IntervalRank — Isotonic Regression with Listwise and Pairwise Constraints

Taesup Moon, Alex Smola[*], Yi Chang, Zhaohui Zheng
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94051
{taesup, smola, yichang, zhaohui}@yahoo-inc.com

ABSTRACT

Ranking a set of retrieved documents according to their relevance to a given query has become a popular problem at the intersection of web search, machine learning, and information retrieval. Recent work on ranking focused on a number of different paradigms, namely, pointwise, pairwise, and list-wise approaches. Each of those paradigms focuses on a different aspect of the dataset while largely ignoring others. The current paper shows how a combination of them can lead to improved ranking performance and, moreover, how it can be implemented in log-linear time.

The basic idea of the algorithm is to use isotonic regression with adaptive bandwidth selection per relevance grade. This results in an implicitly-defined loss function which can be minimized efficiently by a subgradient descent procedure. Experimental results show that the resulting algorithm is competitive on both commercial search engine data and publicly available LETOR data sets.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; G.1.6 [**Numerical Analysis**]: Optimization - *convex programming*

General Terms

Algorithms, Experimentation, Theory

Keywords

learning to rank, isotonic regression, listwise constraints, pairwise constraints

[*]A.S. is also with the Australian National University, Research School of Information Sciences and Engineering

1. INTRODUCTION

Ranking a set of retrieved documents according to their relevance for a given query is a popular problem at the intersection of web search, machine learning, and information retrieval. Over the past decade, a large number of learning to rank algorithms have been proposed [12]. Based on how they treat sets of ratings, they can be effectively categorized into the following three groups: pointwise, pairwise, and listwise approaches.

For a given query and a set of retrieved documents, pointwise approaches try to directly estimate the relevance label for each query-document pair. While one may show that these approaches are consistent for a variety of performance measures [4], they ignore *relative* information within collections of documents. In other words, they ignore that a document with a mediocre rating may actually be desirable if all other documents carry even lower scores (i.e., the one-eyed may be king among the blind). Pairwise approaches, as proposed by [9, 10, 2, 24], take the relative nature of the scores into account by comparing pairs of documents. They ensure that we obtain the correct *order* of documents even in the case when we may not be able to obtain a good estimate of the ratings directly. Finally, listwise approaches, as proposed by [3, 20, 16, 22], treat the ranking in its totality, and they exploit the fact that we may not even care about the entire ranking, and that, furthermore, even among the retrieved subset of documents we may care considerably more about the topmost documents. For a more comprehensive references see [12].

There has been significant discussion about the relative merit of these strategies, and the common wisdom is that the listwise is better than the pairwise, which outperforms the pointwise. However, we argue that the listwise methods, when used naively, ignore some parts that the pointwise or pairwise algorithms are able to capture; pointwise approaches are able to capture the *absolute* relevance of a document for a query. Moreover, they are sensitive to the fact that queries with equal relevance should be rated equally (listwise approaches are typically indifferent of the order without a given relevance tier). Pairwise approaches capture some of these aspects, e.g., by weighting pairwise comparisons according to their score discrepancy.

In this paper, we present a novel algorithm — IntervalRank — which exploits all three of those aspects. That is, we use Isotonic Regression to deal with the listwise, translation-invariant aspects of the ranking problem. Secondly, we add a penalty to ensure that documents within the same grade be rated approximately equally. Finally, we impose large margin constraints between the grades which effectively ensure that different grades are well separated, and thus, ratings are obtained in a reliable fashion.

The idea of incorporating all three approaches to design a learning to rank algorithm has also been attempted in [21]. However, whereas their method is to sequentially vary the focus on the three approaches as the training process goes, we try to incorporate them all at once. Our IntervalRank achieves this by regressing on the pairwise ordinal intervals, which are optimized in a listwise fashion. Size constraints on the intervals ensure that like documents are rated alike. In a departure from previous methods, our loss function is only defined *implicitly* as the solution of a convex optimization problem. We show that this nontrivial variant of Isotonic Regression can be solved in loglinear time (in the number of relevant documents) even though we are dealing with a dense quadratic program. As we shall see, the key idea is to reformulate the intermediate quadratic program as a smaller program in terms of the boundary variables separating different grades and to use the log-barrier interior point algorithm for the reduced problem. The experimental results on both the commercial search engine data and publicly available LETOR data show that our algorithm is competitive with state-of-the-art algorithms.

The rest of this paper is organized as follows; in Section 2, we set notations necessary for the paper and present some preliminary backgrounds for our work. We derive our algorithm IntervalRank in Section 3 and show how we can implement it efficiently. Two experimental results are given in Section 4 to support the competitiveness of the IntervalRank. We conclude the paper in Section 5 with the key contributions of the paper.

2. BACKGROUND

2.1 Notation

We assume that we are in a classical learning-to-rank scenario. That is, at training time, we are given a set of queries $\mathcal{Q} = \{q_1, \ldots, q_Q\}$. For each query q_k, we are also given a set of retrieved documents $D_{q_k} = \{d_{k1}, \ldots, d_{kn_k}\}$ and associated relevance grades $G_{q_k} = \{g_{k1}, \ldots, g_{kn_k}\}$, where n_k is the number of documents for query q_k. The relevance grade g_{kj} is typically represented by numeric values in $\mathbb{R}$ indicating how relevant a document d_{kj} is to the given query q_k. For example, the usual editorial relevance judgments of $\{\text{Bad}, \text{Fair}, \text{Good}, \text{Excellent}, \text{Perfect}\}$ can be mapped into the grade values $\{0, 1, 2, 3, 4\}$, respectively. We will denote by $\mathcal{G}$ the set of possible relevance grades. With slight abuse of notation, we will drop the index k from queries q and associated documents d_{kj} whenever the index is obvious or irrelevant (i.e. q, d_j, G_q, D_q) to simplify notation.

