FSMRank: Feature Selection Algorithm for Learning to Rank

Han-Jiang Lai, Yan Pan, Yong Tang, and Rong Yu

***Abstract*— In recent years, there has been growing interest in learning to rank. The introduction of feature selection into different learning problems has been proven effective. These facts motivate us to investigate the problem of feature selection for learning to rank. We propose a joint convex optimization formulation which minimizes ranking errors while simultaneously conducting feature selection. This optimization formulation provides a flexible framework in which we can easily incorporate various importance measures and similarity measures of the features. To solve this optimization problem, we use the Nesterov's approach to derive an accelerated gradient algorithm with a fast convergence rate $O(1/T^2)$. We further develop a generalization bound for the proposed optimization problem using the Rademacher complexities. Extensive experimental evaluations are conducted on the public LETOR benchmark datasets. The results demonstrate that the proposed method shows: 1) significant ranking performance gain compared to several feature selection baselines for ranking, and 2) very competitive performance compared to several state-of-the-art learning-to-rank algorithms.**

***Index Terms*— Accelerated gradient algorithm, feature selection, generalization bound, learning to rank.**

I. Introduction

RANKING is a central task in many information retrieval systems, such as web search engines, recommendation systems, and advertisement systems. In recent years, there has been growing interest in learning to rank, which is a task that applies machine learning techniques to learn good ranking predictors. Many learning-to-rank algorithms have been proposed [1]–[7], [35]. There are two prime functions of ranking: 1) to deliver highly relevant search results, and 2) to be fast in ranking results.

Feature selection has emerged as a successful mechanism in many machine learning applications including computer vision, signal processing, and bioinformatics. Feature selection is also desirable for learning to rank. First, as the number of useful features for ranking are continuously growing, the time of extracting such high-dimensional features has become a bottleneck in ranking. Second, high-dimensional features may be redundant or noisy, which results in poor generalization performance. Lastly, a ranking model with only a small set of features has less computational cost in prediction.

Recently, considerable efforts have been made on feature selection for ranking. One representative work is [8], where the authors incorporated the features' importance and similarity information into ranking and proposed a two-stage method: 1) in the first stage, they proposed an efficient greedy algorithm to select a subset of features with maximum total importance scores and minimum total similarity scores; and 2) in the second stage, they learned a ranking model on the selected features using an off-the-shelf learning-to-rank algorithm (i.e., RankSVM [10] or RankNet [24]). Pan *et al.* [33] explored both greedy and randomized feature selection methods for ranking by using boosted trees to learn ranking models from the selected features. Dang *et al.* [36] proposed a feature selection method that incrementally partitions features into subsets, and then combines the features within each subset to a single feature by maximizing some ranking metric. Li *et al.* [31] proposed a sort–merge method for feature selection, where the features are partitioned into individual subsets, these subsets are sorted by their ranking performance, and then the neighbor subsets in the sorted list are iteratively merged toward a small-cardinality and accurate feature set. Silva *et al.* [32] proposed to select features for image retrieval via a genetic algorithm approach.

All these methods adopt a two-stage paradigm in common: 1) select a feature subset from the original features and 2) learn a ranking model from the selected feature set. One limitation of this two-stage paradigm is that the selected features in the first stage are not necessarily optimal for learning the ranking predictors in the second stage. In addition, in the feature selection stage (the first stage) of some existing methods, directly selecting features may lead to $0-1$ optimization problems (e.g., [8, eq. (2)]), which is known to be nonconvex and hard to analyze.

In this paper, we develop a new feature selection method for ranking. Our formulation is a joint convex optimization problem which minimizes ranking errors while simultaneously conducting feature selection. This optimization problem is a general framework in which we can easily incorporate various importance measures and similarity measures of the features. Moreover, the convexity of this optimization problem

Manuscript received June 25, 2012; accepted February 12, 2013. Date of publication March 8, 2013; date of current version April 5, 2013. This work was supported in part by the National Science Foundation of China under Grant 61003045, Grant 61033010, and Grant 61272067, the Natural Science Foundation of Guangdong Province, China under Grant 10451027501005667 and Grant S2012030006242, the Educational Commission of Guangdong Province, China, Shenzhen Special Development Fund for Emerging Strategic Industries under Grant JCYJ20120615151826623, and the Fundamental Research Funds for the Central Universities. *(Corresponding author: Y. Pan.)*

H.-J. Lai is with the School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006, China (e-mail: lyfflhj@126.com).

Y. Pan is with the School of Software, Sun Yat-Sen University, Guangzhou 510006, China (e-mail: panyan5@mail.sysu.edu.cn).

Y. Tang is with the School of Computer Science, South China Normal University, Guangzhou 510006, China (e-mail: ytang@scnu.edu.cn).

R. Yu is with the Institute of Intelligent Information Processing, Guangdong University of Technology, Guangzhou 510006, China (e-mail: yurong@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2247628

results in not only efficient algorithms but also comprehensive theoretical results. To solve this optimization problem, we use the Nesterov's approach to derive an accelerated gradient algorithm. After at most T iterations, the proposed algorithm can guarantee an ϵ-accurate solution where $\epsilon = O(1/T^2)$. We provide a generalization bound for the proposed optimization problem using the Rademacher complexities. Furthermore, we empirically show that the proposed method significantly outperforms several feature selection baselines and state-of-the-art learning-to-rank algorithms on the public LETOR datasets.

The remainder of this paper is organized as follows. In Section II, we give a brief review of the related work. In Section III, we investigate a learning-to-rank framework which minimizes ranking errors while simultaneously conducts feature selection, whereas we provide an efficient accelerated gradient algorithm and a generalization bound for the corresponding joint convex optimization problem. In Section IV, we conduct various experiments to evaluate the performance of the proposed method on several public benchmark datasets for learning to rank. Finally, we conclude this paper with a short discussion in Section V.

II. Related Work

A. Learning-to-Rank Methods

Recently, three classes of methods for learning to rank have been explored: pointwise methods [1], [9], [24], pairwise methods [2], [3], [10], [35], and listwise methods [4]–[7].

Pointwise methods take a single document as a sample and its loss function is based on individual documents. McRank [1] is a representative algorithm in this class. It approximates the ranking problem by a multiclass classification problem using individual documents as training samples.

Pairwise methods cast the learning-to-rank problem into a task to classify the preference order within document pairs, where a binary classifier is learned to tell which document should be ranked first within a document pair. Example algorithms include RankSVM, RankBoost, and RankNet. The ranking support vector machine (Ranking SVM) [10] algorithm adopts a large margin optimization approach like the traditional SVM [11], and minimizes the number of incorrectly ordered document pairs. Recently, Chapelle and Keerthi [2] 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. RankBoost [3] is a boosting algorithm for ranking by using pairwise preference data. RankNet [24] is another representative algorithm, which uses neural networks for ranking and adopts cross-entropy surrogate as its loss function. The proposed method in this paper belongs to this class.

