Sparse Learning-to-Rank via an Efficient Primal-Dual Algorithm

Hanjiang Lai, Yan Pan, Cong Liu, Liang Lin, and Jie Wu, *Fellow*, *IEEE*

Abstract—*Learning-to-rank* for information retrieval has gained increasing interest in recent years. Inspired by the success of sparse models, we consider the problem of sparse learning-to-rank, where the learned ranking models are constrained to be with only a few nonzero coefficients. We begin by formulating the sparse learning-to-rank problem as a convex optimization problem with a sparse-inducing ℓ_1 constraint. Since the ℓ_1 constraint is nondifferentiable, the critical issue arising here is how to efficiently solve the optimization problem. To address this issue, we propose a learning algorithm from the primal dual perspective. Furthermore, we prove that, after at most $O(\frac{1}{\epsilon})$ iterations, the proposed algorithm can guarantee the obtainment of an ϵ-accurate solution. This convergence rate is better than that of the popular subgradient descent algorithm. i.e., $O(\frac{1}{\epsilon^2})$. Empirical evaluation on several public benchmark data sets demonstrates the effectiveness of the proposed algorithm: 1) Compared to the methods that learn dense models, learning a ranking model with sparsity constraints significantly improves the ranking accuracies. 2) Compared to other methods for sparse learning-to-rank, the proposed algorithm tends to obtain sparser models and has superior performance gain on both ranking accuracies and training time. 3) Compared to several state-of-the-art algorithms, the ranking accuracies of the proposed algorithm are very competitive and stable.

Index Terms—Learning-to-rank, sparse models, ranking algorithm, Fenchel duality

1 Introduction

Ranking is a crucial task for information retrieval systems, in particular for web search engines. Learning-to-rank is a task that applies machine learning techniques to learn good ranking predictors for sorting a set of entities/documents. It has been drawing increasing interest in information retrieval and machine learning research. Many learning-to-rank algorithms have been proposed in literature such as [3], [4], [5], [6], [7], [8], [9].

In many machine learning applications, such as computer vision and bioinformatics, there is much desire to learn a sparse model. That is, a model with only a few nonzero coefficients with respect to the input features. Models with sparsity constraints are also desirable in ranking. First, some new data sets for ranking, such as the data sets for Yahoo!'s Learning-to-Rank Challenge[1] and Microsoft's data sets for large-scale learning-to-rank,[2] contain high-dimensional features. High-dimensional features lead to the problem that the dense models learned are complicated and hard to interpret. Second, high-dimensional features may be redundant or noisy, which results in poor generalization performance. Lastly, a sparse model has less computational cost in prediction.

The following is an intuitive example to illustrate why sparse learning works: a few strong features can dominate the whole ranking performance. We sort the documents in the TD2004 data set (in LETOR 3.0 [20]), respectively, using each individual feature out of the 64 given features. The results of *Normalized Discounted Cumulative Gain* (NDCG)@10 evaluation metrics (interested readers please refer to Section 8.2 for more details about NDCG) are shown in Fig. 1. For comparison, we generate a random predictor that uses all of the features as follows: we randomly initialize a feature weight vector $\boldsymbol{w}(\boldsymbol{w} = (w_1, w_2, \ldots, w_{64}), \forall i, w_i \geq 0, \sum_{i=1}^{64} w_i = 1)$ for 10 times and sort the documents in decreasing order of the results of the inner product $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$, where $\boldsymbol{x}$ denotes the feature vector of a document, and then we get the average over the 10 NDCG@10 values. We can observe in Fig. 1 that only several strong features, i.e., BM25, language models, PageRank, HITS [20], have obviously higher NDCG@10 values than the average of ten random sorting values (the red line), while many are lower. Moreover, there are some poor features whose NDCG@10 values are very far away from the average value of random sorting. To sum up, there can be cases where the whole ranking performance is dominated by only a small number of strong features, where learning a sparse model with only a few nonzero coefficients is desirable for ranking, and it has the potential to achieve better performance.

To obtain sparse ranking models, a natural way is to construct a ranking model with the smallest number of features. This problem is usually modeled using the ℓ_0 penalty. However, the resulting optimization problem

1. http://learningtorankchallenge.yahoo.com/.
2. http://research.microsoft.com/en-us/projects/mslr/.

- *H. Lai and C. Liu are with the School of Information Science and Technology, Sun Yat-sen University, NO. 132 East Waihuan Road, Guangzhou Higher Education Mega Center, Guangzhou 510006, P.R. China. E-mail: laihanj@student.sysu.edu.cn, gzcong@gmail.com.*
- *Y. Pan and L. Lin are with the School of Software, Sun Yat-sen University, NO. 132 East Waihuan Road, Guangzhou Higher Education Mega Center, Guangzhou 510006, P.R. China. E-mail: panyan5@mail.sysu.edu.cn, linliang@ieee.org.*
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, 1805 N. Broad ST., Room 302, Wachman Hall 302, Philadelphia, PA 19122. E-mail: jiewu@temple DOT edu.*

Manuscript received 15 Aug. 2011; revised 11 Jan. 2012; accepted 21 Feb. 2012; published online 28 Feb. 2012.
Recommended for acceptance by M. Guo.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-08-0545.
Digital Object Identifier no. 10.1109/TC.2012.62.

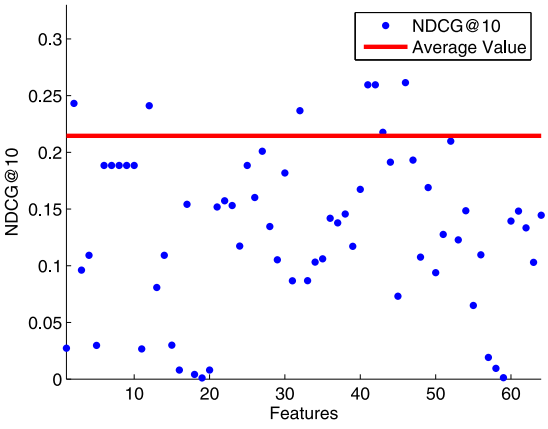


Fig. 1. The red line denotes the average NDCG@10 value over 10 times of random sorting. The dots are the NDCG@10 values of 64 features, respectively. We can see that only a few dots are above the red line.

is an NP-hard combinatorial problem. A simple way to tackle the ℓ_0 penalty is feature selection, which greedily chooses an additional feature at every step to reduce the training error. Although feature selection algorithms directly address the ℓ_0 penalty, they are nonconvex and hard to analyze. A more popular way is to use the ℓ_1 penalty as a convex surrogate of the ℓ_0 penalty. The convex optimization problems with the ℓ_1 penalty not only lead to efficient learning algorithms but also allow comprehensive theoretical analysis. In this paper, we focus on the sparse ranking problem with the ℓ_1 penalty.

However, no effort has been made to tackle the problem of learning a sparse model for ranking with the ℓ_1 constraint except for the work [10], in which the authors proposed a reduction framework to reduce ranking to the problem of importance-weighted pairwise classification. Then, they used an ℓ_1 regularized algorithm to learn a sparse ranking predictor. Despite the improvement they reported, the authors did not justify the individual contribution of the two parts of their solution: the reduction framework and the sparse learning algorithm.

This paper aims to answer the question of how the sparse learning algorithm contributes to improve ranking accuracy. We propose a convergence-provable primal-dual algorithm that optimizes the ℓ_1 regularized pairwise ranking loss. Furthermore, we empirically show in our experiments that our algorithm, using the pairwise ranking loss with sparse-inducing ℓ_1 norm, can significantly outperform the algorithm using the same loss with ℓ_2 norm and can achieve a state-of-the-art performance on several benchmark data sets.

Our contributions in this paper can be summarized as follows: 1) We successfully formulate the sparse learning-to-rank problem as a convex optimization problem by combining a pairwise ranking loss and a sparse-inducing ℓ_1 norm. Since the ℓ_1 norm is nondifferentiable, this optimization problem is difficult to solve. We propose a learning algorithm for this optimization problem from the primal-dual perspective. 2) We prove that, after T iterations, our proposed algorithm can guarantee the obtainment of a solution with desired tolerant optimization error $\epsilon = O(\frac{1}{T})$. Our algorithm has a better convergence rate than the subgradient descent algorithm, which is a popular algorithm for convex and nondifferentiable problems with a convergence rate of $O(\frac{1}{\sqrt{T}})$ [11]. 3) We empirically show that, compared to the methods that learn dense models, learning a ranking model with sparsity constraints can significantly improve the ranking accuracies. Experiment results show that our learning algorithm achieves a state-of-the-art performance on several public benchmark data sets.

2 Related Work

Recently, there has been a lot of research focusing on learning-to-rank in the machine learning community. Two main classes of methods for learning-to-rank have been explored in the last few years: pairwise methods and listwise methods [3], [4], [5], [6], [8], [12]. Our proposed method belongs to the first class.

The first class of methods is based on the so-called pairwise approach, in which a process of learning-to-rank is viewed as a task to classify the preference order within document pairs. Ranking SVM [6], RankBoost [5], and RankNet [3], are notable pairwise algorithms. The Ranking SVM algorithm adopts a large margin optimization approach like the traditional SVM [15]. It minimizes the number of incorrectly ordered instance pairs. Several extensions of Ranking SVM have also been proposed to enhance the ranking performance, such as [13], [14]. RankBoost is a boosting algorithm for ranking by using pairwise preference data. RankNet is another well-known algorithm, which applies Neural Network to rank and uses cross entropy as its loss function. Recently, Chapelle and Keerthi [16] replaced the standard hinge loss in Ranking SVM with a differentiable squared hinge loss and, thus, proposed a Newton descent algorithm to efficiently learn the ranking predictor.

The second class contains the listwise methods in which there are mainly two streams:

1. The first stream optimizes a loss function directly based on the IR evaluation metrics. SVM-MAP [8] adopts the structural support vector machines to minimize a loss function that is the upper bound of the *Mean Average Precision* (MAP) evaluation metrics (interested readers please refer to Section 8.2 for more details about MAP). AdaRank [12] is a boosting algorithm that optimizes an exponential loss, which upper bounds the metrics of MAP and NDCG.
2. The second stream defines several listwise loss functions, which take the list of retrieved documents for the same query as a sample. ListNet [4] defines a loss function based on the KL-divergence between two permutation probability distributions. ListMLE [18] defines another listwise likelihood loss function based on the Luce Model [17].

Another aspect related to the work in this paper is sparse learning, which has been widely applied to many applications in computer vision, signal processing, and bioinformatics. Many learning algorithms have been proposed for sparse classification/regression, such as decomposition algorithms [28], [27], algorithms for ℓ_1 constrained optimization problem [31], [30], [29] (interested readers please refer to [27] for more discussions about sparse classification/regression algorithms).