For a given query q, we denote by

$$\mathcal{O}_q \triangleq \{(d_i, d_j) \in D_q \times D_q : g_i > g_j\} \text{ and}$$
$$\mathcal{T}_q \triangleq \{(d_i, d_j) \in D_q \times D_q : g_i = g_j\}$$

the set of ordered relevance pairs and tied relevance pairs, respectively, generated from the retrieved document set D_q. Moreover, we denote by

$$S_g \triangleq \{i : d_i \in D_q, g_i = g\} \qquad (1)$$

the set of indices of documents in D_q with grade g. We associate a numeric target $y_g \in \mathbb{R}$ with every $g \in \mathcal{G}$. For instance, we may set $y_0 = 0$, $y_1 = 0.5$, $y_2 = 3$, $y_3 = 7$, and $y_4 = 10$ for $\mathcal{G} = \{0, 1, 2, 3, 4\}$. Alternatively we might choose $y_g = 2^g - 1$ for $g \in \mathcal{G}$. Finally,

$$\Delta_{g+1,g} \triangleq y_{g+1} - y_g$$

represents the label margin between grades g and $g + 1$. Whenever clear from the context we denote by y_i the numeric target associated with document d_i and by Δ_{ij} the label margin between documents d_i and d_j.

We assume that each query-document pair (q_k, d_{kj}), where $d_{kj} \in D_{q_k}$, is represented as a feature vector $x_{kj} \in \mathbb{R}^p$. Again, when the context is clear, we will drop the index k for brevity. The ranking function is denoted by $f : \mathbb{R}^p \to \mathbb{R}$. At test time, the ranking function produces a set of scores for the corresponding retrieved documents, and the ranking of documents for a given query is determined by sorting the documents in terms of the scores. Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) are commonly used metrics that measure the quality of ranked list, of which definitions will be given in Section 4.

2.2 Inference

As common in the supervised learning, the estimation problem is often cast as one of the (regularized) risk minimization problem. That is, we assume that we have some loss function

$$l : \mathbb{R}^{2n} \to \mathbb{R}_0^+ \qquad (2)$$

which scores the performance of $f(x)$ relative to the numerical targets y via $l(\{f(x_1), \ldots, f(x_n)\}, \{y_1, \ldots, y_n\})$. For convenience, we use the shorthand

$$l(f, x_k, y_k) := l(\{f(x_{k1}), \ldots, f(x_{kn_k})\}, \{y_{k1}, \ldots, y_{kn_k}\})$$

to denote the loss function applied to the n_k query-document collections and the relevance targets for query q_k. Inference is then carried out by (regularized) empirical risk minimization, that is, by finding some f which minimizes

$$R[f] := \frac{1}{Q} \sum_{k=1}^{Q} l(f, x_k, y_k) + \lambda \Omega[f].$$

Here $\Omega[f]$ is an (optional) regularization term with regularization constant $\lambda \geq 0$, and the first term denotes the average performance of the ranking function f on the aforementioned training set. One may show that under fairly general conditions [19], minimization of a regularized risk term will produce a minimizer f which has good performance on the unseen data (in our case, queries and documents which one might expect to see in the deployed system).

Some of the key questions arising from the regularized risk minimization problem in the context of ranking are a) how to choose a suitable loss function l, b) how to find a flexible enough function class containing f, and c) how to solve the resulting optimization problem. It is the first of the three questions that this paper contributes to. For completeness, we discuss b) and c) briefly.

2.3 Functional gradient descent

The problem of minimizing $R[f]$ has been addressed in a number of different contexts. For instance, we can treat this as a convex minimization problem and use subgradient procedures [15, 5]. This works whenever the objective function itself is convex, and derivatives can be computed efficiently. Similar procedures can be applied to smooth proxies of the ranking performances [2].

For the purpose of this paper, we adopt a functional gradient descent representation [8, 14, 7]. This allows us to deal efficiently with cases where the function class $\mathcal{F}$ containing f is not explicitly available, but, rather, where we are able to draw functions from the class which are well correlated with the functional derivative of $R[f]$. This allows us to minimize $R[f]$ within function classes which are implicitly defined as convex combinations of some "weak learners" (i.e., the function classes defining the generators of the convex set), such as decision trees.

Algorithm 1 Functional gradient descent

```
Require: Training examples $\{(x_i, y_i)\}_{i=1}^n$
Ensure: $f^*$ that minimizes $R[f]$
  initialize $f = f_0 \in \mathcal{F}$
  repeat
    Compute functional gradient
        $g_i = \frac{\partial R[f(x_i)]}{\partial f(x_i)}, \quad i = 1, \ldots, n$
    Find $h \in \mathcal{F}$ via weak learner that is well correlated with $g \in \mathbb{R}^n$
    Update $f \leftarrow f - \nu h$
  until converged
```

Algorithm 1 describes how to obtain a functional gradient at every step. For notational brevity, we simplified the indices of the examples to $i = 1, \ldots, n$ above. Once we compute the functional gradient g, we subsequently obtain a regression tree h that is well correlated with g (i.e. we use the component-wise derivatives as regression targets). Finally, we update f by moving $\nu \in (0, 1)$ into the direction of the current gradient. More sophisticated variants of this procedure are possible. For instance, we may replace g by an update estimate computed by a (non)convex second order optimization procedure such as LBFGS [11].

3. INTERVALRANK

The IntervalRank defines an implicit loss function via a convex optimization problem and applies the functional gradient descent as described in Section 2.3. A nontrivial step of such formulation is to compute the functional gradient of the loss function. Section 3.1 describes how we define the loss function in detail and proves that the solution of the optimization problem is equivalent to the functional gradient of the loss function. Then, Section 3.2 $\sim$ Section 3.4 elaborate how we can solve the optimization problem efficiently by reducing the number of the optimization variables, efficiently evaluating the objective function of the optimization problem, and applying the logarithmic barrier interior point method. Finally, Section 3.5 shows that we can also easily add the pointwise loss function to our loss function.