The listwise methods take the entire group of documents within a query as a sample. They can be categorized into two streams. The first stream optimizes a loss function directly based on IR performance measures [i.e., mean average precision (MAP) or normalized discounted cumulative gain (NDCG)]. Example algorithms include SVM-MAP [4] and AdaRank [5]. The second stream defines listwise loss/performance functions, which take the list of retrieved documents within the same query as an example. Example algorithms include ListNet [6] and ListMLE [7].

B. Feature Selection Methods for Ranking

While many learning-to-rank algorithms have been proposed, much less effort has been made on feature selection methods for ranking. Geng *et al.* [8] proposed a greedy algorithm to select a subset of features with maximum total importance scores and minimum total similarity scores, and thus used those selected features to training a ranking model. Pan *et al.* [33] compared and evaluated different feature selection methods for ranking via using boosted trees to learn ranking predictors from the selected features. Dang *et al.* [36] proposed a feature selection method that incrementally partitions features into subsets via a best-first search, and then combines the features within each subset to a single feature by maximizing some ranking metric via coordinate descent. Li *et al.* [31] proposed a sort–merge method that first partitions the features into individual subsets, sorts the subsets by their ranking performance, and then iteratively merges the neighbor subsets in the sorted list to generate a small-cardinality and accurate feature set. Silva *et al.* [32] proposed a genetic algorithm approach for feature selection in image retrieval.

All these methods adopt a two-stage paradigm in common: select a subset of features via a feature selection method, and then learn a ranking predictor from the selected features. The limitations of these methods are as follows: 1) the two separated stages cannot guarantee that the selected features in the first stage are optimal for learning the ranking predictors in the second stage, and 2) in the feature selection stage (the first stage) of some existing methods, directly selecting features usually leads to 0–1 optimization problems (e.g. [8, eq. (2)]), which is known to be nonconvex and hard to analyze.

There are two main differences between these methods and the proposed method in this paper: 1) in contrast to the two-stage paradigm, our method solves a joint optimization problem which minimizes the ranking errors and simultaneously conducting feature selection, and 2) the optimization objective in the proposed method is convex, which results in not only highly efficient learning algorithms but also comprehensive theoretical results on the convergence rate and the generalization ability.

Since the pairwise methods reduce learning to rank into a classification problem, in principle, many feature selection algorithms [12], [13] for classification can be directly applied to ranking. However, there are significant differences between classification and ranking, such as in their evaluation measures.

Note that, instead of selecting effective features for ranking, an interesting but orthogonal approach is selecting the most informative examples/pairs in the training data by active learning [34].

III. Our Method

A. Optimization Formulation

First, we introduce the notations used in learning to rank. In the problem of learning to rank, there is a

labeled training set $S = \{(q_k, \hat{X}^{(q_k)}, Y^{(q_k)})\}_{k=1}^{n}$ and a test set $T = \{(q_k, \hat{X}^{(q_k)})\}_{k=n+1}^{n+u}$. Here q_k denotes a query, $\hat{X}^{(q_k)} = \{\hat{X}_i^{(q_k)}\}_{i=1}^{n(q_k)}$ denotes the list of corresponding retrieved objects for q_k, and $Y^{(q_k)} = \{y_i^{(q_k)}\}_{i=1}^{n(q_k)}$ is the list of corresponding relevance labels provided by human, $n(q_k)$ represents the number of objects in the retrieved object list belongs to query q_k, and $\hat{X}_i^{(q_k)}$ represents that the ith object in the retrieved object list belongs to query q_k. Each $\hat{X}_i^{(q_k)} \in \mathbb{R}^d$ is a feature vector, and each attribute of $\hat{X}_i^{(q_k)}$ is scaled to [0, 1]. For any vector $\boldsymbol{x}$, we denote $||x||_1 = \sum_i |\boldsymbol{x}_i|$, $||x||_2 = \sqrt{\sum_i \boldsymbol{x}_i^2}$, $||x||_\infty = \max_i |\boldsymbol{x}_i|$ as the ℓ_1 norm, ℓ_2 norm, and ℓ_∞ norm of $\boldsymbol{x}$, respectively.

Now we formally present our feature selection problem for ranking. We formulate our task as the following joint optimization problem:

$$\min_{\hat{\boldsymbol{w}}} \frac{\lambda_1}{2}\hat{\boldsymbol{w}}^T \hat{A}\hat{\boldsymbol{w}} + \lambda_2||\hat{\boldsymbol{w}} \oslash \hat{\boldsymbol{s}}||_1 + f(\hat{\boldsymbol{w}}, (\hat{X}^{(q_k)}, Y^{(q_k)})_{k=1}^{n}) \quad (1)$$

where we denote $\hat{A}$ as the $d \times d$ similarity matrix in which $\hat{A}_{i,j} = \hat{A}_{j,i} \in [0, 1]$ represents the similarity score between the ith feature and the jth feature, and $\oslash$ as the element-wise division operator $||\hat{\boldsymbol{w}} \oslash \hat{\boldsymbol{s}}||_1 = \sum_i |\hat{\boldsymbol{w}}_i/\hat{\boldsymbol{s}}_i|$. And $\hat{\boldsymbol{s}}_i \in (0, 1]$ represents the importance score of the ith feature.

In this optimization problem, the parameters λ_1 and λ_2 control the tradeoff[1] among the following three terms.