Since the pairwise approach reduces the ranking problem to a classification problem on document pairs, in principle, many algorithms for sparse classification can be applied to

TABLE 1
List of Notations

Notations	Meaning
$S=\{(q_i, X_i, Y_i)\}_{i=1}^{n}$	training set
m	dimension of data
p	number of pairs in set P
r	the radius of ℓ_1-ball: $\|\boldsymbol{w}\|_1 \leq r$
K	matrix in $\mathbb{R}^{p\times m}$ that contains the pairwise information
$I_C(\boldsymbol{w})$	$I_C(\boldsymbol{w}) = 0$ if condition C is satisfied, otherwise $I_C(\boldsymbol{w}) = \infty$.

obtain sparse ranking models. However, few efforts have been made to tackle the problem of learning a sparse solution for ranking. Recently, Sun et al. [10] proposed a reduction framework to reduce ranking to importance-weighted pairwise classification and then used an ℓ_1 regularized algorithm to learn a sparse ranking predictor. Despite success, it does not justify the individual contribution of each of its two parts, the reduction framework and the sparse learning algorithm. Sparse learning for ranking is a relatively new topic that needs more exploration.

3 Notations

We introduce the notations used throughout this paper. In the learning-to-rank problem, there is a labeled training set $S = \{(q_k, X_k, Y_k)\}_{k=1}^{n}$ and a test set $T = \{(q_k, X_k)\}_{k=n+1}^{n+u}$. Here, q_k denotes a query, $X_k = \{X_{k,i}\}_{i=1}^{n(q_k)}$ denotes the list of corresponding retrieved objects (i.e., documents) for q_k, and $Y_k = \{y_{k,i}\}_{i=1}^{n(q_k)}$ is the list of corresponding relevance labels provided by human, where $y_{k,i} \in \{0, 1, 2, 3, 4\}$, $n(q_k)$ represents the number of objects in the retrieved object list belongs to query q_k, and $X_{k,i}$ represents the ith object in the retrieved object list belongs to query q_k. Each $X_{k,i} \in \mathbb{R}^m$ is an m-dimensional feature vector and each attribute of $X_{k,i}$ is scaled to the range $[0, 1]$.

We define a pairs set P of comparable object pairs as following: $(k, i, j) \in P$ if and only if $X_{k,i}, X_{k,j}$ belong to the same query q_k and $y_{k,i} \neq y_{k,j}$. We use p to denote the number of pairs in P. In addition, we define an object pairwise comparison error matrix $K \in \mathbb{R}^{p\times m}$ as follows: each pair in P corresponds to a row in K. Denote the lth pair in P as $\{k_l, i_l, j_l\}$, the lth row of K as K_l. We define $K_l = y_{k_l,i_l,j_l}(X_{k_l,i_l} - X_{k_l,j_l})$, where $y_{k_l,i_l,j_l} = 1$ if $y_{k_l,i_l} > y_{k_l,j_l}$, and otherwise $y_{k_l,i_l,j_l} = -1$. Since $X_{i,j} \in [0, 1]^m$ for all i, j, we have $K_l \in [-1, 1]^m$ for all l.

We use $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ to represent the inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Let r denote the radius of an ℓ_1-ball: $\|\boldsymbol{w}\|_1 \leq r$. We introduce an indictor function $I_C(\boldsymbol{w})$: $I_C(\boldsymbol{w}) = 0$ if and only if for a given vector w, condition C is satisfied, otherwise $I_C(\boldsymbol{w}) = +\infty$. The above notations are summarized in Table 1.

4 Problem Statement

The learning-to-rank problem has a wide range of applications in information retrieval systems. We are given a labeled training set $S = \{(q_k, X_k, Y_k)\}_{k=1}^{n}$ and a test set $T = \{(q_k, X_k)\}_{k=n+1}^{n+u}$. The task of learning to rank is to construct a ranking predictor from the training data, and then sort the examples in the test set using the ranking predictor.

Following the common practice in learning-to-rank, in this paper, we only focus on learning a linear ranking predictor $f(x) = \langle w, x\rangle$. Many existing learning-to-rank algorithms use this setting. The SVM methods, such as the recently proposed RankSVM-Struct [19] and RankSVM-Primal [16], are notable algorithms for learning linear ranking predictors, which achieve a state-of-the-art performance on several benchmark data sets. These methods learn ranking models by minimizing the following form of regularized pairwise loss functions:

$$\min_{\boldsymbol{w}'} \frac{1}{2}\|\boldsymbol{w}'\|_2^2 + C \sum_{(k,i,j)\in P} \ell(y_{k,i,j}\boldsymbol{w}'^T(X_{k,i} - X_{k,j})), \qquad (1)$$

where $\ell(x)$ can be the hinge loss $\ell(x) = max(0, 1 - x)$ or the squared hinge loss $\ell(x) = max(0, 1 - x)^2$, and C is a parameter to control the tradeoff between training error and the model complexity. Existing work [34] in learning-to-rank revealed that the classification-based pairwise loss function (i.e., hinge loss) is both an upper bound of 1-NDCG and 1-MAP. When we take $\ell(x) = max(0, 1 - x)$, the objective function given by (1) is the objective of Ranking SVM. There exist several algorithms, such as the quadratic programming [26] or the cutting plane algorithm [19], which minimize this objective function. If $\ell(x) = max(0,\ 1 - x)^2$, the function given by (1) becomes the objective of RankSVM-Primal [16], which is a convex and twice differentiable function that can be optimized directly via an efficient Newton descent algorithm. Despite achieving a state-of-the-art performance, a learning algorithm using these forms of objectives usually obtains dense solutions (most of the ranking predictor's coefficients are nonzero) because of the ℓ_2 regularization term.

Sparse models have been proved to be effective in many applications, including computer vision, signal processing, and bioinformatics. In this paper, we are interested in the particular problem of how the sparse learning algorithm can contribute to the improvement of the ranking accuracy. By replacing the ℓ_2 norm with the spare-inducing ℓ_1 norm, we obtain the following optimization problem:

$$\min_{\boldsymbol{w}'} \|\boldsymbol{w}'\|_1 + C \sum_{(k,i,j)\in P} \frac{1}{p} max(0, 1 - y_{k,i,j}\boldsymbol{w}'^T(X_{k,i} - X_{k,j}))^2. \qquad (2)$$

For any C in the problem in (2), there exists a corresponding r such that the problem in (2) is equivalent to the following optimization problem (see the explanation in [32, Section 1.2]):

$$\begin{aligned} &\min_{\|\boldsymbol{w}'\|_1\leq r} \frac{1}{p} \sum_{(k,i,j)\in P} max(0, 1 - y_{k,i,j}\boldsymbol{w}'^T(X_{k,i} - X_{k,j}))^2 \\ &\quad = \min_{\boldsymbol{w}'}\ I_{\|\boldsymbol{w}'\|_1\leq r}(\boldsymbol{w}') + \frac{1}{p}\sum_{i=1}^{p} max(0, 1 - (K\boldsymbol{w}')_i)^2. \end{aligned} \qquad (3)$$

The predictor $\boldsymbol{w}'$ is an m-dimensional vector constrained in the ℓ_1-ball of radius r. It is well known that the ℓ_1 constrained optimization problems like (3) usually lead to sparse solutions, but the ℓ_2 regularized formulation like (1)

does not (see [33, pages 14-15] for a detailed explanation based on a geometrical intuition).

For ease of analysis, we scale the radius of ℓ_1-ball to 1 and define $\boldsymbol{w} = \frac{1}{r}\boldsymbol{w}'$. The problem (3) can be rewritten as the following:

$$\min_{\boldsymbol{w}} G(\boldsymbol{w}) = \min_{\boldsymbol{w}} I_{\|\boldsymbol{w}\|_1 \leq 1}(\boldsymbol{w}) + \frac{r^2}{p}\sum_{i=1}^{p} max\left(0, \frac{1}{r} - (K\boldsymbol{w})_i\right)^2. \tag{4}$$

The objective function given by (4) is similar to that of the RankSVM-Primal, except for a different regularization term.

Since ℓ_1 is not differentiable everywhere, it is challenging to optimize the objective in (4), and the Newton descent algorithm used in RankSVM-Primal cannot be applied. To minimize a convex but nondifferentiable function, a straightforward way is to use the popular subgradient descent algorithm. However, the subgradient descent method has a slow convergence rate of $O(\frac{1}{\epsilon^2})$, where ϵ is the expected optimization precision. In this paper, we propose an efficient and convergence-provable optimization algorithm for the problem in (4), with a faster convergence rate of $O(\frac{1}{\epsilon})$.

5 Our Algorithm

5.1 Overview

In this section, we present our algorithm to solve the sparse learning-to-rank problem in (4). Our algorithm is based on the theory of Fenchel Duality [24], which has been used in several machine learning algorithms such as the boosting variants in [1]. Our algorithm follows the genetic algorithmic framework proposed in [1]. Since Fenchel Duality is the key ingredient in our designing methodology, we call our algorithm "FenchelRank" for short.

Let $D(\boldsymbol{w}) = -G(\boldsymbol{w})$. Then, the problem in (4) is equivalent to the following optimization problem:

$$\max_{\boldsymbol{w}} D(\boldsymbol{w}) = \max_{\boldsymbol{w}} -I_{\|\boldsymbol{w}\|_1 \leq 1}(\boldsymbol{w}) - \frac{r^2}{p}\sum_{i=1}^{p} max\left(0, \frac{1}{r} - (K\boldsymbol{w})_i\right)^2. \tag{5}$$

To maximize the objective in (5), we propose an iterative algorithm, which iteratively constructs a sequence of weight vectors: $\boldsymbol{w}_1 \rightarrow \cdots \rightarrow \boldsymbol{w}_t \rightarrow \boldsymbol{w}_{t+1} \rightarrow \cdots \rightarrow \boldsymbol{w}_T$, such that $\{D(\boldsymbol{w}_t)\}_{t=1}^{T}$ is a monotonically increasing sequence of function values: $D(\boldsymbol{w}_1) \leq \cdots \leq D(\boldsymbol{w}_t) \leq D(\boldsymbol{w}_{t+1}) \leq \cdots \leq D(\boldsymbol{w}_T)$. Suppose $\boldsymbol{w}^*$ is the best solution for $D(\boldsymbol{w})$ (i.e., the Bayes error is minimized), and $D^* = D(\boldsymbol{w}^*)$. With the constructed sequence $\{D(\boldsymbol{w}_t)\}_{t=1}^{T}$, we will prove that after T iterations, $D(\boldsymbol{w}_T)$ is guaranteed to be an ϵ-accurate solution (i.e., $D^* - D(\boldsymbol{w}_T) \leq \epsilon$) with $\epsilon = O(\frac{1}{T})$.

To improve efficiency in practice, we further define an early stopping criterion for the algorithm using the properties of Fenchel duality. Obviously, we can compare $D^* - D(\boldsymbol{w}_T)$ with ϵ to determine whether the algorithm can be stopped. However, D^* is unknown. To derive an upper bound of $D^* - D(\boldsymbol{w}_t)$, we construct another sequence of function values $P(\boldsymbol{d}_t)$ such that $P(\boldsymbol{d}_t) \geq D^*$, where $P(\boldsymbol{d}_t)$ is the primal form whose Fenchel dual form is $D(\boldsymbol{w}_t)$. Therefore, we can use $P(\boldsymbol{d}_t) - D(\boldsymbol{w}_t) \geq D^* - D(\boldsymbol{w}_t)$ to derive an early stopping criterion (see Sections 5.3 and 5.4 for more details).