3.1 Loss Functions

We begin our analysis by discussing isotonic regression as proposed by [25]. From now on, we will only consider a single query q since all the procedures can be easily parallelized. The key idea in isotonic regression ranking is to estimate the document grades up to a constant (latent variable) offset. This takes into account that it is the *relative* order of the documents that are being retrieved that matters for ranking performance. The corresponding loss function can be written as

$$l(f, x, y) := \min_{w \in \mathbb{R}} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i) + w)^2. \tag{3}$$

While this is desirable in its own right, it misses an important part of the ranking problem, namely that individual category labels should be well separated. This is achieved by defining

$$l(f, x, y) := \frac{1}{2} \|\delta^*\|_2^2 \tag{4}$$

where $\delta^* \in \mathbb{R}^n$ solves the optimization problem

$$\begin{aligned} &\underset{\delta \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2} \|\delta\|_2^2 \\ &\text{subject to } f(x_i) + \delta_i - f(x_j) - \delta_j \geq \Delta_{ij} \text{ for } (i, j) \in \mathcal{O}_q \end{aligned} \tag{5}$$

That is, the loss function (4) measures the minimal norm of the offset vector $\delta \in \mathbb{R}^n$ we need to apply to $f(x_1), \ldots, f(x_n)$ such that the list of shifted scores are consistent with the pairwise target label margins $\{\Delta_{ij} : (i, j) \in \mathcal{O}_q\}$. Note that this is a listwise optimization with pairwise constraints.

A naive approach to solve the optimization problem (5) would be at least cubic in terms of the number of variables used; the number of constraints encoded in O_q is typically quadratic in the number of documents n. Taking into account of the sparsity of the constraint matrix, an efficient brute force solution of the resulting quadratic program would require at least $O(n^3)$ operations to compute the reduced Karush-Kuhn-Tucker system [18], and thus, solving (5) has at least cubic cost. We shall see in Section 3.2 that efficient variable reduction and reordering can reduce this to $O(n \log n)$ cost.

Before we do so, let us extend the objective function in two ways. Firstly, the pairwise constraints ignore the fact that documents with equal relevance scores y_i should lead to ratings $f(x_i)$ which are also close. This can be addressed by a parameter-tying constraint on $\mathcal{T}_q$. Secondly, while a separation by Δ_{ij} is desirable, it may not always be entirely achievable. Hence, we relax the separation constraint by adding another slack variable. This leads to the following loss function: Define l as in (4) where $\delta^* \in \mathbb{R}^n$ solves

$$\begin{aligned} &\underset{\delta, \epsilon, \xi}{\text{minimize}} \ \frac{1}{2} \|\delta\|_2^2 + \frac{\lambda_1}{2} \|\epsilon\|_2^2 + \frac{\lambda_2}{2} \|\xi\|_2^2 \\ &\text{subject to } f(x_i) + \delta_i - f(x_j) - \delta_j \geq \Delta_{ij} - \epsilon_{g_i} \text{ for } (i, j) \in \mathcal{O}_q \\ &\qquad |f(x_i) + \delta_i - f(x_j) - \delta_j| \leq \xi_{g_i} \qquad \text{for } (i, j) \in \mathcal{T}_q \end{aligned} \tag{6}$$

Note that we only need to enforce adjacent constraints for intervals since they automatically enforce the larger-range constraints, hence the vector $\epsilon \in \mathbb{R}^{|\mathcal{G}|-1}$ rather than matrix-valued variable suffices. The relaxation ϵ takes care of overly optimistic separation requirements Δ_{ij}. Moreover, the addition of the constraint on $\mathcal{T}_q$ and the slack variables $\xi \in \mathbb{R}^{|\mathcal{G}|}$ ensures that ratings of the same score are tightly packed.

For the sake of using the functional gradient descent method as in Section 2.3, we need to compute the gradient of the loss function (4) with respect to f. In other words, we need to compute the gradient of the minimum of the objective function (6) with respect to f, which requires the following stability result from convex duality, e.g. [1, Section 5.6.3]:

Lemma 1 *Consider the convex minimization problem (F and c_i are convex functions)*

$$\underset{x}{\text{minimize}} \;\; F(x) \tag{7}$$
$$\text{subject to} \;\; c_i(x) \leq z_i \text{ for all } i \tag{8}$$

and let Λ_i be the Lagrange multipliers associated with c_i. Moreover, let $x^*(z), \Lambda^*(z)$ be the solution and Lagrange multipliers associated with z. Then, if strong duality holds and $F(x^*(z))$ is differentiable, we have $\partial_z F(x^*(z)) = -\Lambda^*(z)$.

Lemma 2 *The gradient of the loss satisfies $\partial_f l(f, x, y) = \delta$.*

Proof. We begin by writing out the Lagrange function of (6). In order to obtain linear constraints we replace the absolute value function by a pair of constraints, i.e. $|a| \leq b$ transforms into $a \leq b$ and $-a \leq b$. This yields

$$\begin{aligned} L(\delta, \epsilon, \xi, \Lambda) =& \frac{1}{2}\|\delta\|_2^2 + \frac{\lambda_1}{2}\|\epsilon\|_2^2 + \frac{\lambda_2}{2}\|\xi\|_2^2 \\ &- \sum_{i,j \in \mathcal{O}_q} \Lambda_{ij}\left[\Delta_{ij} - \epsilon_{g_i} - f(x_i) - \delta_i + f(x_j) + \delta_j\right] \\ &- \sum_{i,j \in \mathcal{T}_q} \Lambda_{ij}\left[f(x_i) + \delta_i - f(x_j) - \delta_j - \xi_{g_i}\right] \\ &- \sum_{i,j \in \mathcal{T}_q} \bar{\Lambda}_{ij}\left[-f(x_i) - \delta_i + f(x_j) + \delta_j - \xi_{g_i}\right] \end{aligned}$$