1) The first term is a penalty on redundancy of the large weighted features. That is, since $\hat{\boldsymbol{w}}^T \hat{A}\hat{\boldsymbol{w}} = \sum_{i,j} \hat{\boldsymbol{w}}_i \hat{A}_{i,j} \hat{\boldsymbol{w}}_j$, two features i and j $(i \neq j)$ with a large similarity score $\hat{A}_{i,j}$ are penalized not to simultaneously have large weights $\hat{\boldsymbol{w}}_i$ and $\hat{\boldsymbol{w}}_j$.
2) The second term is a weighted ℓ_1 norm that encourages the Selection of the effective features and removal of the noisy features. That is, a feature i with a small importance score $\hat{\boldsymbol{s}}_i$ is penalized not to have a large weight $\hat{\boldsymbol{w}}_i$.
3) The last term represents the loss function modeling ranking errors. For simplicity, we denote $f(\hat{\boldsymbol{w}}, (\hat{X}^{(q_k)}, Y^{(q_k)})_{k=1}^{n})$ as $f(\hat{\boldsymbol{w}})$. The loss function $f(\hat{\boldsymbol{w}})$ can be any convex function with L-Lipschitz gradient: for all vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f)$, we have $f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \langle f'(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + L/2||\boldsymbol{x} - \boldsymbol{y}||_2^2$, where L is the Lipschitz constant. In this paper, we use the squared hinge loss $f(w) = 1/p \sum_{((X_i^{(q_k)}, Y_i^{(q_k)}), (X_j^{(q_k)}, Y_j^{(q_k)})) \in P} \max(0, 1 - y_{i,j}^{(q_k)} \langle \boldsymbol{w}, X_i^{(q_k)} - X_j^{(q_k)}\rangle)^2$, where P is a set of comparable object pairs. $((X_i^{(q_k)}, Y_i^{(q_k)}), (X_j^{(q_k)}, Y_j^{(q_k)})) \in P$ if and only if $(X_i^{(q_k)}, Y_i^{(q_k)})$, $(X_j^{(q_k)}, Y_j^{(q_k)})$ belong to the same query q_k and $Y_i^{(q_k)} \neq Y_j^{(q_k)}$. $y_{i,j}^{(q_k)} = 1$ if $Y_i^{(q_k)} > Y_j^{(q_k)}$, otherwise $y_{i,j}^{(q_k)} = -1$. p denotes the number of pairs in P.

The optimization problem in (1) has the following advantages: 1) it provides a flexible framework to incorporate various importance and similarity information of features, by assigning the importance vector $\hat{\boldsymbol{s}}$ and the similarity matrix $\hat{A}$, respectively, and 2) it is a convex problem that minimizes the ranking errors while simultaneously conducting feature selection. As a result, it has not only highly efficient learning algorithms but also comprehensive theoretical results on the convergence rate and the generalization bound.

The optimization in (1) has the following equivalent form:

$$\min_{\boldsymbol{w}:\ \boldsymbol{w} \in R_+^{2d}} \psi(\boldsymbol{w}) = \frac{\lambda_1}{2}\boldsymbol{w}^T A\boldsymbol{w} + \lambda_2 \sum_{i=1}^{2d} \frac{\boldsymbol{w}_i}{\boldsymbol{s}_i} + f(\boldsymbol{w}, (X^{(q_k)}, Y^{(q_k)})_{k=1}^{n}) \quad (2)$$

where R_+^{2d} denotes a $2d$-dimensional nonnegative space. If a vector $v = (v_1, v_2, \ldots, v_{2d})$ belongs to R_+^{2d}, we have $v_i \geq 0$ for all $i \in \{1, 2, \ldots, 2d\}$. In the rest of this paper, we denote $f(\boldsymbol{w}, (X^{(q_k)}, Y^{(q_k)})_{k=1}^{n})$ as $f(\boldsymbol{w})$ for short. Note that we extend the dimension of the object feature vectors from d to $2d$, i.e., $X^{(q_k)} = [\hat{X}^{(q_k)}; -\hat{X}^{(q_k)}]$ $(k = 1, 2, \ldots, n)$. A is the corresponding $2d \times 2d$ similarity matrix where $A_{i,j}$ represents the similarity between the ith and the jth features on the new $2d$-dimensional feature space. And $\boldsymbol{s}$ is the corresponding importance vector defined on the new $2d$-dimensional feature space. We can verify that, if $\boldsymbol{w}^* \in \mathbb{R}^{2d}$ is an ϵ-accurate solution of (2), then $\hat{\boldsymbol{w}}^* \in \mathbb{R}^d$ defined by $\hat{\boldsymbol{w}}_i^* = \boldsymbol{w}_i^* - \boldsymbol{w}_{d+i}^* (1 \leq i \leq d)$ is also an ϵ-accurate solution of (1). To sum up, if we find an efficient algorithm to approximate the solution of (2), then the output of the algorithm can be easily translated to an approximate solution of (1). Therefore we will focus on solving (2).

After getting a solution $\boldsymbol{w}$ by solving (2), one can use $\boldsymbol{w}$ as a ranking predictor on the test data. Concretely, for $X^{(q_k)} = \{X_1^{(q_k)}, X_2^{(q_k)}, \ldots, X_{n(q_k)}^{(q_k)}\}$ within a query q_k, one can use $\boldsymbol{w}$ to generate a list of scores $\{\boldsymbol{w}^T X_1^{(q_k)}, \boldsymbol{w}^T X_2^{(q_k)}, \ldots, \boldsymbol{w}^T X_{n(q_k)}^{(q_k)}\}$, and then obtain the final rank results by sorting these scores.

In the rest of this section, we will present the importance measures and the similarity measures used to calculate $\boldsymbol{s}$ and A, respectively. After that, we will present an efficient algorithm to solve the optimization problem in (2), with a very fast convergence rate of $O(1/T^2)$.

B. Importance and Similarity Measures

In principle, various importance and similarity measures can be used in our optimization framework, such as the measures used in [8]. For simplicity, we use the Pearson correlation coefficient [14] to calculate both the importance of features and the similarity between features. Pearson correlation coefficient is simple and has been shown to be effective in many feature selection methods [12]. The results in our experiments indicate that the Pearson correlation coefficient is also effective in learning to rank.

Next, we define the Pearson correlation coefficient in the ranking problem. Let $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ be the ith and the jth

[1]We can control each term's contribution to the optimization objective (1) by different combinations of parameters. i.e., if we use a λ_1 that satisfies $\lambda_1/2 \gg \lambda_2$ and $\lambda_1/2 \gg 1$, then the optimization objective (1) will obtain a model $\hat{\boldsymbol{w}}$ that fits the similarity information (in the first term) well. Similarly, if $\lambda_2 \gg \lambda_1/2$ and $\lambda_2 \gg 1$, then (1) will obtain a model $\hat{\boldsymbol{w}}$ that fits the importance information (in the second term) well. If $1 \gg \lambda_1/2$ and $1 \gg \lambda_2$, then (1) will obtain a model $\hat{\boldsymbol{w}}$ that has small errors on the training data (in the third term).

feature columns in $\{X^{(q_k)}\}_{k=1}^{n}$ $(1 \le i, j \le 2d)$. The Pearson correlation coefficient of $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ is defined as $PC(\boldsymbol{h}_i, \boldsymbol{h}_j) = (\mathrm{cov}(\boldsymbol{h}_\mathrm{i}, \boldsymbol{h}_\mathrm{j}))/(\sqrt{\mathrm{var}(\boldsymbol{h}_\mathrm{i})\mathrm{var}(\boldsymbol{h}_\mathrm{j})})$. cov represents the covariance of two feature columns: i.e., $\mathrm{cov}(\boldsymbol{h}_\mathrm{i}, \boldsymbol{h}_\mathrm{j}) = \sum_{k=1}^{N}(\boldsymbol{h}_\mathrm{i}^{(\mathrm{k})} - \overline{\boldsymbol{h}}_\mathrm{i})(\boldsymbol{h}_\mathrm{j}^{(k)} - \overline{\boldsymbol{h}}_\mathrm{j})$. var represents the variance in a feature column: i.e., $\mathrm{var}(\boldsymbol{h}_i) = \sum_{k=1}^{N}(\boldsymbol{h}_i^{(k)} - \overline{\boldsymbol{h}}_i)^2$. $\boldsymbol{N}$ is the number of documents in the training set, $h_i^{(k)}$ is the kth element in $\boldsymbol{h}_i$, and $\overline{\boldsymbol{h}}_i$ is the mean of all the elements in $\boldsymbol{h}_i$.