The skeleton of the proposed algorithm is shown in Algorithm 1. The input of the algorithm includes a data matrix K, a desired optimization tolerance ϵ, a maximum number of iterations T, and the radius r of an ℓ_1-ball. In this algorithm, the sign function $sign(\alpha) = 1$ if $\alpha \geq 0$, otherwise $sign(\alpha) = -1$; $\mathbf{0}_m$ denotes the m-dimensional vector with all zeros, and $\boldsymbol{e}^i$ is the vector with all zeros except the ith element being 1. The algorithm initializes $\boldsymbol{w}$ to be $\mathbf{0}_m$. It stops if the early stopping criterion is satisfied (Line 2), or the maximal iteration, T, is reached.

Algorithm 1. FenchelRank algorithm
Input: pairwise data matrix K, desired accuracy ϵ, maximal iteration number T and the radius r of ℓ_1 ball.
Output: linear ranking predictor w
Initialize: $w_1 = \mathbf{0_m}$
1. For $t = 1, 2, \cdots, T$ do
//check if the early stopping criterion is satisfied
2. IF $\|g_t\|_\infty + \langle d_t, -Kw_t \rangle \leq \epsilon$
return w_t as ranking predictor w

Here $d_t = \nabla f^*(-Kw_t) = \frac{\partial f^*(-Kw)}{\partial(Kw)}|_{w=w_t}$
and $g_t = d_t^T K$
//Greedily choose a feature to update
3. Choose $j_t = argmax_j \, |(g_t)_j|$
//Compute an appropriate step size
4. Let $\mu_t = arg\max_{0 \leq \mu_t \leq 1} D((1-\mu_t)w_t + \mu_t sign((g_t)_{j_t}) e^{j_t}))$
//Update the model with the chosen feature and step size
5. Update $w_{t+1} = (1-\mu_t)w_t + \mu_t sign((g_t)_{j_t}) e^{j_t}$
7. end For
8. return w_T as ranking predictor w

In each iteration, the algorithm has three main steps: 1) checking the early stopping criterion (Line 2); 2) greedily choosing a feature to update (Line 3); and 3) finding an appropriate step size and updating the weights (Lines 4-5). In the following, we first review the properties of Fenchel duality. Then, we present how to construct the sequences $\{D(\boldsymbol{w}_t)\}_{t=1}^{T}$, $\{\boldsymbol{d}_t\}_{t=1}^{T}$ and $\{P(\boldsymbol{d}_t)\}_{t=1}^{T}$. After that, we specify the three main steps, respectively. Finally, we provide the theoretical analysis of the algorithm.

5.2 Properties of Fenchel Duality

The main properties of Fenchel Duality are the *Fenchel conjugate* (Definition 1) and the *Fenchel Duality inequalities* (Lemma 1 and 2).

Definition 1. *The Fenchel conjugate of function f is defined as* $f^*(\boldsymbol{\theta}) = \max_{\boldsymbol{x} \in dom f}(\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle - f(\boldsymbol{x}))$.

Lemma 1 (Fenchel-Young Inequality: [2], Proposition 3.3.4). *Any points $\boldsymbol{\theta}$ in the domain of function f^* and $\boldsymbol{x}$ in the domain of function f satisfy the inequality:*

$$f(\boldsymbol{x}) + f^*(\boldsymbol{\theta}) \geq \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle. \tag{6}$$

The equality holds if and only if $\boldsymbol{\theta} \in \partial f(\boldsymbol{x})$.

Lemma 2 (Fenchel Duality Inequality: [2], Theorem 3.3.5). *Let function $f : R^p \rightarrow (-\infty, +\infty]$ and $g : R^m \rightarrow (-\infty, +\infty]$ be two closed and convex functions, K be a $\mathbb{R}^{p \times m}$ matrix,*

$$\sup_{\boldsymbol{w}} -f^*(-K\boldsymbol{w}) - g^*(\boldsymbol{w}) \leq \inf_{\boldsymbol{d}} f(\boldsymbol{d}) + g(\boldsymbol{d}^T K). \tag{7}$$

The equality holds when $0 \in (dom(g) - K^T dom(f))$.

5.3 Constructing the Sequences

To construct the sequences $\{D(\boldsymbol{w}_t)\}_{t=1}^T$, $\{P(\boldsymbol{d}_t)\}_{t=1}^T$, and $\{\boldsymbol{d}_t\}_{t=1}^T$, we define

$$g^*(\boldsymbol{w}) = I_{\|\boldsymbol{w}\|_1 \leq 1}(\boldsymbol{w}) \quad \text{and} \quad f^*(\boldsymbol{\theta}) = \frac{r^2}{p}(max(0, 1+\boldsymbol{\theta}))^2.$$

We have $f^*(-K\boldsymbol{w}) = \frac{r^2}{p}\sum_{i=1}^p max(0, \frac{1}{r} - (K\boldsymbol{w})_i)^2$. The objective in (5) can be rewritten by combining f^* and g^*:

$$\max_{\boldsymbol{w}} D(\boldsymbol{w}) = \max_{\boldsymbol{w}} -f^*(-K\boldsymbol{w}) - g^*(\boldsymbol{w}). \tag{8}$$

This is exactly the same as the left-hand side of (7). Accordingly, we can define the upper bound of (5) by the right hand side of (7):

$$\min_{\boldsymbol{d}} P(\boldsymbol{d}) = \min_{\boldsymbol{d}} f(\boldsymbol{d}) + g(\boldsymbol{d}^T K), \tag{9}$$

where g and f are the Fenchel conjugates of g^* and f^*, respectively, which is given by the following lemma.

Lemma 3. *The Fenchel conjugate of*

$$f^*(-K\boldsymbol{w}) = \sum_{i=1}^p \frac{r^2}{p}\max\left(0, \frac{1}{r} - (K\boldsymbol{w})_i\right)^2 \quad is$$
$$f(\boldsymbol{d}) = \sum_{i=1}^p \left(\frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i + I_{d_i \geq 0}(d_i)\right).$$

The Fenchel conjugate of $g^*(\boldsymbol{w}) = I_{\|\boldsymbol{w}\|_1 \leq 1}(\boldsymbol{w})$ *is* $g(\boldsymbol{d}^T K) = \|\boldsymbol{d}^T K\|_\infty$.

Proof. Let $x = -K\boldsymbol{w}$. According to Definition 1, we have

$$f(\boldsymbol{d}) = \max_{\boldsymbol{x}} \langle \boldsymbol{d}, \boldsymbol{x}\rangle - f^*(\boldsymbol{x})$$
$$= \sum_{i=1}^p \max_{x_i} d_i x_i - \frac{r^2}{p} max\left(0, \frac{1}{r} + x_i\right)^2.$$

Let $h(d_i) = \max_{x_i} d_i x_i - \frac{r^2}{p} max(0, \frac{1}{r} + x_i)^2$, thus $f(\boldsymbol{d}) = \sum_{i=1}^p h(d_i)$.

If $d_i < 0$, letting $x_i \to -\infty$, we have

$$h(d_i) = d_i x_i - \frac{r^2}{p}\max\left(0, \frac{1}{r} + x_i\right)^2 = d_i x_i \to \infty. \tag{10}$$

If $d_i > 0$, we, respectively, discuss the following two cases: 1) If $(\frac{1}{r} + x_i) \geq 0$, then $h(d_i) = \max_{x_i \geq -\frac{1}{r}} d_i x_i - \frac{r^2}{p}(\frac{1}{r} + x_i)^2 = \max_{x_i \geq -\frac{1}{r}} -\frac{r^2}{p}(x_i + \frac{1}{r} - \frac{p}{2r^2}d_i)^2 + \frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i$, which implies that $h(d_i) \leq \frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i$. The equality holds when $x_i + \frac{1}{r} - \frac{p}{2r^2}d_i = 0$. 2) If $(\frac{1}{r} + x_i) \leq 0$, then

$$h(d_i) = \max_{x_i < -\frac{1}{r}} d_i x_i.$$

the maximum of $d_i x_i$ is obtained when $x_i = -\frac{1}{r}$, thus $h(d_i) < -\frac{1}{r}d_i \leq \frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i$.

To sum up, we get $h(d_i) = \frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i$ when $d_i \geq 0$, otherwise $h(d_i) = \infty$.

Therefore, we have

$$f(\boldsymbol{d}) = \sum_{i=1}^p \left(\frac{p}{4r^2}d_i^2 - \frac{1}{r}d_i + I_{d_i \geq 0}(d_i)\right).$$

Next, we prove that the Fenchel conjugate of $g^*(\boldsymbol{w})$ is

$$g(\boldsymbol{d}^T K) = \|\boldsymbol{d}^T K\|_\infty g(\boldsymbol{d}^T K) = \max_{\boldsymbol{w}} \langle \boldsymbol{d}^T K, \boldsymbol{w}\rangle - g^*(\boldsymbol{w})$$
$$= \max_{\|\boldsymbol{w}\|_1 \leq 1} \langle \boldsymbol{d}^T K, \boldsymbol{w}\rangle = \max_i |(\boldsymbol{d}^T K)_i| = \|\boldsymbol{d}^T K\|_\infty.$$

□

Therefore, the upper bound function can be rewritten as:

$$\min_{\boldsymbol{d}} P(\boldsymbol{d}) = \min_{\boldsymbol{d}} f(\boldsymbol{d}) + g(\boldsymbol{d}^T K)$$
$$= \min_{\boldsymbol{d} \geq \boldsymbol{0}} \frac{p}{4r^2}\|\boldsymbol{d}\|_2^2 - \frac{1}{r}\|\boldsymbol{d}\|_1 + \|\boldsymbol{d}^T K\|_\infty. \tag{11}$$

Next, we describe the construction of $\boldsymbol{d}_t$. At the beginning of each iteration, one is given $\boldsymbol{w}_t$ from previous iteration. We define $\boldsymbol{d}_t$ by:

$$\boldsymbol{d}_t = \nabla f^*(-Kw_t) = \frac{\partial f^*(-Kw)}{\partial(Kw)}|_{w=w_t}, \tag{12}$$

where the ith coordinate of $\boldsymbol{d}_t$ is defined by:

$$\boldsymbol{d}_t^{(i)} = \begin{cases} \frac{2r^2}{p}\left(\frac{1}{r} - (K\boldsymbol{w}_t)_i\right) & \text{if } \frac{1}{r} - (K\boldsymbol{w}_t)_i \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

We use this definition of $\boldsymbol{d}_t$ because it satisfies the equality condition of the inequality (6) in Lemma 1, and, thus, we can use it to derive the early stopping criterion in the next section.

