theory*, pages 48–62, Berlin, Germany, 2005. Springer-Verlag.

N. Craswell, H. Zaragoza, and S. Robertson. Microsoft cambridge at trec-14: Enterprise track. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, 2005.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford University, Dept. of Statistics, 1998.

I. Heller and C. Tompkins. An extension of a theorem of dantzig's. In H.W. Kuhn and A.W. Tucker, editors, *Linear Inequalities and Related Systems*, volume 38 of *Annals of Mathematics Studies*. 1956.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA, 2000. MIT Press.

D. R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58: 30–37, 2004.

K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 41–48. New York: ACM, 2002.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002.

T. Joachims. A support vector method for multivariate performance measures. In *Proc. Intl. Conf. Machine Learning*, pages 377–384, San Francisco, California, 2005. Morgan Kaufmann Publishers.

R.M. Karp. An algorithm to solve the $m \times n$ assignment problem in exptected time $o(mn \log n)$. *Networks*, 10(2):143–152, 1980.

R. Khanna, U. Sawant, S. Chakrabarti, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46 (5):604–632, November 1999.

H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

Quoc V. Le and A. J. Smola. Direct optimization of ranking measures. *Journal of Machine Learning Research*, 2007. submitted.

D.L. Lee, H. Chuang, and K. Seamons. Document ranking and the vector space model. *IEEE Transactions on Software*, 14(2):67–75, 1997.

A. Lewis and M. Overton. Nonsmooth optimization via BFGS. *SIAM Journal on Optimization*, 2009. to appear.

P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 897–904. MIT Press, Cambridge, MA, 2008.

L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–246, Cambridge, MA, 2000. MIT Press.

I. Matveeva, C. Burges, T. Burkard, A. Laucius, and L. Wong. High accuracy retrieval with multiple nested ranker. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 437–444, 2006.

J. Munkres. Algorithms for the assignment and transportation problems. *Journal of SIAM*, 5(1):32–38, 1957.

Y. Nesterov and J.-P. Vial. Confidence level solutions for stochastic programming. Technical Report 2000/13, Université Catholique de Louvain - Center for Operations Research and Economics, 2000.

J.B. Orlin and Y. Lee. Quickmatch: A very fast algorithm for the assignment problem. Working Paper 3547-93, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, March 1993.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA, November 1998.

J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 275–281. ACM, 1998.

N. Ratliff, J. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March 2007.

J. Rennie and N. Srebro. Fast maximum margin matrix factoriazation for collaborative prediction. In *Proc. Intl. Conf. Machine Learning*, 2005.

M. Richardson, A. Prakash, and E. Brill. Beyond pagerank: machine learning for static ranking. In L. Carr, D. De Roure, A. Iyengar, C.A. Goble, and M. Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW*, pages 707–715. ACM, 2006. URL http://doi.acm.org/10.1145/1135777.1135881.

S. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *Proceedings of the 9th Text REtrieval Conference*, pages 25–40, 2000.

S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Text REtrieval Conference 3*. Department of Commerce, National Institute of Standards and Technology, 1994. NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3).

S. Romdhani, B. Schölkopf, P. Torr, and A. Blake. Fast face detection, using a sequential reduced support vector evaluation. TR 73, Microsoft Research, Redmond, WA, 2000. To appear in: Proceedings of the International Conference on Computer Vision 2001.

C. Rudin. Ranking with a P-norm push. In *Proceedings of the 19th Conference on Learning Theory (COLT)*, volume 4005 of *Lecture Notes in Artificial Intelligence*, pages 589–604. Springer, Berlin, 2006.

G. Salton, editor. *The SMART retrieval system: experiments in automatic document processing.* Prentice-Hall, Englewood Cliffs, US, 1971.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* MacGraw-Hill (New York NY), 1983.

A. J. Smola, S. V. N. Vishwanathan, and Quoc V. Le. Bundle methods for machine learning. In Daphne Koller and Yoram Singer, editors, *Advances in Neural Information Processing Systems 20*, Cambridge MA, 2007. MIT Press.

N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In P. Auer and R. Meir, editors, *Proc. Annual Conf. Computational Learning Theory*, number 3559 in Lecture Notes in Artificial Intelligence, pages 545–560. Springer-Verlag, June 2005.

N. Srebro, N. Alon, and T. Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005a. MIT Press.

N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005b. MIT Press.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA, 2004. MIT Press.

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

A. Tewari and P.L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer, Berlin, 1982.

P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

E. Voorhees. Overview of the trect 2001 question answering track. In *TREC*, 2001.

A. Wayne. Inequalities and inversions of order. *Scripta Mathematica*, 12(2):164–169, 1946.

M. Weimer, A. Karatzoglou, Quoc V. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. *Mathematical Programming Study*, 3:143–175, 1975.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. 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*, pages 271–278. ACM, 2007.

A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15: 915–936, 2003.