For the importance of the features, we define $\boldsymbol{s}_i = |PC(\boldsymbol{h}_i, \boldsymbol{Y})|$ where $\boldsymbol{Y}$ represents the columns of relevance labels with respect to the documents in the training set.

For the similarity between two features, we define the similarity matrix A as $A_{i,j} = |PC(\boldsymbol{h}_i, \boldsymbol{h}_j)|$.

C. Algorithm

We propose to employ Nesterov's method [15] to solve the optimization problem in (2). Nesterov [15] proposed an accelerated gradient method for the smooth optimization problems, which has the optimal convergence rate for a black-box model. Subsequent work [16] extended this method to solve the composite optimization problems of the form $\min_{\boldsymbol{w}} \ell(\boldsymbol{w}) + r(\boldsymbol{w})$, where $\ell(\boldsymbol{w})$ is a convex function with Lipschitz-continuous gradient, and $r(\boldsymbol{w})$ is simple, convex, but nonsmooth. Nesterov's method has a much faster convergence rate, i.e., $O(1/T^2)$, than the traditional gradient methods such as subgradient descent ($O(1/\sqrt{T})$) and gradient descent ($O(1/T)$).

First, we reformulate the optimization problem in (2) to the form of $\min_{\boldsymbol{w}} \ell(\boldsymbol{w}) + r(\boldsymbol{w})$. For simplicity, we denote $f(\boldsymbol{w}, (X^{(q_k)}, Y^{(q_k)})_{k=1}^{n})$ as $f(\boldsymbol{w})$. Let $r(\boldsymbol{w}) = \lambda_2 \sum_i \boldsymbol{w}_i/\boldsymbol{s}_i$ and $\ell(\boldsymbol{w}) = \lambda_1/2\boldsymbol{w}^T A\boldsymbol{w} + f(\boldsymbol{w})$. It is easy to verify that $r(\boldsymbol{w})$ is a simple convex function and $\ell(\boldsymbol{w})$ is convex with Lipschitz-continuous gradient.

Next, we present our accelerated gradient algorithm to solve (2). Different from the traditional gradient methods which perform descent by simply using the (sub) gradient, accelerated gradient methods need to solve the following optimization problem (known as the projection step) in each iteration:

$$\arg \min_{\boldsymbol{w}_\mathrm{t}:\boldsymbol{w}_\mathrm{t}\in \mathrm{R}_+^{2\mathrm{d}}} \mathrm{Q}(\boldsymbol{w}_\mathrm{t}, \boldsymbol{z}_\mathrm{t}) = \lambda_2 \sum_{i=1}^{2d} \frac{\boldsymbol{w}_i}{\boldsymbol{s}_i} + \langle \ell^{'}(\boldsymbol{z}_t), \boldsymbol{w}_t - \boldsymbol{z}_t \rangle + \frac{L}{2}||\boldsymbol{w}_t - \boldsymbol{z}_t||^2 \quad (3)$$

where $\boldsymbol{w}_t$ and $\boldsymbol{z}_t$ represent the ranking model and an estimation of the ranking model at iteration t, respectively. The function $Q(\boldsymbol{w}_t, \boldsymbol{z}_t)$ is a combination of the nonsmooth part $r(\boldsymbol{w}_t)$ and a quadratic approximation of the smooth part $\ell(\boldsymbol{w}_t)$.

The proposed feature selection method for ranking (FSMRank) algorithm is shown in Algorithm 1. FSMRank adopts Nesterov's algorithmic framework with Nemirovski's line search scheme [17], which has been applied in several machine learning tasks [18], [19]. FSMRank takes λ_1, λ_2, and the Lipschtiz constant L_0 as its input parameters. λ_1 and λ_2 are chosen by cross-validation. The two critical issues arising here are: 1) how to efficiently and exactly solve (3) in the projection step, and 2) how to set the parameter L_0. We will discuss these issues in the following subsections, respectively.

Algorithm 1 FSMRank

Input: $S = \{X^{(q_k)}, Y^{(q_k)}\}_{k=1}^{n}$, λ_1, λ_2, T and L_0; Output: $\boldsymbol{w}_T$
Initialize: $\boldsymbol{w}_0 = \boldsymbol{z}_1 = \boldsymbol{0}$, $\alpha_1 = 1$, $\gamma = 2$, $L = L_0/\gamma$ [10]
For $t = 1, \ldots, T$
  Let $g = \ell^{'}(z_t)$.
  While (true)
    $\boldsymbol{w}_t = argmin_{\boldsymbol{w}_t:\boldsymbol{w}_t\in R_+^{2d}} \; Q(\boldsymbol{w}_t, \boldsymbol{z}_t)$ (**projection step**)
    If $\ell(\boldsymbol{w}_t) \le \ell(\boldsymbol{z}_t) + \langle \ell^{'}(\boldsymbol{z}_t), \boldsymbol{w}_t - \boldsymbol{z}_t \rangle + \frac{L}{2}||\boldsymbol{w}_t - \boldsymbol{z}_t||^2$
      break;
    End If
    $L = \gamma L$;
  End While
  IF $\frac{|F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t-1})|}{|F(\boldsymbol{w}_{t-1})|} \le \epsilon_s$
    break;
  $\alpha_{t+1} = \frac{1+\sqrt{1+4\alpha_t^2}}{2}$
  $\boldsymbol{z}_{t+1} = \boldsymbol{w}_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(\boldsymbol{w}_t - \boldsymbol{w}_{t-1})$
End For

1) Solving the Projection Step: The optimization problem in (3) has the following closed-form solution:

$$w_t^{(i)} = \begin{cases} 0 & \text{if } Lz_t^{(i)} - g_i - \lambda_2/s_i \le 0 \\ \frac{Lz_t^{(i)} - g_i - \lambda_2/s_i}{L} & \text{otherwise} \end{cases}$$

where $\boldsymbol{g} = \ell^{'}(\boldsymbol{z}_t)$ and $i = 1, \ldots, 2d$. It can be easily derived by taking the derivative of (3) with respect to $\boldsymbol{w}_t$ to zero.