For a suitably chosen matrix A and an offset vector b the terms in Λ (and $\bar{\Lambda}$) can be subsumed by the expression $\Lambda^\top[b + A(f + \delta, \epsilon, \xi)]$ which leads to

$$L = \frac{1}{2}\|\delta\|_2^2 + \frac{\lambda_1}{2}\|\epsilon\|_2^2 + \frac{\lambda_2}{2}\|\xi\|_2^2 - \Lambda^\top[b + A(f + \delta, \epsilon, \xi)]$$

Now denote by A_f the upper slice of A pertaining to $f + \delta$. First order optimality conditions require that

$$\partial_\delta L = \delta - A_f^\top \Lambda = 0 \text{ and hence } \delta = A_f^\top \Lambda.$$

If we were to change f by an infinitesimal amount df, the constraints of the optimization problem would change by $A_f df$. Since the objective function is quadratic and strongly convex and all constraints are linear we may apply Lemma 1. It states that the change in the objective function is given by $\Lambda^\top A_f df$ and therefore by $\delta^\top df$. This proves the claim that δ is the variational derivative with respect to f. $\square$

This result is surprising since it implies that in the first order, changes in f are directly reflected in δ. It also means that computing gradients is as simple as solving the optimization problem itself. Note that for more general convex penalties $\Omega[\delta]$ instead of $\frac{1}{2}\|\delta\|^2$, the connection is somewhat more complex. The condition $\delta = A_f^\top \Lambda$ is replaced by

$$\partial_\delta \Omega[\delta] = A_f^\top \Lambda.$$

This leads to the gradient $\partial_\delta \Omega[\delta]$. We believe that implicitly defined loss functions constitute a fertile field of research, as long as they are computationally accessible for estimation.

3.2 Variable Reduction

The key trick to an efficient solution of the Quadratic Program (6) is to eliminate all but the boundary variables between different grades such that, regardless of the size of the initial optimization problem, we have a smaller subproblem which only scales with the number of different grades. Without loss of generality, we assume that there exists at least one document for all the relevance grades in the training data. If this were not the case, we could simply drop the corresponding value y_i from the problem and obtain an identical problem with a smaller number of relevance grades.

The key observation when solving (6) is that we may rewrite the constraints such that the boundaries between different grades are explicitly specified.

Lemma 3 *For fixed slack variables ϵ and ξ, let $P_1(\epsilon, \xi)$ and $P_2(\epsilon, \xi)$ be polyhedra, where P_1 and P_2 are defined via*

$$\begin{aligned} P_1(\epsilon, \xi) = \Big\{ z \in \mathbb{R}^n : & \; z_i - z_j \geq \Delta_{ij} - \epsilon_{g_i} \text{ for all } (i,j) \in \mathcal{O}_q, \\ & |z_i - z_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \mathcal{T}_q \Big\} \end{aligned}$$

$$\begin{aligned} P_2(\epsilon, \xi) = \Big\{ & z \in \mathbb{R}^n, l \in \mathbb{R}^{|\mathcal{G}|}, u \in \mathbb{R}^{|\mathcal{G}|} : \\ & z_i \in [l_{g_i}, u_{g_i}] \text{ for all } 1 \leq i \leq n, \\ & l_g \leq u_g \leq l_g + \xi_g \text{ for all } g \in \mathcal{G}, \\ & l_{g+1} - u_g \geq \Delta_{g+1,g} - \epsilon_g \text{ for all } g \in \mathcal{G}\backslash\{g_{\max}\} \Big\}, \end{aligned}$$

where $g_{\max} \triangleq \max\{g : g \in \mathcal{G}\}$. Then the projection of P_2 onto its first n coordinates is equivalent to P_1.

Proof. For any $z \in P_1$ we define $l_g := \min_{i:y_i=g} z_i$ and $u_g := \max_{i:y_i=g} z_i$. Since, in particular, the extremes satisfy the conditions of P_1, hence they also satisfy the conditions of P_2. Likewise, for any $(z, l, u) \in P_2$ the conditions on z are equivalent to those in P_1. The projection preserves this property. $\square$

Lemma 4 *The optimization problems arising by replacing the constraints P_1 in (6) by P_2 are equivalent.*

Proof. The objective functions match, and the projection of P_2 onto $\mathbb{R}^n$ equals P_1 if we define $z_i = f(x_i) - \delta_i$. $\square$

The reason for introducing P_2 is that we may now eliminate δ entirely from the optimization problem since we are able to express the problem in terms of the boundaries between different grades. First, recall (1) that S_g stands for the set of indices of all documents with grade g. Also, from now on, let $f_i \triangleq f(x_i)$. Then, we have following theorem.

Theorem 5 *The optimization problem (6) is equivalent to the following problem:*

$$\begin{aligned} \underset{l,u,\epsilon,\xi}{\text{minimize}} \;\; & \frac{1}{2}\sum_{g \in \mathcal{G}} \sum_{i \in S_g} \left[(l_g - f_i)_+^2 + (f_i - u_g)_+^2\right] \\ & + \frac{\lambda_1}{2}\|\epsilon\|_2^2 + \frac{\lambda_2}{2}\|\xi\|_2^2 \\ \text{subject to} \;\; & l_g \leq u_g \leq l_g + \xi_g, \; \forall g \in \mathcal{G} \\ & l_{g+1} - u_g \geq \Delta_{g+1,g} - \epsilon_g, \; \forall g \in \mathcal{G}\backslash\{g_{\max}\} \end{aligned}$$

where $(\eta)_+ \triangleq \max(0, \eta)$, and $l \in \mathbb{R}^{|\mathcal{G}|}$, $u \in \mathbb{R}^{|\mathcal{G}|}$. Moreover, $\delta \in \mathbb{R}^n$ is given by $\delta_i = f_i - \max(l_{g_i}, \min(u_{g_i}, f_i))$.