5.4 Checking the Early Stopping Criterion

The early stopping criterion is satisfied when the gap between the primal form and the dual does not exceed the desired optimization tolerance ϵ in iteration t:

$$P(\boldsymbol{d}_t) - D(\boldsymbol{w}_t) \leq \epsilon. \tag{13}$$

Now, we show how to compute $P(\boldsymbol{d}_t) - D(\boldsymbol{w}_t)$. In each iteration, we update the weights by an update rule (see Section 5.6) which ensures that $\|\boldsymbol{w}_t\|_1 \leq 1$ for all t (Line 5). Therefore, we have $g^*(\boldsymbol{w}_t) = 0$ for all t, and $D(\boldsymbol{w}_t) = -f^*(-K\boldsymbol{w}_t)$. With (11), we have

$$P(\boldsymbol{d}_t) - D(\boldsymbol{w}_t) = \|\boldsymbol{d}_t{}^T K\|_\infty + f(\boldsymbol{d}_t) + f^*(-K\boldsymbol{w}_t). \tag{14}$$

It is easy to verify that $f^*(-K\boldsymbol{w}_t)$ is differentiable. Hence, with (12) and Lemma 1, the following equality holds:

$$f(\boldsymbol{d}_t) + f^*(-K\boldsymbol{w}_t) = \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle. \tag{15}$$

Combining (15) and (14), we get

$$P(\boldsymbol{d}_t) - D(\boldsymbol{w}_t) = \|\boldsymbol{d}_t{}^T K\|_\infty + \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle. \tag{16}$$

Finally, the early stopping criterion is

$$\|\boldsymbol{d}_t{}^T K\|_\infty + \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle \leq \epsilon. \tag{17}$$

5.5 Greedily Choosing a Feature

At iteration t, the algorithm greedily selects a feature j_t, which has the largest absolute value of $(\boldsymbol{d}_t^T K)_j$. This step is similar to the step of selecting a weak learner (i.e., by finding a single feature with the best edge), which is a common building block

in many boosting algorithms (i.e., AdaBoost [25]).

5.6 Updating the Weights of the Ranking Model

Given the selected feature from the previous section, we set $\boldsymbol{w}_{t+1}$ to be the convex combination of $\boldsymbol{w}_t$ and the selected feature

$$\boldsymbol{w}_{t+1} = (1-\mu_t)\boldsymbol{w}_t + \mu_t sign\big(\big(d_t^T K\big)_{j_t}\big)\boldsymbol{e}^{j_t}. \tag{18}$$

The coefficient of the combination, denoted by μ_t, is calculated so as to maximize the increase of the dual objective

$$\begin{aligned}\mu_t &= \underset{0\le\mu\le 1}{argmax}\ D(\boldsymbol{w}_{t+1}) \\ &= \underset{0\le\mu\le 1}{argmax}\ D\big((1-\mu)\boldsymbol{w}_t + \mu sign\big(\big(\boldsymbol{d}_t^T K\big)_{j_t}\big)\boldsymbol{e}^{j_t}\big).\end{aligned} \tag{19}$$

Let K_i be the ith row in K. Denoting

$$\boldsymbol{b}_t = sign\big(\big(\boldsymbol{d}_t^T K\big)_{j_t}\big)\boldsymbol{e}^{j_t} - \boldsymbol{w}_t \quad \text{and} \quad \boldsymbol{a}_t^{(i)} = \frac{1}{r} - K_i\boldsymbol{w}_t.$$

Equation (19) can be simplified as the following quadratic equation:

$$\mu_t = \underset{0\le\mu\le 1}{argmin} \sum_{i=1}^{p} max\big(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu\big)^2. \tag{20}$$

Equation (20) only contains a single variable μ and can be solved as follows: For all i, $max(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu)^2 = 0$ if $\boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t^{\mu} \le 0$, otherwise

$$max\big(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu\big)^2 = \big(\boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu\big)^2.$$

For all $K_i\boldsymbol{b}_t \neq 0$, we denote $\boldsymbol{c}_t^{(i)} = \frac{\boldsymbol{a}_t^{(i)}}{K_i\boldsymbol{b}_t}$.Let

$$L = \{0, 1\} \cup \big\{\boldsymbol{c}_t^{(i)} | \boldsymbol{c}_t^{(i)} \in [0, 1]\big\}.$$

We sort the elements in L in a decreasing order: $1 = l_1 \ge l_2 \ge \cdots \ge l_n = 0$, where $l_k \in L (1 \le k \le n)$. In every interval $[l_{k+1}, l_k]$, the values of $max(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu)^2$ ($i = 1$to p) are known in advance, and, thus, the objective in (20) becomes a simple quadratic function of μ that has a closed-form solution. Hence, we can find $n-1$ candidate values of μ_t by solving the $n-1$ subproblems in (21), respectively, and, thus, obtain the final μ_t by selecting the one that minimizes $\sum_{i=1}^{p} max(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu)^2$

$$\mu_t^{(k,k+1)} = \underset{l_{k+1}\le\mu\le l_k}{argmin} \sum_{i=1}^{p} max\big(0, \boldsymbol{a}_t^{(i)} - K_i\boldsymbol{b}_t\mu\big)^2. \tag{21}$$

To ensure that $\boldsymbol{w}_t$ satisfies the constraint of ℓ_1-ball, $\boldsymbol{w}_t \le 1$, we initialize the weight vector $\boldsymbol{w}_1$ to be the zero vector and restrict the range of coefficient μ_t to be in $[0, 1]$. It is easy to verify that $\|\boldsymbol{w}_1\|_1 \le 1$, and for any t, $\|\boldsymbol{w}_{t+1}\|_1 = \|(1-\mu_t)\boldsymbol{w}_t + \mu_t\boldsymbol{e}^i\| \le (1-\mu_t)\|\boldsymbol{w}_t\| + \mu_t\|\boldsymbol{e}^i\| \le 1$.

We note that, in FenchelRank, we can also use

$$\mu_t = max\left(min\left(0, \frac{\langle \boldsymbol{d}_t, K\boldsymbol{b}_t\rangle}{\|K\boldsymbol{b}_t\|_2^2}\right), 1\right)$$

to find an appropriate step size in each iteration. This is similar to the calculation rule used in [1]. The analysis in [1] shows that this kind of calculation rule of μ_t leads to a sufficient increase of the objective. The calculation rule used in FenchelRank obtains a μ_t by directly maximizing the increase of the objective, which results in a larger increase of the objective than the rule in [1].

6 Theoretical Analysis

In this section, we analyze the convergence rate of FenchelRank and show the correctness of the early stopping criterion. Please remind that, in each iteration t, the algorithm maintains a primal-dual pair: the dual form $D(\boldsymbol{w}_t)$ given in (5) and the primal form $P(\boldsymbol{d}_t)$ given in (11). Suppose $\boldsymbol{w}^*$ be the optimal solution for $D(\boldsymbol{w})$, that is $\boldsymbol{w}^* = argmax_{\boldsymbol{w}} D(\boldsymbol{w})$. We define $\epsilon_t = D(\boldsymbol{w}^*) - D(\boldsymbol{w}_t)$ as the gap between the optimal solution and the solution obtained at iteration t.

6.1 Convergence Rate

The following theorem establishes the upper bound of required iterations to obtain an ϵ-accurate solution.

Theorem 1. *The FenchelRank algorithm terminates after at most $16r^2/\epsilon - 1$ iterations and returns an ϵ-accurate solution, where $r^2 \ge 0.125$.*

Our proof is based on the properties of Fenchel Duality and strongly convex functions.

Before presenting the proof, we first introduce some concepts and lemmas, which will be used in the analysis.

A function f is convex if for all $x, y \in dom(f)$, and $0 \le \theta \le 1$, $f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta)f(y)$. A vector ϕ is a subgradient of a function f at x if:

$$\forall y, \langle \phi, y - x\rangle \le f(y) - f(x). \tag{22}$$

We define the set of subgradients of f at x as $\partial f(x)$. Note that if f is convex and differentiable at x, then $\partial f(x)$ consists of only a single vector, the gradient of f at x, denoted by $\nabla f(x)$.

Definition 2. *A closed and convex function f is λ-strongly convex w.r.t. a norm $\|\cdot\|$ if $\forall x, y$ and $\forall g \in \partial f(x)$ have*

$$f(y) - f(x) - \langle y - x, g\rangle \ge \frac{\lambda}{2}\|x - y\|^2. \tag{23}$$

For instance, $\frac{1}{2}\|\boldsymbol{d}\|_2^2$ is 1-strongly convex w.r.t. norm $\|\cdot\|_2$.

The following two lemmas will be used in the proof of Theorem 1. We refer the readers to [1] for the proofs of these two lemmas.

Lemma 4 ([1], Lemma 18). *Let f be a closed and λ-strong convex over S with respect to a norm $\|\cdot\|_*$. Let f^* be the Fenchel conjugate of f; then f^* is differentiable and its gradient satisfies $\nabla f^*(w) = argmax_{d\in S}\langle w, d\rangle - f(d)$. For all x, y we have*

$$f^*(x + y) - f^*(x) \le \langle \nabla f^*(x), y\rangle + \frac{1}{2\lambda}\|y\|^2. \tag{24}$$

Lemma 5 ([1], Lemma 20). *Let $1 \ge \epsilon_1 \ge \epsilon_2 \ge \ldots$ be a sequence such that, for all $t \ge 1$, we have $\epsilon_t - \epsilon_{t+1} \ge r\epsilon_t^2$ for some constant $r \in (0, 0.5)$. Then, for all t we have $\epsilon_t \le \frac{1}{r(t+1)}$.*

Now, we turn to the proof of Theorem 1. Some proof techniques are similar to those of [1].

Proof. since $f(\boldsymbol{d}) = \frac{p}{4r^2}\|\boldsymbol{d}\|_2^2 - \frac{1}{r}\sum_i \boldsymbol{d}_i$ is $\frac{p}{2r^2}$-strongly convex with the l_2 norm $\|\boldsymbol{d}\|_2^2$, using Lemma 4, we have

$$f^*(\boldsymbol{x}+\boldsymbol{y}) - f^*(\boldsymbol{x}) \leq \langle \nabla f^*(\boldsymbol{x}), \boldsymbol{y}\rangle + \frac{r^2}{p}\|\boldsymbol{y}\|_2^2, \tag{25}$$

for all $\boldsymbol{y}, \boldsymbol{x} \in dom(f^*)$. Since

$$\boldsymbol{w}_{t+1} = (1-\mu_t)\boldsymbol{w}_t + \mu_t sign((\boldsymbol{d}_t^T K)_{j_t})\boldsymbol{e}^{j_t},$$

we can rewrite it into

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mu_t(sign((\boldsymbol{d}_t^T K)_{j_t})\boldsymbol{e}^{j_t} - \boldsymbol{w}_t).$$

For ease of presentation, we denote

$$\boldsymbol{b}_t = (sign((\boldsymbol{d}_t^T K)_{j_t})\boldsymbol{e}^{j_t} - \boldsymbol{w}_t,$$

thus $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mu_t \boldsymbol{b}_t$.