2) Setting the Lipschitz Constant: In the projection step, the Lipschitz constant L of the smooth part $\ell(\boldsymbol{w})$ is not known in advance. To address this issue, in each iteration, we use Nemirovski's line search scheme [17] to find a suitable value of L. The only thing we need to determine is an upper bound of L, denoted by L_0. The parameter L_0 is determined by the following theorem.

Theorem 1: The Lipschitz constant of $\ell(\boldsymbol{w}) = 1/p \sum_{((X_i^{(q_k)}, Y_i^{(q_k)}), (X_j^{(q_k)}, Y_j^{(q_k)}))\in P} \max(0, 1 - y_{i,j}^{(q_k)} \langle \boldsymbol{w}, \boldsymbol{x}_i^{(q_k)} - \boldsymbol{x}_j^{(q_k)} \rangle)^2 + \lambda_1/2\boldsymbol{w}^T A\boldsymbol{w}$ is less than or equal to $4d/p \sum_{i=1}^{p} ||\boldsymbol{x}_i^{(q_k)} - \boldsymbol{x}_j^{(q_k)}||_\infty^2 + \lambda_1 2d \le (4 + 2\lambda_1)d$, where $2d$ is the number of features.

The proof can be found in Appendix I.

To further improve the efficiency of FSMRank, we introduce an early stopping criterion for FSMRank. Concretely, FSMRank stops when either of the following conditions is satisfied: 1) it reaches the maximum iteration number T, or 2) at iteration t, the objective value of (2) with the weight vector $\boldsymbol{w}_t$ is not significantly changed compared to the objective value of (2) with the weight vector $\boldsymbol{w}_{t-1}$. Denote $F(\boldsymbol{w})$ as the objective value of (2) with the weight vector $\boldsymbol{w}$. At each iteration t, FSMRank stops if the corresponding change $|F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t-1}|$ satisfies

$$\frac{|F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t-1})|}{|F(\boldsymbol{w}_{t-1})|} \le \epsilon_s$$

where ϵ_s is a sufficiently small tolerance, i.e., we set $\epsilon_s = 10^{-4}$ in our experiments.

D. Theoretical Analysis

In contrast to the greedy algorithm in [8], the main benefit of our joint optimization problem (2) is its convexity, which results in comprehensive theoretical results, such as the regret bound for the proposed algorithm (Theorem 2) and the generalization bound for our optimization problem (Theorem 3).

Theorem 2 establishes the regret bound for FSMRank.

Theorem 2: Let $\{\boldsymbol{w}_t\}_{t=1}^{T}$ be the sequence generated by Algorithm 1, and $\boldsymbol{w}^*$ be a solution of the problem (2). For any $t \geq 1$ and $\psi(\boldsymbol{w})$ defined in (2), we have

$$\psi(\boldsymbol{w}_t) - \psi(\boldsymbol{w}^*) \leq 2L_0||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2/(t+1)^2.$$

Theorem 2 reveals the convergence rate of FSMRank: after at most T iterations, FSMRank is guaranteed to obtain an ϵ-accurate solution where $\epsilon = O(1/T^2)$. Proofs can be directly derived from the proofs in pp. 163–165 in [17].

The objective in (2) is a novel formulation that encodes both importance and similarity information for feature selection. A natural question is whether there exists a generalization error bound of an output ranking model by solving the objective in (2) (i.e., an output model from Algorithm 1). A generalization error bound of a ranking model is an expected value that measures how far the empirical error (the value of the objective function) on the current training set is from the prediction error (the value of the objective function) on any unseen dataset.[2] In other words, given an output ranking model $\boldsymbol{w}$ from a learning algorithm, if we denote $\mathrm{Err}_{\mathrm{training}}(\boldsymbol{w})$ as the objective value on the current training set, and $\mathrm{Err}_{\mathrm{unseen}}(\boldsymbol{w})$ as the objective value on an unseen dataset (i.e., the test set), then the generalization error bound is an upper bound on $\mathrm{Err}_{\mathrm{unseen}}(\boldsymbol{w}) - \mathrm{Err}_{\mathrm{training}}(\boldsymbol{w})$, which measures the performance of the model w on unseen data.

For ease of analysis, we first rewrite (2) into its equivalent form. It can be verified that, for any λ_2 in the problem in (2), there exists a one-to-one corresponding C such that the problem in (2) is equivalent to the following optimization problem (see the explanation in [38, Sec. 1.2]):

$$\min_{\boldsymbol{w}\in R_+^{2d}, \sum_{i=1}^{2d}\frac{w_i}{s_i}\leq C} \ell(\boldsymbol{w}) = f(\boldsymbol{w}) + \frac{\lambda_1}{2}\boldsymbol{w}^T A\boldsymbol{w}. \tag{4}$$

In the objective in (4), the nonsmooth weighted ℓ_1 term becomes a constraint, in which the parameter C controls the model complexity of $\boldsymbol{w}$ (i.e., a larger C encourages the model to select more features, and a smaller C encourages to select less). Roughly speaking, the combination of the ℓ_2 part with similarity information and the ranking error part can be viewed as a new form of loss function, which is smooth (with a smoothness parameter L_0) and easy to analyze.

Theorem 3 shows the generalization bound for (4).

Theorem 3: Let W be the set of all possible linear ranking models learned by solving the optimization problem in (4). For any $\theta > 0$ and every ranking model $w \in W$, with probability at least $1-\theta$, we have

$$\ell(\boldsymbol{w}) \leq \overline{\ell}(\boldsymbol{w}) + 2L_0\left[2C\sqrt{\frac{2d}{p}} + (C^2 + \frac{\lambda_1}{2})\sqrt{\frac{2log(4d)}{p}}\right] + \left[(1+C)^2 + \frac{\lambda_1}{2}C^2 - 1\right]\sqrt{\frac{log(1/\theta)}{2p}}$$

where $\ell(\boldsymbol{w})$ and $\overline{\ell}(\boldsymbol{w})$ are the prediction error on unseen data and the empirical training error, respectively, $L_0 = (4+2\lambda_1)d$, and p is the (object pair) sample size.

The proof can be found in Appendix II.