PROOF. Since on $P_2(\epsilon, \xi)$ the constraints in terms of $\delta_i = f_i - z_i$ decouple, we can see that the optimization problems in delta are solved by $\delta_i = f_i - \max(l, \min(u, f_i))$. Plugging this value into the objective function yields $(l_{g_i} - f_i)_+^2 + (f_i - u_{g_i})_+^2$, which proves the theorem. □

The benefit of Theorem 5 is that we now have a reduced optimization problem in $4|\mathcal{G}| - 1$ variables with simple neighboring constraints (between adjacent grades) rather than $n + 2|\mathcal{G}| - 1$ variables and considerably more complicated constraints.

3.3 Computing the objective function

To solve the optimization problem of Theorem 5, e.g., by means of a logarithmic barrier function, we need to be able to compute values and gradients of the objective function cheaply. A naive implementation would require $O(n)$ operations, since we need to carry out a sum over $O(n)$ terms. A more efficient strategy is to sort the values $\{f_i : i \in S_g\}$ for each $g \in \mathcal{G}$ once (this costs $O(n \log n)$ operations in QuickSort), and then, simply perform a lookup to decide for which set of f_i the terms $(l_g - f_i)_+^2$ or $(f_i - u_g)_+^2$ are nonzero. It works as follows:

Assume that the documents $\{f_i : i \in S_g\}$ with grade g are sorted in the ascending order. Then, define the linear and quadratic partial sums for grade g as

$$L_{gi} := \sum_{j=1}^{i} f_j \qquad \text{and} \qquad Q_{gi} := \sum_{j=1}^{i} f_j^2 \tag{9}$$

for $i = 1, \ldots, |S_g|$, which can be computed in $O(n)$ time and $O(n)$ space. Denote by $a_g \triangleq \max\{i : f_i \le l_g\}$ and $b_g \triangleq \min\{i : f_i \ge u_g\}$ the indices of the largest and smallest elements f_i exceeding l_g and u_g, respectively. Then, for grade g, we can write

$$\begin{aligned}
&\frac{1}{2}\sum_{i \in S_g} (l_g - f_i)_+^2 \\
=&\frac{1}{2}\sum_{i \in S_g} (l_g^2 + f_i^2 - 2 l_g f_i)\mathbf{1}\{f_i \le l_g\} \qquad (10)\\
=&\frac{1}{2} a_g l_g^2 + \frac{1}{2} Q_{g,a_g} - l_g L_{g,a_g},
\end{aligned}$$

where $\mathbf{1}\{f_i \le l_g\}$ in (10) is 1 if $f_i \le l_g$, and 0 otherwise. A similar partial sum can be computed with regard to the upper boundary. This sum now only contains as many terms as we have different grades. An initial lookup to construct a_g and b_g costs at most $\log n$ time. Subsequent lookups are likely to be $O(1)$ since at every optimization step we will adjust l and u only slightly, so it is unlikely that the indices will change by more than one or two at a time. In the same fashion we can compute gradients with respect to l and u.

This means that up to a negligible $O(n \log n)$ expense to sort the values f_i initially, all other operations are $O(\log n)$ or $O(1)$ respectively, thus greatly reducing the computational cost of the optimization procedure making it virtually independent of the number of documents involved. Such an algorithm makes the implementation of loss functions such as those defined via (6) possible in the first place: this is the reason why [25] only consider the Isotonic Regression problem without the constraint $|f(x_i) + \delta_i - f(x_j) - \delta_j| \le \xi_{g_i}$ for $(i, j) \in \mathcal{T}_q$ since this reduced optimization problem can be solved more efficiently, albeit with reduced ranking performance.

3.4 Logarithmic Barrier

For completeness we briefly describe the basic log-barrier template for constrained convex optimization:

Algorithm 2 Logarithmic Barrier

```
initialize feasible parameters $l, u, \epsilon, \xi$
for $\mu = 1$ step: $\mu \leftarrow 2\mu$ end: $\mu > 100$ do
  Minimize objective function with constraints added via
  $\frac{1}{2}\sum_{g \in \mathcal{G}}\sum_{i \in S_g}\left[(l_g - f_i)_+^2 + (f_i - u_g)_+^2\right] + \frac{\lambda_1}{2}\|\epsilon\|_2^2 + \frac{\lambda_2}{2}\|\xi\|_2^2$
  $- \mu^{-1}\sum_{g \in \mathcal{G}} \log(u_g - l_g) + \log(l_g + \xi_g - u_g)$
  $- \mu^{-1}\sum_{g \in \mathcal{G}\setminus\{g_{\max}\}} \log(l_{g+1} - u_g - \Delta_{g+1,g} - \epsilon_g)$
end for
```

The inner loop of the optimization algorithm proceeds by using conjugate gradient descent with line search which is run until approximate convergence is achieved (we do not need to run it to full convergence since we keep on adjusting μ during the process of the optimization procedure). Note that there is no need to optimize the problem to high precision (i.e. very large μ) — a smaller μ imposes an additional benefit for having even larger separation between the grades than required by Δ_{ij} and ϵ.

3.5 Adding a Pointwise Loss

Besides the pairwise and effectively listwise constraints on the scores it is easy to add pointwise regression loss to the objective function. This ensures that we obtain a calibration that is correct in absolute terms. Since gradients are additive, it is straightforward to add this to the listwise loss function as it stands. Hence we have

$$l(f, x, y) = l_{\text{listwise}}(f, x, y) + \frac{\lambda_3}{2}\|y - f(x)\|_2^2 \tag{11}$$

While the changes effected by the pointwise loss are not massive (it constitutes a consistent loss function in its own right, though), adding a score calibration exploits information that is not exploited by the listwise loss functions discussed in Section 3.1.