Letting $\boldsymbol{x} = -K\boldsymbol{w}_t, \boldsymbol{y} = -\mu_t K\boldsymbol{b}_t$, we have $\boldsymbol{x}+\boldsymbol{y} = -K\boldsymbol{w}_{t+1}$. In terms of (12), we have $\nabla f^*(\boldsymbol{x}) = \nabla f^*(-K\boldsymbol{w}_t) = \boldsymbol{d}_t$. Substituting $\boldsymbol{x}, \boldsymbol{y}, \nabla f^*(\boldsymbol{x})$ into (25), we obtain

$$\begin{aligned} &f^*(-K\boldsymbol{w}_t - \mu_t K\boldsymbol{b}_t) - f^*(-K\boldsymbol{w}_t) \\ &\quad\leq \langle -\mu_t K\boldsymbol{b}_t, \boldsymbol{d}_t\rangle + \frac{r^2}{p}\|\mu_t K\boldsymbol{b}_t\|_2^2. \end{aligned} \tag{26}$$

According to the definition $\epsilon_t = D(\boldsymbol{w}^*) - D(\boldsymbol{w}_t)$, we have

$$\begin{aligned} \epsilon_t - \epsilon_{t+1} &= D(\boldsymbol{w}_{t+1}) - D(\boldsymbol{w}_t) \\ &= f^*(-K\boldsymbol{w}_t) - f^*(-K\boldsymbol{w}_{t+1}). \end{aligned} \tag{27}$$

Combining (26) and (27), we get

$$\epsilon_t - \epsilon_{t+1} \geq \langle \mu_t K\boldsymbol{b}_t, \boldsymbol{d}_t\rangle - \frac{r^2}{p}\|\mu_t K\boldsymbol{b}_t\|_2^2. \tag{28}$$

According to Theorem 2 (see below), we have

$$\epsilon_t \leq \|\boldsymbol{d}_t^T K\|_\infty + \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle = \langle \boldsymbol{d}_t, K\boldsymbol{b}_t\rangle. \tag{29}$$

Since each attribute of $X_{i,j}$ is scaled to the range $[0,1]$, we can verify $K_{i,j} \in [-1,1]$. We have

$$\begin{aligned} \|K\boldsymbol{b}_t\|_2^2 &= \|K((sign((\boldsymbol{d}_t^T K)_{j_t})\boldsymbol{e}^{j_t} - \boldsymbol{w}_t)\|_2^2 \\ &\leq 2\|Ksign((\boldsymbol{d}_t^T K)_{j_t})\boldsymbol{e}^{j_t}\|_2^2 + 2\|K\boldsymbol{w}_t\|_2^2 \\ &= 2\|K\boldsymbol{e}^{j_t}\|_2^2 + 2\|K\boldsymbol{w}_t\|_2^2 \\ &= 2\sum_{i=1}^{p}\left(K_{i,j_t}^2 + \left(\sum_{j=1}^{d} K_{i,j}\boldsymbol{w}_t^j\right)^2\right) \\ &\leq 2\sum_{i=1}^{p}\left(K_{i,j_t}^2 + \left(\sum_{j=1}^{d} |\boldsymbol{w}_t^j|\right)^2\right) \\ &\leq 2\sum_{i=1}^{p}(1+1) = 4p, \end{aligned} \tag{30}$$

where the first inequality follows from the Cauchy-Schwarz inequality, the second inequality follows from $K_{i,j} \in [-1,1]$,and the last inequality utilizes the constraint $|\boldsymbol{w}|_1 \leq 1$.

Substituting (29) and (30) into (28), we have

$$\epsilon_t - \epsilon_{t+1} \geq \mu_t\epsilon_t - \frac{r^2}{p}\mu_t^2 * 4p = \mu_t\epsilon_t - 4r^2\mu_t^2. \tag{31}$$

In terms of (27), to maximize the increase $D(\boldsymbol{w}_{t+1}) - D(\boldsymbol{w}_t)$, we need to maximize the right side of (31). We can verify that:

$$\begin{aligned} \mu_t\epsilon_t - 4r^2\mu_t^2 &= 4r^2\left(\frac{\epsilon_t}{8r^2}\right)^2 - \left(\mu_t - \frac{\epsilon_t}{8r^2}\right)^2 \\ &\leq 4r^2\left(\frac{\epsilon_t}{8r^2}\right)^2 = \frac{\epsilon_t^2}{16r^2}. \end{aligned} \tag{32}$$

The last equality holds if we set $\mu_t = \epsilon_t/8r^2$. Thus, we have

$$\epsilon_t - \epsilon_{t+1} \geq \epsilon_t^2/16r^2. \tag{33}$$

Equation (33) implies FenchelRank guarantees that the dual objective $D(\boldsymbol{w})$, at least, increases by $\epsilon_t^2/16r^2$ at each iteration.

Because $D(\boldsymbol{w}_1) = -f^*(0) = -\frac{r^2}{p}\sum_{(i,j)\in P}\frac{1}{r^2} = -1$ and $D(\boldsymbol{w}^*) = -\frac{r^2}{p}\sum_{(i,j)\in P} max(0, \frac{1}{r} - y_{ij}\boldsymbol{w}^{*T}(x_i - x_j))^2 \leq 0$, we have $\epsilon_1 = D(\boldsymbol{w}^*) - D(\boldsymbol{w}_1) \leq 1$. We also have $\epsilon_t - \epsilon_{t+1} \geq \epsilon_t^2/16r^2$ for all t according to (33), that is $1 \geq \epsilon_1 \geq \epsilon_2 \geq \cdots \geq \epsilon_t$. Using Lemma 5, we get $\epsilon_t \leq \frac{16r^2}{t+1}$. In other words, if the iteration $t \geq \frac{16r^2}{\epsilon} - 1$, then Algorithm 1 can guarantee the obtainment of an ϵ-accurate solution. This concludes our proof. □

6.2 Correctness of the Early Stopping Criterion

In practice, the convergence rate of the FenchelRank algorithm is pessimistic. The algorithm usually needs a much less iterations to converge. To improve the efficiency in practice, we define an early stopping criterion (Line 2 in Algorithm 1).

Now we show that, if the early stopping criterion is satisfied at iteration t, the solution $\boldsymbol{w}_t$ is guaranteed to be an ϵ-accurate solution.

Theorem 2. *For all t, we have $\epsilon_t \leq \|\boldsymbol{d}_t^T K\|_\infty + \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle$.*

The proof is similar to [1, Lemma 11].

As a consequence of Theorem 2, if $\|\boldsymbol{d}_t^T K\|_\infty + \langle \boldsymbol{d}_t, -K\boldsymbol{w}_t\rangle \leq \epsilon$, then we have $\epsilon_t \leq \epsilon$, and, thus, the algorithm obtains an ϵ-accurate solution at iteration t. This proves the correctness of the early stopping criterion.

7 Discussions

The generic algorithmic framework of FenchelRank is similar to that of the algorithms in [1]. However, the algorithms in [1] are problem dependent, i.e., they depend on their loss functions. The generic algorithmic framework in [1] is not directly applicable in other problems. There are two major differences between FenchelRank and the algorithms in [1]: 1) In this paper, our pairwise ranking loss function used in the sparse learning-to-rank problem (4) is different from those in [1]. It is nontrivial to modify the generic algorithmic framework in [1] to work with a particular loss function, which includes deriving the Fenchel dual form of the loss function, designing how to find an appropriate step size, and deriving the results of convergence rate. 2) In each iteration, FenchelRank uses a

different rule to calculate the step size for weights updating. This rule maximizes the increase of the dual objective, and it can result in a much faster convergence of the objective than the rule similar to [1].

8 Experiments

In this section, we evaluate the ranking performance of our proposed method on several publicly available benchmark data sets. The results show that: 1) The proposed FenchelRank, with the sparse-inducing ℓ_1 constraint, has superior ranking accuracies gain compared to RankSVM-Primal, whose objective shares the same loss function with ours but uses the ℓ_2-norm as the regularization term. 2) Compared to other sparse learning-to-rank algorithms, FenchelRank obtains sparser ranking models and has superior performance on both accuracies and efficiency. 3) Compared to other state-of-the-art algorithms for ranking, FenchelRank also achieves competitive performance on ranking accuracies.

8.1 Data Sets

We conduct our experiments on the LETOR 3.0 and LETOR 4.0 data collections [20], which are publicly available benchmarks for learning-to-rank. LETOR 3.0 contains seven data sets for four search tasks of document retrieval: topic distillation (TD2003, TD2004), home page finding (HP2003, HP2004), named page finding (NP2003, NP2004), and medical document retrieval (OHSUMED). LETOR 4.0 contains two large-scale data sets, MQ2007 and MQ2008. In our experiments, we test the performances of ranking algorithms on four small data sets, TD2004, HP2004, NP2004, OHSUMED, and one large-scale data set, MQ2008. In each data set, the features are extracted from the query-document pairs, covering a wide range including low-level features (i.e., TF, IDF) and high-level features (i.e., BM25, language models).

Each of the five data sets is divided into five folds. There is a training set, a validation set, and a test set in each fold, which can be used to conduct cross validation.

8.2 Evaluation Measures

To evaluate the performance of ranking models, we use MAP [21] and NDCG [23] as the evaluation measures.

MAP is a standard evaluation measure widely used in information retrieval systems. It works for cases with binary relevance judgments: relevant and irrelevant. MAP is the mean of average precisions over a set of queries. Precision at position $j(P@j)$ [21] represents the proportion of relevant documents within the top j retrieved documents, which can be calculated by:

$$P(j) = \frac{N_{pos}(j)}{j}, \tag{8.1}$$

where $N_{pos}(j)$ denotes the number of relevant documents within the top j documents. Given a query q_i, the average precision of q_i is defined as the average of all $P@j(j = 1, 2, \ldots, n)$ and can be calculated by the following equation:

$$AvgP_i = \sum_{j=1}^{M} \frac{P(j) \times pos(j)}{N_{pos}}, \tag{8.2}$$

where j is the position, M is the number of retrieved documents and $pos(j)$ is an indicator function. If the document at position j is relevant, then $pos(j)$ is 1, or else $pos(j)$ is 0. N_{pos} represents the total number of relevant documents for query q_i. $P(j)$ is the precision at the given position j.

NDCG is another popular evaluation criterion for comparing ranking performance in information retrieval. Unlike MAP, NDCG can deal with the cases which have more than two levels of relevance judgments. Given a query q_i, DCG score at position m is defined as:

$$DCG@m = \sum_{j=1}^{m} \frac{2^{r(j)} - 1}{log(1 + j)}, \tag{8.3}$$

where $r(j)$ is the grade of the jth document. Then, NDCG score at position m in the ranking list of documents can be calculated by the equation as follows:

$$NDCG@m = \frac{1}{Z_m} DCG@m, \tag{8.4}$$

where Z_m is the maximum value of $DCG@m$, which means that the value of NDCG ranges from 0 to 1.

8.3 Experiment Protocols

To evaluate the performance of our proposed method, we conduct three experiments.

The aim of the first experiment is to empirically justify whether the sparse models can help to improve ranking performance. For fair comparison, we compare the ranking accuracies and sparsity ratios of FenchelRank versus RankSVM-Primal. RankSVM-Primal shares the same loss function with FenchelRank but uses ℓ_2 norm as the regularization term. Thus, RankSVM-Primal learns a dense model while FenchelRank learns a sparse one. This comparison shows us how the sparse models can contribute to improve the ranking accuracies.