By replacing L_0 with $(4+2\lambda_1)d$, Theorem 3 indicates that, for any output ranking model $\boldsymbol{w}$ by solving (4), the generalization bound is $O(d\sqrt{\frac{2d}{p}} + \sqrt{\frac{log(1/\theta)}{2p}})$, i.e., the bound has a complexity of $d^{1.5}$ with respect to the number of features (d), and has a complexity of $p^{-0.5}$ with respect to the number of object pairs in the training set (p).

IV. Experiments

In this section, we evaluate the ranking performance of FSMRank and compare its performance to several state-of-the-art algorithms on the public LETOR [25] benchmark for learning to rank.

A. Datasets and Baselines

We evaluate the performance on the publicly available LETOR [25] benchmark. We use three datasets from LETOR 3.0 (OHSUMED, HP2004, NP2004) and one dataset (MQ2008) from LETOR 4.0. The OHSUMED [26] dataset consists of 106 queries, with 64 features extracted from the online medical information database, MEDLINE. It consists of 348 566 records from 270 medical journals and 16 140 query documents pairs where relevance judgments are made. The HP2004 and NP2004 [29] datasets are from the "GOV" corpus, which is based on a January 2002 crawl of .gov web sites. In these two datasets, there are 64 features in total extracted from the the query-document pairs, covering a wide range including low-level features (i.e., TF-IDF [25]) and high-level features (i.e., BM25 [28] language models [27]). MQ2008 [30] is a large-scale dataset from LETOR 4.0.

For the purpose of cross-validation, each dataset is prefolded into five folds and each fold contains a training/validation/test set, respectively.

RankSVM-Primal [2], RankSVM-Struct [10], RankBoost [3] ListNet [6], and AdaRank-NDCG [5] were selected as baselines in our experiments, because they are the state-of-the-art algorithms. RankSVM-Primal, RankSVM-struct, and RankBoost optimize pairwise loss functions. ListNet and AdaRank-NDCG are listwise methods.

B. Evaluation Measures

To evaluate the performance of ranking models, we use MAP [22] and NDCG [23] as the evaluation measures.

MAP is a standard evaluation measure widely used in information retrieval. MAP is the mean of average precisions over a set of queries. Precision at position j ($P@j$) [22]

[2]We assume that this unseen dataset and the current training set are generated from the same distribution, i.e., for pairwise ranking, we assume the object pairs in both the unseen dataset and the current training set are generated by i.i.d. sampling. The same assumption is used in other work of generalization analysis for ranking [37].

represents the proportion of relevant documents within the top j retrieved documents, which can be calculated by $P@j = N_{\text{pos}}(j)/j$ where $N_{\text{pos}}(j)$ 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

$$\text{Avg}P_i = \sum_{j=1}^{M} P@j \times \text{pos}(j)/N_{\text{pos}}$$

where j is the position, M is the number of retrieved documents, and $\text{pos}(j)$ is an indicator function. If the document at position j is relevant, then $\text{pos}(j)$ is 1, or else $\text{pos}(j)$ 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 in information retrieval. Given a query q_i, DCG score at position m is defined as

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

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 $\text{NDCG}@m = 1/Z_m \text{DCG}@m$ where Z_m is the maximum value of $DCG@m$, which means that the value of NDCG ranges from 0 to 1. In the rest of this paper, we use $N@m$ as the abbreviation of $\text{NDCG}@m$ (which means the NDCG value at position m).

C. Experiment Protocols

To verify the effectiveness of FSMRank, we conduct four experiments.

1) We compare the ranking accuracies of FSMRank to several feature selection algorithms for ranking.
2) We compare the ranking accuracies of FSMRank to several state-of-the-art learning-to-rank algorithms.
3) We test the effects of the importance information of features and the similarity information between features, respectively. This experiment can empirically tell us whether these two kinds of information contribute to the improvement of ranking performance.
4) We evaluate and compare the efficiency of FSMRank to the sub-gradient descent algorithm.

The experimental results show that FSMRank shows significant ranking performance gain compared to both the feature selection baselines and the state-of-the-art learning-to-rank baselines.

In all the experiments, the parameters λ_1 and λ_2 are chosen by cross-validation, and the maximum number of iterations T is fixed to 400.

D. Experimental Results

1) Comparing to the Feature Selection Methods for Ranking: We implement two feature selection methods for ranking and compare their performance to FSMRank. The first method is a variant of GAS-E in [8], where we carefully implement the greedy algorithm GAS as a preprocessor to select t features, and thus use RankSVM-Primal [2] (instead of RankSVM/RankNet used in [8][3]) to learn a ranking model on the selected t features. The second method, referred to as FSMSVM, is a simple feature selection algorithm that adopts a two-stage strategy. In the first stage, we learn a ranking model by FSMRank, and sort the features decreasingly by their weights in the learned model. Then we construct a new training set containing only the top t features with large weights. In the second stage, we use RankSVM-Primal to learn a ranking model on the newly constructed training set.

First, we compare the overall performance of these feature selection methods. In practice, for GAS-E/FSMSVM, we do not know how many features should be selected on a certain dataset. Hence, we determine the number of features to select (i.e., the parameter t in the optimization objective (2) in [8]) by cross-validation.

As shown in Table I, the proposed FSMRank outperforms GAS-E and FSMSVM on all the four datasets from the LETOR benchmark. Here is some statistics. On HP2004, FSMRank obtains 0.8383 of NDCG@10, which indicates a relative improvement of 6.8% over GAS-E and 6.4% over FSMSVM (with p-values ≤ 0.05). On NP2004, FSMRank shows a 4.0% increase over GAS-E, and a 2.7% increase over FSMSVM with respect to NDCG@10 (with p-values ≤ 0.05). On OHSUMED, FSMRank shows a 3.8% increase over FSMSVM, and a 2.5% increase over GAS-E with respect to NDCG@10 (with p-values ≤ 0.05).

Second, for a detailed comparison, we follow the settings in [8] to compare the algorithms' ranking accuracies w.r.t different number of selected features.[4] Different from the baselines, FSMRank cannot directly select a given number of features because it minimizes ranking errors and simultaneously conducts feature selection. Hence, for FSMRank, we learn a ranking model $\boldsymbol{w}$ on all the features, and sort the features decreasingly by their weights in $\boldsymbol{w}$. In this sense, we can easily construct a model containing k $(1 \leq k \leq d)$ features, by simply combining the top k features (using their associated weights in $\boldsymbol{w}$) in the sorted list. Note that all these results are the average values over the five folds in each dataset.

Figs. 1–3 show the comparison results with respect to different number of selected features (the x-axis). Two observations can be made from the results.

1) While the accuracies of the baseline methods vary on different number of selected features, the accuracies of FSMRank are very stable only if the number of the selected features is not very small (i.e., ≥ 10 on OHSUMED, ≥ 20 on HP2004).
2) FSMRank has significant performance gain compared to the baseline methods.