4. EXPERIMENTAL EVALUATION

We established in the previous section that a combined point and listwise loss function which ensures a large margin between ratings can be implemented efficiently. We now demonstrate that our loss function works well in practice, outperforming the state of the art on a number of datasets both public (Letor 3.0 OHSUMED) and proprietary (commercial search engine).

Two types of experiments are required to corroborate our claims: firstly, we show that the combined loss function composed of a listwise and a pointwise score outperforms the listwise-only loss function. Secondly, we show that our loss function outperforms competing approaches.

4.1 Performance Metrics

For comparison purposes we use a number of performance metrics — the Discounted Cumulative Gain@k (NDCG@k), the Precision@k (P@k), and the Mean Average Precision

(MAP). These three metrics tend to provide a rich representation of what matters in *editorially annotated* ranking problems. For conciseness we give a brief definition of the metrics below. We use r to denote the ranks of the documents after the scoring function f has been applied. That is, r_i is the rank of document i.

NDCG@k The NDCG at position k (NDCG@k) for a ranked document list of a query q is defined as a position and rating weighted score which is then normalized such that the maximum NDCG score is 1 for a perfect ranking. We have the definition

$$\mathrm{NDCG@k}[r] := Z \sum_{i=1}^{k} \frac{2^{g_{r_i}} - 1}{\log(1+i)}. \qquad (12)$$

Here g_{r_i} is the relevance grade of document ranked at i. Denote by $\hat{r} := \mathrm{argsort}[g]$ the optimally sorted version of the document collection. In this case we can write $Z = \sum_{i=1}^{k} \frac{2^{g_{\hat{r}_i}} - 1}{\log(1+i)}$. The motivation for truncating the sum at k is that in a search engine we are only interested in the top-k results of a query rather than a sorted order of the entire document collection.

P@k The Precision at position k for a ranked document list of a query q is defined as

$$\mathrm{P@k}[r] := \frac{1}{k} \sum_{i=1}^{k} I\left\{g_{r_i} = \max(g)\right\}. \qquad (13)$$

Here $I\{\mathrm{expr}\}$ is the indicator function that assumes the value 1 if expr = TRUE and 0 otherwise. Hence, P@k only considers the top-ranking documents as relevant and computes the fraction of such documents in the top-k elements of the ranked list.

MAP The mean of the Average Precision over test queries is defined as the mean over the precision scores for all retrieved relevant documents. It is given by

$$\mathrm{MAP} = \frac{\sum_{k=1}^{n} \mathrm{P@k} \times I\left\{g_{r_k} = \max(g)\right\}}{\sum_{k=1}^{n} I\left\{g_{r_k} = \max(g)\right\}}. \qquad (14)$$

As before, n is the number of documents associated with query q. On the OHSUMED dataset, $\max(g) = 2$ (we only have 3 different grades).

4.2 OHSUMED

Data.

The OHSUMED data set is one of the data sets contained in the LETOR 3.0 package [13]. It is widely used in information retrieval to evaluate the performance of various learning to rank algorithms. OHSUMED contains queries, the contents of the retrieved documents, and the relevance judgments of the document to the associated queries.

A number of features have been added including BM25, HostRank and topical PageRank. In addition, the LETOR 3.0 package contains the results of several learning to rank algorithms such as RankBoost [6], RankSVM [10], AdaRank [23], FRank [17], and ListNet [3] as baselines.

OHSUMED contains medical publication abstracts. There are 106 queries and a total of 16,140 query-document pairs with associated relevance judgements. Each query-document pair is represented by a 25-dimensional feature vector. The relevance grade judgments are given in three levels, i.e., $\mathcal{G} = \{0, 1, 2\}$, where 0 means *not relevant*, 1 means *possibly relevant*, and 2 means *definitely relevant*. Among the query-document pairs, 11,303 are judged as $g = 0$, 2,585 are judged as $g = 1$, and $2,252$ are judged as $g = 2$ (see Figure 1).

LETOR 3.0 also contains TREC and GOV. Unfortunately those datasets are not well suited to ranking since they only contain binary ratings (relevant, irrelevant), which reduces the ranking problem to binary classification.

Results.

In our experiments we used 5-fold crossvalidation where $\frac{1}{5}$ each were used for validation and testing and $\frac{3}{5}$ for training. Appropriately all models were trained using the training set, tuned on the validation set, and tested on the test set.

We used 125 trees, i.e. we performed 125 steps of the functional gradient descent method (Algorithm 1). The parameters for IntervalRank, namely the number of nodes in each tree and the shrinkage step size, the weight parameters λ_1 and λ_2 for the slack variables in our optimization problem, and the steplength of the functional gradient descent procedure ν were adjusted on the validation set.

Table 1 contains a summary of the results. IntervalRank performs very well on NDCG@1-3, and P@1-2 compared to the reference algorithms. Note that the improvements are significant on those metrics. However, we also see that for the bottom positions of the ranked list, IntervalRank's benefits are less significant. Overall, the experiments show that IntervalRank is state of the art and a useful addition to the machine learning ranking toolbox. Note that on datasets with only three grades some of the more detailed distinctions between elementwise, pairwise and listwise ranking are less pronounced.

4.3 Commercial search engine

Data.

The commercial search engine data is collected by sampling queries from the query logs and by manually labeling a number of associated documents with human understandable grades. In our collection we have five relevance grades, i.e., $\mathcal{G} = \{\mathrm{Bad}, \mathrm{Fair}, \mathrm{Good}, \mathrm{Excellent}, \mathrm{Perfect}\}$. We collected total 8,180 queries with 341,300 query-document pairs for the training set and 916 queries with 32,008 query-document pairs. The distribution of the relevance grades for each set are given in Table 2.

The feature vector x_i for each query-document pair (q, d_i) consists of the following three categories:

Query features They only depend on the query q and have constant values across D_q. For example, the number of terms in the query, language of the query, or the query classification result fall into this category.

Document features They only depend on the document and do not vary over queries. The number of inbound links pointing to the document or the spam score of the document are such features.