In the second experiment, we compare the ranking accuracies, sparsity ratios and training time of FenchelRank versus other methods for sparse ranking, i.e., RSRank [10], a variant of RSRank based on truncated gradient descent method and a feature selection method based on RankSVM-Primal.

In the last experiment, we compare the ranking accuracies of FenchelRank versus other state-of-the-art learning-to-rank methods. Since our method learns linear predictors, we choose three linear ranking methods and one tree-based nonlinear ranking method for comparison.

In all of the experiments, the parameter r in our method is chosen in the set $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ by cross validation. Since the top 10 documents ranked by a ranking model are viewed as the most important ones in web search, we choose the parameters, which achieve the best value of NDCG@10 on the validation set as test parameters on the test data. In all of our experiments, we fix the maximum iteration $T = 1,000$ and desired optimization accuracy $\epsilon = 0.001$ on all the data sets. If the algorithm meets the desired accuracy ϵ or it reaches the maximum iteration T, then it stops and returns the learned ranking model.

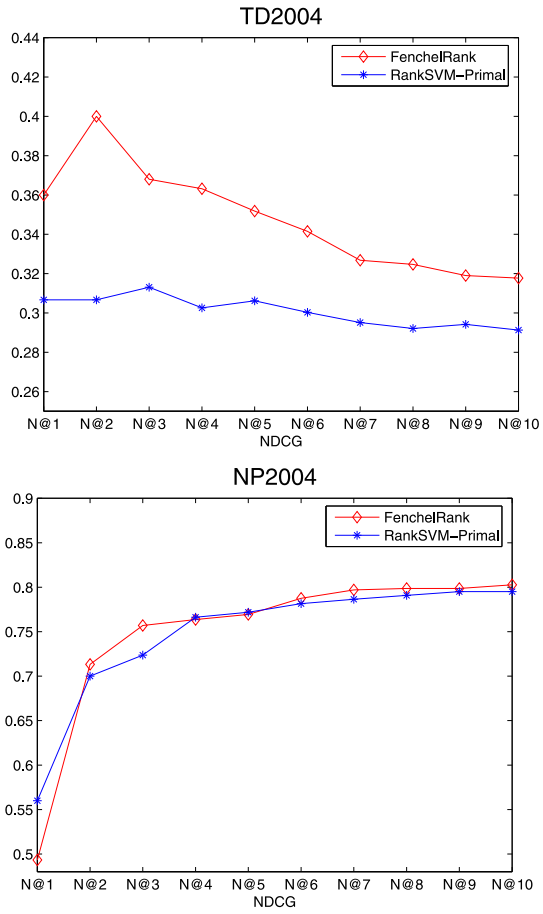


Fig. 2. Ranking accuracies of FenchelRank and RankSVM-Primal on TD2004 (top) and NP2004 (bottom).

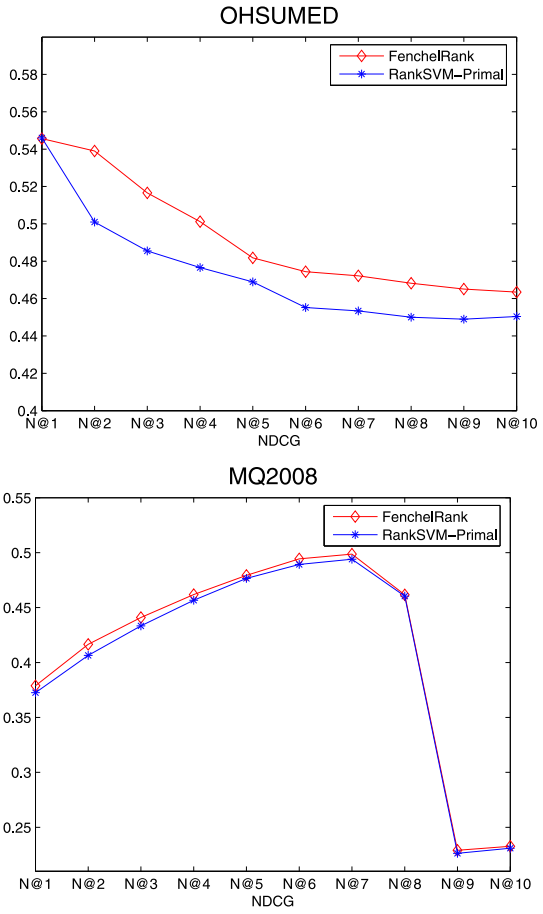


Fig. 3. Ranking accuracies of FenchelRank and RankSVM-Primal on OHSUMED (top) and MQ2008 (bottom).

8.4 Comparing with RankSVM-Primal

8.4.1 Results on Ranking Accuracies

In our first experiment, we compare the performance of FenchelRank with RankSVM-Primal. RankSVM-Primal is one of the state-of-the-art learning-to-rank algorithms. Both algorithms use the same pairwise loss function (the second term in (5)). The experiment results of RankSVM-Primal on the four data sets in LETOR 3.0 are cited from the LETOR website,[3] while the results on MQ2008 are obtained by directly running the open source code of RankSVM-Primal.[4]

Figs. 2, 3, 4, and Table 2 show the ranking accuracies of FenchelRank versus RankSVM-Primal on the five data sets with respect to NDCG and MAP. We can observe that the FenchelRank algorithm, based on our proposed objective function with the sparse-inducing ℓ_1 norm, performs significantly better than RankSVM-Primal. Here are some statistics. On TD2004, FenchelRank performs 0.3202 at NDCG@10, a 9.9 percent increase compared with RankSVM-Primal. On HP2004, the value of NDCG@10 using the FenchelRank algorithm is 0.8274, a 7 percent increase compared with RankSVM-Primal. On OHSUMED, FenchelRank performs 0.4637 at NDCG@10, compared to 0.4504 of RankSVM-Primal, which indicates a 3 percent increase.

3. http://research.microsoft.com/en-us/um/beijing/projects/letor/letor3baseline.aspx.
4. http://olivier.chapelle.cc/primal/.

In addition, we provide the significance test (t-test) results of FenchelRank versus RankSVM-Primal. As can be seen in Table 3, the improvement of FenchelRank over RankSVM-primal is statistically significant (i.e., with a p-value less than 0.05) on OHSUMED, HP2004, and TD2004. This validates that learning a sparse ranking model can help to make statistically significant improvement in ranking.

Since the only difference between the objectives of FenchelRank and RankSVM-Primal is the regularization

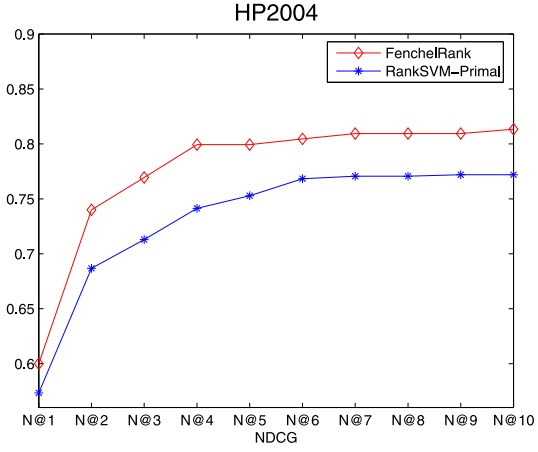


Fig. 4. Ranking accuracies of FenchelRank and RankSVM-Primal on HP2004.

TABLE 2
MAP on the Five Data Sets from the LETOR Distribution

MAP	RankSVM-Primal	FenchelRank
TD2004	0.2061	**0.2368**
NP2004	0.6755	**0.6830**
HP2004	0.6712	**0.7447**
MQ2008	0.4744	**0.4785**
OHSUMED	0.4446	**0.4486**

TABLE 3
The T-Test Results of FenchelRank versus RankSVM-Primal

	N@1	N@3	N@5	N@10	MAP
OHSUMED					
p-value	0.5	**0.0053**	0.0602	**0.0455**	0.0838
TD2004					
p-value	0.2089	**0.0189**	**0.0268**	**0.0204**	**0.0005**
HP2004					
p-value	**0.0258**	**0.0144**	**0.0163**	**0.0267**	**0.0137**
NP2004					
p-value	0.5	0.0924	0.3300	0.1352	0.4034
MQ2008					
p-value	0.3506	0.1239	0.2731	0.3891	0.1650

term, the experimental results can empirically justify that sparsity constraints (i.e., ℓ_1-norm) on learning a ranking model can help to significantly improve the ranking accuracy. The idea of learning a sparse ranking model can also be applied in other learning-to-rank methods and have the potential to boost their ranking performance.

8.4.2 Results on Sparsity Ratios

To investigate the differences between the learned models of FenchelRank and RankSVM-Primal, we report their sparsity ratios. We define the sparsity ratio on a ranking model w as $\frac{|w|_0}{m}$, where $|w|_0$ represents the number of nonzero coefficients in w and m represents the dimension of w. For each data set, we first calculate the sparsity ratio on the ranking model obtained in each of the five folds, respectively. Then, we average the five sparsity ratios as the final sparsity ratio. Table 4 shows the sparsity ratios on the five data sets obtained by FenchelRank and RankSVM-Primal. The ranking models obtained by FenchelRank are significantly sparser than those of RankSVM-Primal. This validates that the FenchelRank algorithm tends to obtain sparse ranking models. Interestingly, on NP2004, HP2004, and TD2004 (all of their documents are from the .GOV corpus [20]), the indexes of the most frequent features appearing in the final models obtained by FenchelRank are 22, 23, 27, 40, 46, 52, and 61, where 22, 23, and 61 are BM25 features, 27 and 40 are features of language models, 46 is hyper-link feature, and 52 is HostRank feature [20]. This is desirable because the features like BM25 and language models are known to be effective for document retrieval. Moreover, most of these frequent features have high values of NDCG@10 in Fig. 1.

8.5 Comparing with Other Methods for Sparse Ranking

8.5.1 Comparing with RSRank and Its Variant

To compare the ranking performance of the proposed FenchelRank algorithm with other methods for sparse ranking, we carefully implement the RSRank algorithm [10]. The RSRank algorithm optimizes the following objective ([10, (15)], with our notations: $\boldsymbol{w}' = \beta$ and $C = \frac{1}{g}$):

$$\min_{\boldsymbol{w}'} \|w'\|_1 + C \sum_{(k,i,j)\in P} W(y_i, y_j, \boldsymbol{w}')\psi(\boldsymbol{w}'^T(X_{k,i} - X_{k,j})), \quad (8.5)$$

where $W(y_i, y_j, \boldsymbol{w}')$ is a weight function that assigns different importance to different document pairs, and $\psi(\boldsymbol{w}'^T(X_{k,i} - X_{k,j}))$ is a modified Huber loss, which plays a similar role as the squared hinge loss in FenchelRank. In RSRank, the truncated gradient descent algorithm is applied to solve the ℓ_1 regularized optimization problem in (8.5).