Here is some statistics. On OHSUMED, when the number of selected features is greater than 5, FSMRank has a

[3]We choose RankSVM-Primal as the off-the-shelf ranking algorithm in GAS-E because, like FSMRank, RankSVM-Primal uses the squared hinge loss as its ranking loss.

[4]Note that, in practice, the best number of features to select is usually unknown and should be chosen by cross-validation. Here we show the results with different number of selected features only for a detailed investigation.

TABLE I

RANKING ACCURACIES ON THE FOUR DATASETS FROM THE LETOR DISTRIBUTION. HERE WE USE $N@m$ AS THE ABBREVIATION OF NDCG@m

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10	MAP
OHSUMED											
GAS-E	**0.5547**	0.5032	0.4794	0.4805	0.4720	0.4626	0.4585	0.4570	0.4551	0.4502	0.4475
FSMSVM	0.5492	0.4908	0.4690	0.4703	0.4640	0.4549	0.4508	0.4496	0.4435	0.4447	0.4441
FSMRank	0.5394	**0.5149**	**0.5013**	**0.4916**	**0.4824**	**0.4757**	**0.4706**	**0.4666**	**0.4634**	**0.4613**	**0.4498**
HP2004											
GAS-E	0.6133	0.6933	0.7280	0.7564	0.7679	0.7679	0.7702	0.7747	0.7789	0.7849	0.6925
FSMSVM	**0.6267**	0.7000	0.7136	0.7520	0.7635	0.7635	0.7730	0.7797	0.7839	0.7879	0.7014
FSMRank	0.6133	**0.7933**	**0.8070**	**0.8187**	**0.8187**	**0.8255**	**0.8302**	**0.8383**	**0.8383**	**0.8383**	**0.7205**
NP2004											
GAS-E	**0.5600**	0.7000	0.7236	0.7595	0.7617	0.7720	0.7791	0.7791	0.7917	0.7958	0.6736
FSMSVM	0.5467	0.7133	0.7538	0.7830	0.7830	0.7927	0.7927	0.8060	0.8060	0.8060	0.6748
FSMRank	0.5467	**0.7800**	**0.7784**	**0.7943**	**0.8000**	**0.8072**	**0.8190**	**0.8279**	**0.8279**	**0.8279**	**0.6837**
MQ2008											
GAS-E	0.3601	0.4076	0.4345	**0.4604**	0.4772	0.4893	0.4960	0.4614	0.2247	0.2290	0.4767
FSMSVM	0.3652	0.4009	0.4278	0.4519	0.4701	0.4857	0.4927	0.4575	0.2210	0.2251	0.4744
FSMRank	**0.3686**	**0.4126**	**0.4399**	**0.4604**	**0.4791**	**0.4940**	**0.4994**	**0.4633**	**0.2281**	**0.2327**	**0.4771**

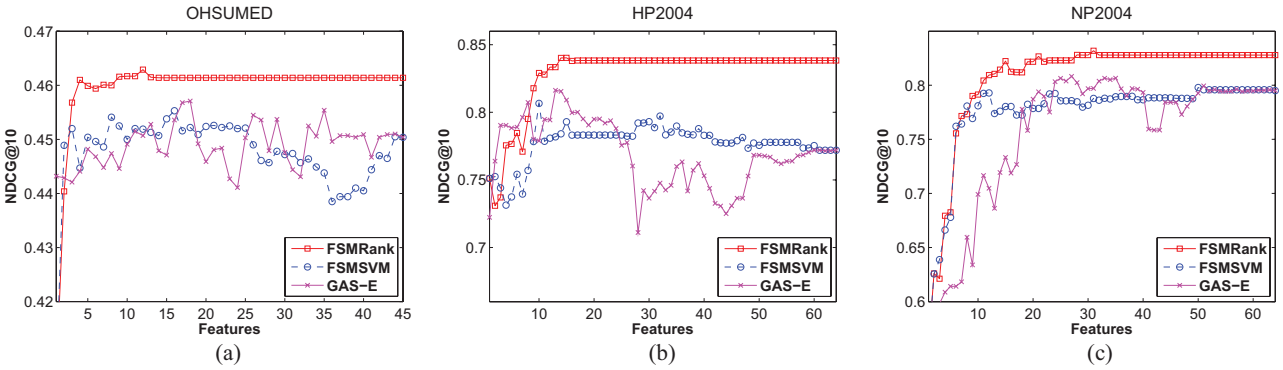


Fig. 1. NDCG@10 results on (a) OHSUMED, (b)HP2004, and (c) NP2004 from LETOR 3.0 distribution.

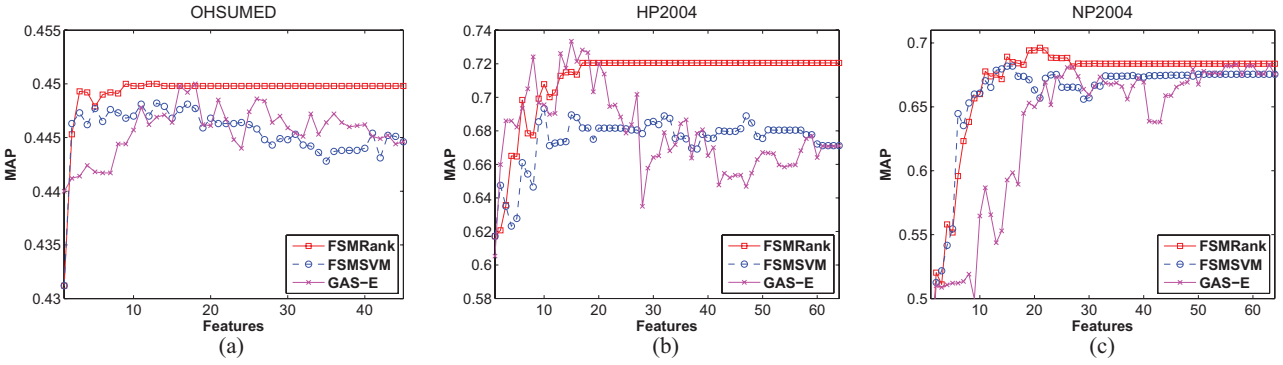


Fig. 2. MAP results on (a) OHSUMED, (b) HP2004, and (c) NP2004 from LETOR 3.0 distribution.