Query-document features They explicitly depend on both the query and the document. This includes for instance the number of times query terms occur in the document or in the anchor text.

Algorithms	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
RankBoost	0.4632	0.4504	0.4555	0.4543	0.4494	0.5576	0.5481	0.5609	0.5580	0.5447	0.4411
RankSVM	0.4958	0.4331	0.4207	0.4240	0.4164	0.5974	0.5494	0.5427	0.5443	0.5319	0.4334
FRank	0.5300	0.5008	0.4812	0.4694	0.4588	0.6429	0.6195	0.5925	0.5840	0.5638	0.4439
ListNet	0.5326	0.4810	0.4732	0.4561	0.4432	0.6524	0.6093	0.6016	0.5745	0.5502	0.4457
AdaRank.MAP	0.5388	0.4789	0.4682	0.4721	0.4613	0.6338	0.5959	0.5895	0.5887	0.5674	0.4487
AdaRank.NDCG	0.5330	0.4922	0.4790	0.4688	0.4673	0.6719	0.6236	0.5984	0.5838	0.5767	0.4498
IntervalRank	**0.5628**	**0.5448**	**0.4900**	0.4703	0.4609	**0.6892**	**0.6522**	0.5768	0.5556	0.5488	0.4466

Table 1: Performance on the OHSUMED dataset (we use N@k to denote NDCG@k). Note that IntervalRank outperforms other ranking algorithms in the top NDCG categories and that it is very competitive otherwise.

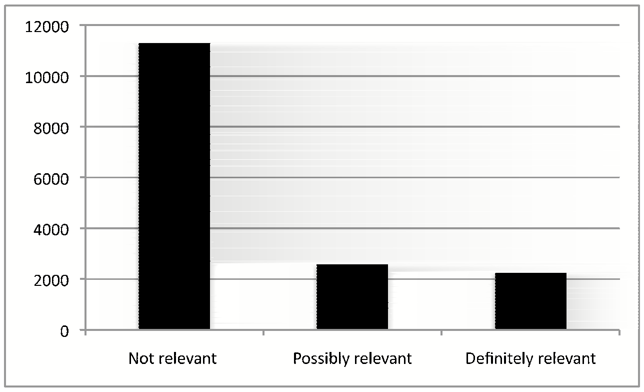


Figure 1: Grade distribution on OHSUMED

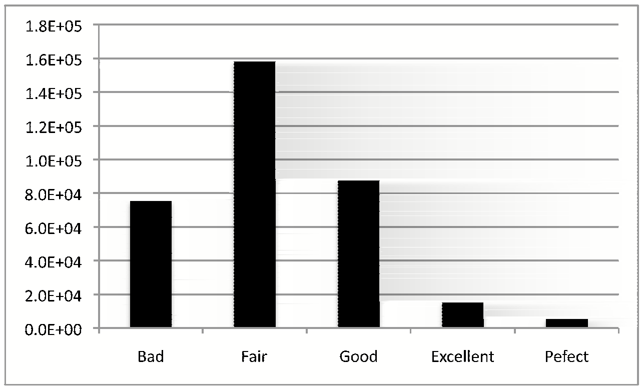


Figure 2: Grade distribution on search engine data

Comparison to other algorithms.

We compared IntervalRank with three other algorithms that we have implemented: the regression based pointwise algorithm (GBDT) [4], the pairwise algorithm using squared hinge loss (GBRank) [26], and the listwise algorithm using a probabilistic approach (ListMLE) [22]. All of these three algorithms are implemented within the functional gradient descent framework with decision trees as weak learners. The various algorithms differ only in the choice of the loss function.

GBDT: This is simply a squared loss which weighs the deviation between target score y_i and estimate $f(x_i)$, that is

$$l(f, x, y) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

GBRank: It uses a loss function quite related to IntervalRank. The main difference is that it tries to regress explicitly on the large margin interval boundaries Δ_{ij} and moreover, that it aims for vanishing pairwise difference between identical grades.

$$\begin{aligned} l(f, x, y) = & \sum_{(i,j) \in \mathcal{O}_q} (\Delta_{ij} - f(x_i) + f(x_j))_+^2 \\ & + \lambda_1 \sum_{(i,j) \in \mathcal{T}_q} (f(x_i) - f(x_j))^2 \end{aligned}$$

Here λ_1 is a user-defined parameter calibrating the trade-off between the monotonicity and the clustering constraints.
ListMLE: It uses a logistic regression model rather than a hinge loss in the ordinal regression setting of [9].

$$l(f, x, y) = -\sum_{i=1}^{n} \log(1 + \sum_{j>i} \exp(f(x_{r_j}) - f(x_{r_i})))$$

Here r_i is the original index of the document that should be ranked at i-th position.

Results.

We used 600 trees for all four loss functions and we compared NDCG@1 and NDCG@5 performance for evaluation purposes. Besides directly computing the NDCG values, we make one practical consideration: we used only the two most relevant documents per host. For real world search engines this is a very reasonable restriction, since retrieval of multiple similar documents from the same host may deteriorate the quality of the result set considerably. Therefore, in our test result, only two documents from the same hosts that have the highest ranking scores were allowed to be listed in the retrieved result, and then the NDCG values were computed from that list (this restriction was applied to all four algorithms).

To test whether an additive pointwise score is needed for good performance we report results for GBRank and IntervalRank with and without the regression loss

$$l(f, x, y) = \frac{\lambda_3}{2} \|y - f(x)\|_2^2.$$

For ListMLE this is added by default since it produces rather poor performance without it. We denote the regression-calibrated variants by GBRank$_{\text{Reg}}$ and IntervalRank$_{\text{Reg}}$.