The loss function of RSRank (the second part in (8.5)) is quite different from that of FenchelRank. For fair comparison, we implement TGRank, a variant of RSRank. TGRank solves the optimization problem in (2) using truncated gradient descent. Because the objective in (2) is equivalent to the objective of FenchelRank in (3) (see Section 4), the main differences between TGRank and FenchelRank are in their learning algorithms.

Table 5 shows the ranking accuracies of FenchelRank, RSRank,[5] and TGRank on four data sets. The results show that FenchelRank has superior ranking accuracies compared to the other two algorithms for sparse ranking.

In addition, we compare the sparsity ratios of the models learned by FenchelRank, RSRank, and TGRank, respectively. As can be seen in Table 4, FenchelRank

TABLE 4
The Sparsity Ratios on the LETOR Distribution
(Denote RankSVM-Primal as RSVM-P for Short)

Dataset	RSVM-P	FenchelRank	RSRank	TGRank
OHSUMED	0.8	0.2889	0.5156	**0.2178**
HP2004	1.0	**0.1875**	0.7312	0.7343
TD2004	1.0	**0.5**	0.8187	0.7636
NP2004	1.0	**0.2906**	0.8437	0.8
MQ2008	0.8697	**0.3435**	0.5261	0.3957

TABLE 5
Ranking Accuracies of FenchelRank, RSRank, and TGRank on the LETOR Distribution

	N@1	N@3	N@5	N@10	MAP
OHSUMED					
TGRank	**0.5616**	0.4923	0.4813	0.4524	0.4479
RSRank	0.5395	0.4831	0.4684	0.4503	**0.4486**
FenchelRank	0.5456	**0.5166**	**0.4826**	**0.4637**	**0.4486**
HP2004					
TGRank	0.6000	0.7732	0.7878	0.8025	0.6972
RSRank	0.6267	0.7771	0.7905	0.8115	0.7171
FenchelRank	**0.6667**	**0.7961**	**0.8181**	**0.8274**	**0.7447**
NP2004					
TGRank	0.5600	0.7505	0.7762	0.8033	0.6799
RSRank	**0.5733**	0.7601	**0.7915**	**0.8204**	**0.6922**
FenchelRank	0.5600	**0.7636**	0.7808	0.8157	0.6830
MQ2008					
TGRank	0.3626	0.4303	0.4712	0.2277	0.4709
RSRank	0.3686	0.4340	0.4734	0.2282	0.4781
FenchelRank	**0.3762**	**0.4402**	**0.4790**	**0.2317**	**0.4785**

5. Note that we could not reproduce the results on OHSUMED reported in [10], possibly due to the algorithm chooses the initial model from a set of random initialized models by cross validation.

TABLE 6
Training Time (Seconds) of FenchelRank versus TGRank on HP2004 and NP2004

	FenchelRank			TGRank		
HP2004						
	r	seconds	ratio	C	seconds	ratio
Fold1	22.45	13.12	0.1406	0.1	31.01	0.6563
Fold2	19.56	3.82	0.1406	0.1	29.23	0.6406
Fold3	18.94	4.38	0.1563	0.1	33.18	0.6094
Fold4	20.23	11.42	0.1406	0.1	25.73	0.5938
Fold5	22.29	14.40	0.1250	0.1	26.30	0.6875
NP2004						
Fold1	16.19	12.75	0.1875	0.1	24.70	0.6875
Fold2	15.30	8.83	0.1719	0.1	28.72	0.5938
Fold3	18.36	16.44	0.1719	0.1	31.28	0.6875
Fold4	20.10	17.69	0.2031	0.1	29.08	0.7031
Fold5	18.56	11.79	0.1406	0.1	29.03	0.5781

obtains the lowest sparsity ratios on the four data sets. This indicates that FenchelRank tends to obtain sparser ranking models than other sparse learning-to-rank algorithms, which may be the underlying reason that FenchelRank outperforms the baseline algorithms.

To investigate the efficiency of FenchelRank, we also provide the results of training time and the corresponding sparsity ratios of FenchelRank over TGrank on two data sets. The training time depends on the values of the regularization parameter (i.e., C in TGRank or r in FenchelRank). To make fair comparison, we take the following comparing procedure: On each fold, we use TGRank to learn a ranking model $\boldsymbol{w}_{TG}$ by fixing the regularization parameter $C = 0.1$ and the maximum number of iterations $T = 1{,}000$. Let $\boldsymbol{w}^\star$ be the best model (i.e., the Bayes error is minimized) for the objective of TGRank (2). As explained in Section 4 (see the detailed explanation in [32, Section 1.2]), for any C in the objective of TGRank (2), there exists a corresponding $r = C|\boldsymbol{w}^\star|_1$ such that the objective in (2) is equivalent to the objective of FenchelRank (3). Since $\boldsymbol{w}^\star$ is unknown, we approximately set $r = C|\boldsymbol{w}_{TG}|_1$ and set the maximum number of iterations $T = 1{,}000$ in FenchelRank, and, thus, learn a ranking model $\boldsymbol{w}_{Fenchel}$. Then, we compare the training time of these two algorithms under this parameter setting. Table 6 shows the results of training time and the corresponding sparsity ratios of ranking models. The results show that: 1) The training time of FenchelRank is less than that of TGRank. This is due to the fast convergence rate and the early stopping criterion in FenchelRank. 2) The corresponding sparsity ratios of FenchelRank are also significantly smaller than those of TGRank. This indicates that FenchelRank tends to obtain sparser ranking models.

8.5.2 Comparing with a Feature Selection Method

We also implement a simple feature selection algorithm based on RankSVM-Primal, and compare its ranking performance with FenchelRank. The feature selection algorithm belongs to the filter methods and it adopts a two-stage strategy: 1) In the first stage, we sort the documents in a training set, respectively, using each of the given features. The features with high NDCG@10 values are more important for ranking than those with low NDCG@10 values. We construct a new training set by selecting the top k features with high NDCG@10 values and removing other features from the original training set. 2) In the second stage, we use RankSVM-Primal to learn a ranking model on the newly constructed training set containing only k features.

TABLE 7
Ranking Accuracies on OHSUMED and HP2004

	N@1	N@3	N@5	N@10	MAP
OHSUMED					
FS-Rank(k= 5)	0.5228	0.4811	0.4633	0.4494	0.4416
FS-Rank(k=10)	0.5426	0.5018	0.4722	0.4528	0.4486
FS-Rank(k=15)	0.5457	0.5027	0.4778	0.4573	0.4492
FS-Rank(k=20)	0.5457	0.4989	0.4758	0.4573	0.4489
FS-Rank(k=25)	**0.5648**	0.5012	0.4770	0.4542	**0.4511**
RankSVM-Primal	0.5460	0.4855	0.4689	0.4504	0.4446
FenchelRank	0.5456	**0.5166**	**0.4826**	**0.4637**	0.4486
HP2004					
FS-Rank(k= 5)	0.6000	0.7226	0.7737	0.7888	0.6866
FS-Rank(k=10)	0.6133	0.7364	0.7739	0.7891	0.6908
FS-Rank(k=15)	0.6000	0.7613	0.7945	0.8085	0.6979
FS-Rank(k=20)	0.6267	0.7231	0.7529	0.7797	0.6938
FS-Rank(k=25)	0.5867	0.7038	0.7347	0.7658	0.6718
RankSVM-Primal	0.5733	0.7129	0.7528	0.7720	0.6712
FenchelRank	**0.6667**	**0.7961**	**0.8181**	**0.8274**	**0.7447**

The ranking accuracies on OHSUMED and HP2004 are shown in Table 7. Two observations can be made from the results: 1) The simple feature selection algorithm based on RankSVM-Primal performs better than the original RankSVM-Primal algorithm. This validates our intuition that a small number of strong features can dominate the whole ranking performance and learning a sparse model with only a few nonzero coefficients is desired for ranking. 2) The proposed FenchelRank algorithm still shows significant ranking performance gain compared to the feature selection algorithm. This is because FenchelRank is an ℓ_1 optimization algorithm, which minimizes the training error while simultaneously conducts model selection.

8.6 Comparing with Other State-of-the-Art Methods

In the last experiment, we compare FenchelRank with other state-of-the-art learning-ro-rank algorithms. Since our proposed method learns a linear ranking model, we select RankSVM-struct, AdaRank-NDCG, and ListNet as baselines in our experiments because they are state-of-the-art algorithms that also learn linear models. We also select RankBoost as a representative of tree-based algorithm that learns nonlinear ranking models.

The results on five data sets with respect to NDCG and MAP are shown in Table 8. Two observations can be made from the results: 1) While the accuracies of the baseline algorithms vary from one data set to another, the proposed FenchelRank performs very competitively. On OHSUMED, HP2004, and NP2004, FenchelRank performs the best. On HP2004, its value of MAP is 0.7447, compared to 0.6914 of the second best algorithm, which indicates a 7.7 percent increase; its value of NDCG@10 also indicates a 2.7 percent increase compared to the second best algorithm. On OHSUMED, FenchelRank's value of NDCG@10 is 0.4637, compared to 0.4523 of the second best algorithm, which indicates a 2.4 percent increase. On TD2004 and MQ2008, FenchelRank is in the second place among all of the five algorithms. 2) Among the five algorithms in comparison,

TABLE 8
Ranking Accuracies on the Five Data Sets from the LETOR Distribution

	N@1	N@3	N@5	N@10	MAP
OHSUMED					
ListNet	0.5326	0.4732	0.4432	0.4410	0.4457
AdaRank -NDCG	0.5330	0.4790	0.4673	0.4496	**0.4498**
RankSVM -Struct	**0.5515**	0.4850	0.4729	0.4523	0.4478
RankBoost	0.4632	0.4555	0.4494	0.4302	0.4411
FenchelRank	0.5456	**0.5166**	**0.4826**	**0.4637**	0.4486
HP2004					
ListNet	0.6000	0.7213	0.7694	0.7845	0.6899
AdaRank -NDCG	0.5867	0.7512	0.7920	0.8057	0.6914
RankSVM -Struct	0.5867	0.7248	0.7523	0.7666	0.6784
RankBoost	0.5067	0.6989	0.7211	0.7428	0.6251
FenchelRank	**0.6667**	**0.7961**	**0.8181**	**0.8274**	**0.7447**
NP2004					
ListNet	0.5333	0.7587	**0.7965**	0.8128	0.6720
AdaRank -NDCG	0.5067	0.6722	0.7122	0.7384	0.6269
RankSVM -Struct	**0.5600**	0.7321	0.7746	0.7977	0.6771
RankBoost	0.4267	0.6274	0.6512	0.6914	0.5640
FenchelRank	**0.5600**	**0.7636**	0.7808	**0.8157**	**0.6830**
MQ2008					
ListNet	0.3754	0.4324	0.4747	0.2303	0.4775
AdaRank -NDCG	**0.3826**	**0.4420**	**0.4821**	0.2307	**0.4824**
RankSVM -Struct	0.3627	0.4286	0.4695	0.2279	0.4696
RankBoost	0.3856	0.4288	0.4666	0.2255	0.4775
FenchelRank	0.3762	0.4402	0.4790	**0.2317**	0.4785
TD2004					
ListNet	0.3600	0.3573	0.3325	0.3175	0.2231
AdaRank -NDCG	0.4267	0.3688	0.3514	0.3163	0.1936
RankSVM -Struct	0.3467	0.3371	0.3192	0.3090	0.2196
RankBoost	**0.5067**	**0.4295**	**0.3878**	**0.3504**	**0.2614**
FenchelRank	0.3600	0.3725	0.3462	0.3202	0.2368

the proposed FenchelRank algorithm appears to be the most stable algorithm on all of the data sets. For instance, ListNet archives a good performance on NP2004, while it performs poorly on OHSUMED. AdaRank-NDCG performs well on TD2004 and HP2004, but yields a poor performance on OHSUMED and NP2004. In summary, compared with other state-of-the-art algorithms, the proposed FenchelRank algorithm achieves competitive and stable performance on ranking. We further provide the significant tests (t-test) results of FenchelRank versus other algorithms, respectively. The results in Table 9 indicate that FenchelRank makes statistically significant improvement (i.e., p-value less than 0.05) over other state-of-the-art algorithms on OHSUMED and HP2004.