Figure 3 shows the (host-name limited) performance with regard to the NDCG@1 and NDCG@5 scores as a function of the number of decision trees. Multiple configurations of parameters were used for each algorithm in training, and the results show the performances of the best configurations on the test set. As before IntervalRank outperforms other algorithms in both both metrics. However, we again observe larger improvements in NDCG@1 than in NDCG@5, which suggests that IntervalRank performs well on the top portion of the list.

A remark regarding the magnitude of the improvement is in order: a change of 1% may appear small, however it is significant for web search in a commercial setting. Also note that the difference between the competing algorithms which represent about 5 years of progress in learning to rank is only approximately 4%.

5. CONCLUSION

In this paper we presented a novel combined loss function which outperforms related ranking functions on both commercial and publicly available datasets. It relies on a number of key ideas:

- An adaptive listwise approach which incorporates tied pairwise constraints.
- An efficient $O(n \log n)$ algorithm for solving the resulting optimization problem.
- A duality result which allows us to compute the loss gradient efficiently.
- The combination of listwise and pointwise losses which ensures that we capture all relevant pieces of information inherent in an editorially annotated dataset.

We believe that each of those four pieces is useful in its own right to design improved ranking functions and their application extends beyond machine learning ranking to collaborative filtering [20] and similar preference-related problems.

Acknowledgments.

We thank Mingrui Wu and Hongyuan Zha for helpful suggestions and discussions. We also thank Shihao Ji for his help in experiments.

6. REFERENCES

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, England, 2004.

[2] 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.

[3] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[4] D. Cossock and T. Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

[5] C. Do, Q. Le, and C. Foo. Proximal regularization for online and batch learning. In *International Conference on Machine Learning ICML*, 2009.

[6] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[7] J. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Stanford University, 1999.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[9] 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.

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

[11] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[12] T. Y. Liu. *Learning to Rank for Information Retrieval.* Now Publishers, 2009.

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

[14] 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.

[15] A. Smola, S. V. N. Vishwanathan, and Q. Le. Bundle methods for machine learning. In D. Koller and Y. Singer, editors, *Advances in Neural Information Processing Systems 20*, Cambridge MA, 2007. MIT Press.

[16] M. Taylor, J. Guiver, S. Robertson, and T. Minka. SoftRank: Optimising non-smooth rank metrics. In *Proceedings of International ACM Conference on Web Search and Data Mining*, 2008.

[17] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. FRank: A ranking method with fideltiy loss. In *Proceedings of International ACM SIGIR Conference on Research and development in information retrieval*, 2007.

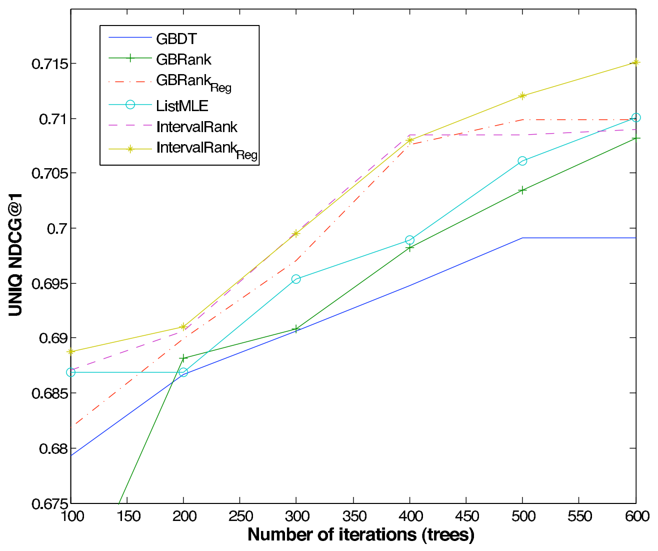
[18] R. J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer Academic, Hingham, 1997.

[19] V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

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

[21] M. Wu, H. Zha, Z. Zheng, and Y. Chang. Smoothing DCG for learning to rank: A novel approach using smoothed hinge functions. In *Proceedings of CIKM (Short Paper)*, 2009.

[22] F. Xia, T. Y. Liu, J. Wang, W. Zhang, and H. Li.



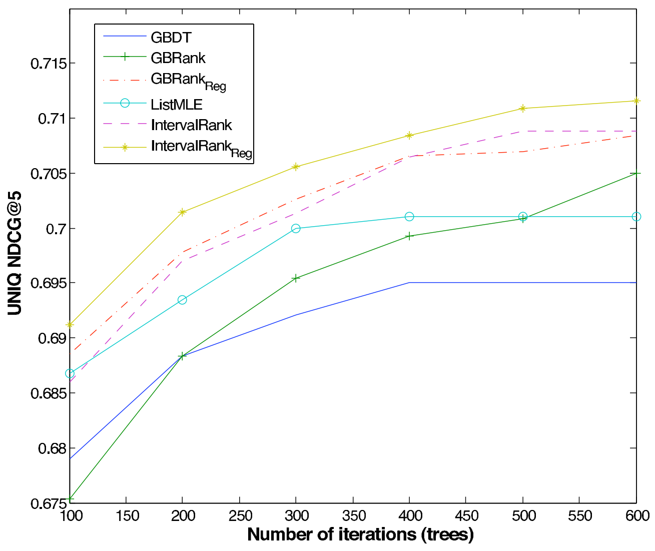


Figure 3: The regression calibrated version of IntervalRank outperforms all other ranking losses both for NDCG@1 and NDCG@5 for any number of trees. Also note that adding a pointwise loss to the pairwise and listwise approaches always improved performance.

Listwise approach to learning to rank - Theory and algorithm. In *International Conference on Machine Learning (ICML)*, 2008.

[23] J. Xu and L. Hang. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, 2007. ACM Press.

[24] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgments. In *Proccedings of Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

[25] Z. Zheng, H. Zha, and G. Sun. Query-level learning to rank using isotonic regression. In *Proccedings of the 46th Annual Allerton Conference on Communication, Control and Computing.* Allerton, IL, 2008.

[26] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, Cambridge, MA, 2008.