TABLE 9
p-Value on OHSUMED and HP2004

	N@1	N@3	N@5	N@10	MAP
FenchelRank versus ListNet					
OUSUMED	0.3561	**0.0019**	**0.0002**	**0.0023**	0.1675
HP2004	0.0663	**0.0136**	**0.0346**	**0.0409**	**0.0387**
FenchelRank versus AdaRank-NDCG					
OUSUMED	0.3174	**0.0029**	**0.0326**	**0.0076**	0.7050
HP2004	0.0907	0.1040	0.1245	0.1642	0.0650
FenchelRank versus RankBoost					
OUSUMED	**0.0120**	**0.0015**	**0.0167**	**0.0006**	**0.0203**
HP2004	**0.0005**	**0.0070**	**0.0022**	**0.0022**	**0.0002**

9 Conclusion

Sparse learning-to-rank is a relatively new research topic. The difficulty lies in the sparse-inducing norm, i.e., ℓ_1 norm, which is usually nonsmooth and hard to optimize. In this paper, we address this challenge and propose a convergency-provable primal-dual algorithm for sparse learning-to-rank. We prove that, after T iterations, our proposed algorithm can guarantee the obtainment of a solution with a desired tolerant optimization error of $\epsilon = O(\frac{1}{T})$. This convergence rate is better than that of the popular subgradient descent algorithm. Furthermore, we empirically show in our experiments that the proposed algorithm, with the sparse-inducing ℓ_1 norm, can outperform the algorithm using the same loss with the ℓ_2 norm and can achieve state-of-the-art performance on several benchmark data sets.

Acknowledgments

This work was funded in part by National Science Foundation of China (grant No. 61003045, 61003241, and 61033010), Natural Science Foundation of Guangdong Province, China (grant No. 10451027501005667), Educational Commission of Guangdong Province, China, and the Fundamental Research Funds for the Central Universities. The authors thank Dr. Zengya Sun for his help to implement RSRank. Yan Pan is the corresponding author of this paper.

References

[1] S.S. Shwartz and Y. Singer, "On the Equivalence of Weak Learnability and Linear Separability: New Relaxations and Efficient Boosting Algorithms," *Machine Learning J.*, vol. 80, no. 2, pp. 141-163, 2010.

[2] J. Borwein and A. Lewis, *Convex Analysis and Nonlinear Optimization*. Springer, 2006.

[3] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Halmilton, and G. Hullender, "Learning to Rank Using Gradient Descent," *Proc. Int'l Conf. Machine Learning (ICML '05)*, pp. 89-96, 2005.

[4] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li, "Learning to Rank: From Pairwise Approach to Listwise Approach," *Proc. Int'l Conf. Machine Learning (ICML '07)*, pp. 129-136, 2007.

[5] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer, "An Efficient Boosting Algorithm for Combining Preferences," *J. Machine Learning Research*, vol. 4, pp. 933-969, 2003.

[6] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD '02)*, pp. 133-142, 2002.

[7] P. Li, C.J.C. Burges, and Q. Wu, "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting," *Proc. Neural Information Processing System (NIPS '07)*, pp. 845-852, 2007.

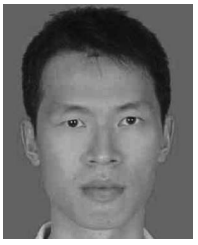
[8] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A Support Vector Method for Optimizing Average Precision," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07)*, pp. 271-278, 2007.

[9] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "SoftRank: Optimising Non-Smooth Rank Metrics," *Proc. Int'l Conf. Web Search and Data Mining (WSDM '08)*, pp. 77-86, 2008.

[10] Z.Y. Sun, T. Qin, J. Wang, and Q. Tao, "Robust Sparse Rank Learning for Non-Smooth Ranking Measures," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '09)*, pp. 259-266, 2009.

[11] D.P. Bertsekas, *Nonlinear Programming,* second ed. Athena Scientific, 1999.
[12] J. Xu and H. Li, "AdaRank: A Boosting Algorithm for Information Retrieval," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07),* pp. 391-398, 2007.
[13] Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon, "Adapting Ranking SVM to Document Retrieval," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06),* pp. 186-193, 2006.
[14] T. Qin, X.D. Zhang, D.S. Wang, W.Y. Xiong, and H. Li, "Ranking with Multiple Hyperplanes," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07),* pp. 279-286, 2007.
[15] V. Vapnik, S. Golowich, and A.J. Smola, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing," *Proc. Ann. Conf. Neural Information Processing Systems (NIPS '97),* pp. 281-287, 1997.
[16] O. Chapelle and S.S. Keerthi, "Efficient Algorithms for Ranking with SVMs," *Information Retrieval J.,* vol. 13, no. 3, pp. 201-215, 2010.
[17] J.I. Marden, *Analyzing and Modeling Rank Data.* Chapman & Hall, 1995.
[18] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise Approach to Learning to Rank: Theory and Algorithm," *Proc. Int'l Conf. Machine Learning (ICML '08),* pp. 1192-1199, 2008.
[19] T. Joachims, "Training Linear SVMs in Linear Time," *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD '06),* pp. 217-226, 2006.
[20] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "LETOR: Benchmark Data Set for Research on Learning to Rank for Information Retrieval," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07),* pp. 129-136, 2007.
[21] R.B. Yates and B.R. Neto, *Modern Information Retrieval.* Addison Wesley, 1999.
[22] W.R. Hersh, C. Buckley, T.J. Leone, and D.H. Hickam, "OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research," *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '94),* pp. 192-201, 1994.
[23] K. Jarvelin and J. Kekalainen, "Cumulated Gain-Based Evaluation of IR Techniques," *ACM Trans. Information Systems,* vol. 20, no. 4, pp. 422-446, 2002.
[24] R.M. Rifkin and R.A. Lippert, "Value Regularization and Fenchel Duality," *J. Machine Learning Research,* vol. 8, pp. 441-479, 2007.
[25] Y. Freund and R.E. Schapire, "A Short Introduction to Boosting," *J. Japanese Soc. for Artificial Intelligence,* vol. 14, no. 5, pp. 771-780, 1999.
[26] T. Joachims, "Making Large-Scale SVM Learning Practical," *Advances in Kernel Methods - Support Vector Learning,* B. Scholkopf, C. Burges, and A. Smola eds., MIT Press, 1999.
[27] G.X. Yuan, K.W. Chang, C.J. Hsieh, and C.J. Lin, "A Comparison of Optimization Methods and Software for Large-Scale ℓ_1-Regularized Linear Classification," *J. Machine Learning Research,* vol. 11, no. 1, pp. 3183-3234, 2010.
[28] P. Tseng and S. Yun, "A Coordinate Gradient Descent Method for Nonsmooth Separable Minimization," *Math. Programming,* vol. 117, nos. 1/2, pp. 387-423, 2009.
[29] J. Duchi, S.S. Shwartz, Y. Singer, and T. Chandra, "Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions," *Proc. Int'l Conf. Machine Learning (ICML '08),* pp. 272-279, 2008.
[30] J. Kim, Y. Kim, and Y. Kim, "A Gradient-Based Optimization Algorithm for LASSO," *J. Computational and Graphical Statistics,* vol. 17, no. 4, pp. 994-1009, 2008.
[31] J. Liu, J. Chen, and J.P. Ye, "Large-Scale Sparse Logistic Regression," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '09),* pp. 547-556, 2009.
[32] D.L. Donoho and Y. Tsaig, "Fast Solution of ℓ_1 Minimization Problems when the Solution May be Sparse," *IEEE Trans. Information Theory,* vol. 54, no. 11, pp. 4789-4812, Nov. 2008.
[33] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with Sparsity-Inducing Penalties," Technical Report HAL 00613125-v2, HAL, 2011.
[34] W. Chen, T.Y. Liu, Y.Y. Lan, Z.M. Ma, and H. Li, "Ranking Measures and Loss Functions in Learning to Rank," *Proc. Advances in Neural Information Processing Systems (NIPS '09),* pp. 315-323, 2009.

Hanjiang Lai received the BS degree from Sun Yat-sen University in 2009, and is currently working toward the PhD degree in the School of Information Science and Technology, Sun Yat-sen University. His main research interests include machine learning algorithms and learning to rank.



Yan Pan received the BS and PhD degrees in computer science from Sun Yat-sen University in 2002 and 2007, respectively. He is currently an assistant professor at Sun Yat-sen University. His research interests include machine learning algorithms, learning to rank, and computer vision.

Cong Liu received the BS degree in microelectronics from South China University of Technology in 2002, the MS degree in computer software and theory from Sun Yat-sen University, Guangzhou, China, in 2005, and the PhD degree in the Department of Computer Science and Engineering, Florida Atlantic University. He is currently an assistant professor at Sun Yat-sen University. His main research interests include routing in mobile ad hoc networks and delay-tolerant networks.

Liang Lin received the BS and PhD degrees from Beijing Institute of Technology in 1999 and 2008, respectively. He studied in the Department of Statistics at the University of California, Los Angeles (UCLA), as a visiting scholar during 2006-2007. He was a postdoctoral research fellow at the Center for Image and Vision Science at UCLA. He is currently an associate professor at Sun Yat-sen University. His research interests include object recognition, graph and shape matching, image parsing, and visual tracking.

Jie Wu is the chair and a professor in the Department of Computer and Information Sciences, Temple University. Prior to joining Temple University, he was a program director at US National Science Foundation. His research interests include wireless networks and mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He has published more than 550 papers in various journals and conference proceedings. He serves in the editorial board of the *IEEE Transactions on Computers*, *Journal of Parallel and Distributed Computing*, *IEEE Transactions on Mobile Computing*. He was also a general cochair for IEEE MASS 2006, IEEE IPDPS 2008, and DCOSS 2009. He is a program cochair for IEEE INFOCOM 2011. He has served as an IEEE Computer Society distinguished visitor. Currently, he is the chairman of the IEEE Technical Committee on Distributed Processing (TCDP) and an ACM distinguished speaker. He is a fellow of the IEEE.