1.3–5.2% increase over FSMSVM, and a 1.1–4.6% increase over GAS-E with respect to NDCG@10. On HP2004, when the number of selected features is greater than 20, FSMRank indicates a 5.1–8.5% increase over FSMSVM, and a 5.4–17.8% increase over GAS-E with respect to NDCG@10. On HP2004, when the number of selected features is greater than 20, FSMRank shows a 4.5–7.6% increase over FSMSVM, and a 2.6–13.4% increase over GAS-E with respect to MAP. The above listed improvements are statistically significant (with the p-values ≤ 0.05). FSMRank is a one-stage method that minimizes the training errors and simultaneously conducts feature selection. This may be the underlying reason why FSMRank outperforms the baselines.

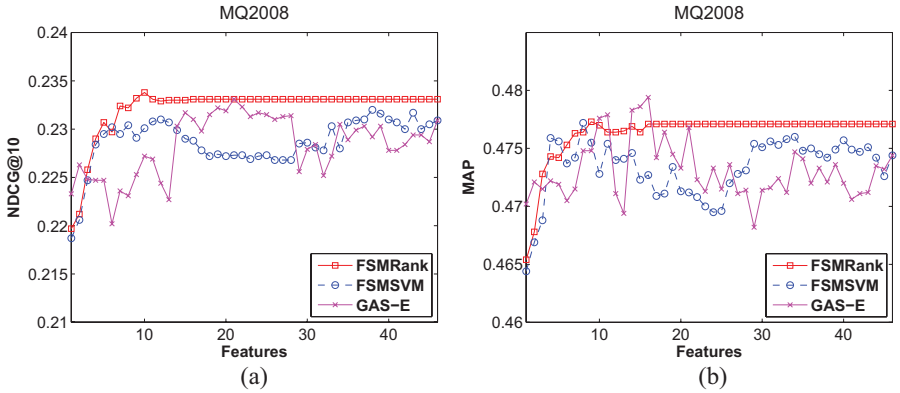


Fig. 3. NDCG@10 and MAP results on (a) and (b) MQ2008 from LETOR 4.0 distribution.

Remark 1: From the results in Figs. 1–3, we can see that FSMRank outperforms the two baselines in most cases. However, with respect to the MAP results on HP2004 [Fig. 2(b)], the GAS-E algorithm shows better performance than FSMRank when the number of selected features is in the range 13–18.

One question arising here is that the results in Fig. 2(b) seems to be inconsistent with the results in Table I, where FSMRank shows better performance over GAS-E on HP2004 with respect to MAP. The main reason which leads to this inconsistency is that the results in Table I and the results in Figs. 1–3 are obtained from different settings. For the results in Table I, the number of features to select (the parameter t) is chosen by cross-validation. It is worth noting that the best value of t chosen from validation set (i.e., the t that has the highest MAP value on the validation set) is not necessarily the one that obtains the highest MAP on the test set. Hence, the results on HP2004 in Table I is not as good as the highest MAP value in the curse of GAS-E in Fig. 2(b).

Remark 2: The results in Figs. 1–3 show that, when the number of selected features is small, the proposed feature selection algorithms have unstable performance. Roughly speaking, when the number of the selected features is small (i.e., less than 10 from the total 64 features), most of the selected features may be effective/relevant features (and few of them are noisy/irrelevant features). For GAS-E and FSMSVM, the weights of the selected features are learned by SVM which uses an ℓ_2 norm. For the proposed FSMRank, the weights of the selected features are learned by solving (2), which has a weighted ℓ_1 norm. Existing studies showed that, on a dataset in which most features are relevant, a model learned via an ℓ_2-regularized formulation usually has better prediction performance than the corresponding model learned via an ℓ_1-regularized formulation (e.g., the explanation in [39, Sec.2.2.2]). This may be the underlying reason why the proposed FSMRank has unstable performance with small number of selected features.

2) Comparing to the State-of-the-Art Algorithms: We compare the ranking accuracies of FSMRank to six state-of-the-art learning-to-rank algorithms, including three pairwise algorithms (RankSVM-Primal, RankSVM-Struct, and RankBoost) and three listwise algorithms (SVMMAP, AdaRank-NDCG, and ListNet)[5].

The comparison results are shown in Table II. Compared to the baseline algorithms, FSMRank shows significant ranking performance gain on OHSUMED, HP2004, and NP2004, and achieves competitive performance on MQ2008. Here is some statistics. On OHSUMED, FSMRank's values of NDCG@3, NDCG@5, and NDCG@10 show a 1.9%–3.2% increase (with p-values $\leq$ 0.05 in t-tests) compared to the second best algorithm. On HP2004, FSMRank's values of NDCG@3, NDCG@5, and NDCG@10 show a 2.1%–7.0% increase (with p-values $\leq$ 0.05 in t-tests) compared to the second best algorithm.

Remark 3: In Table II, FSMRank outperforms other algorithms on OHSUMED, HP2004, and NP2004, but it performs competitive on MQ2008. To investigate what factors might influence the performance of the feature selection algorithms, we take HP2004, NP2004, and MQ2008 as examples and conduct a comparison on their features. Concretely, we sort the documents in each dataset, respectively, using each individual feature out of the 64 given features. The results of each sorted feature's NDCG@10 value are shown in Fig. 4. Two observations can be made from Fig. 4.

1) On NP2004/HP2004, only a few strong features have obviously high NDCG@10 values, while many are low. Moreover, there are some poor (or noisy) features whose NDCG@10 values are far away from the top features.
2) On MQ2008, the NDCG@10 values of many features are close.

And there are less poor features in MQ2008 than in NP2004/HP2004. To sum up, there are many poor (or noisy) features on NP2004/HP2004, and the strong features and the poor features are easy to distinguish. Hence, feature selection can effectively select the strong features and remove the noisy features, and thus lead to good performance on this kind of datasets. On the other hand, on MQ2008, the NDCG@10 values of the features are close, where the effectiveness of feature selection is not so obvious. This may be the underlying reason

[5]The results of these algorithms are cited from the LETOR website at http://research.microsoft.com/en-us/um/beijing/projects/letor/letor3baseline.aspx

TABLE II
RANKING ACCURACIES ON THE THREE DATASETS FROM THE LETOR DISTRIBUTION. HERE WE USE $N@m$ AS THE ABBREVIATION OF NDCG@m

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10	MAP
OHSUMED											
RankSVM-primal	0.5460	0.5010	0.4855	0.4766	0.4689	0.4552	0.4534	0.4500	0.4490	0.4504	0.4446
RankSVM-struct	**0.5515**	0.5000	0.4850	0.4820	0.4729	0.4584	0.4570	0.4587	0.4568	0.4523	0.4478
RankBoost	0.4632	0.4504	0.4555	0.4543	0.4494	0.4439	0.4412	0.4361	0.4326	0.4302	0.4411
ListNet	0.5326	0.4810	0.4732	0.4561	0.4432	0.4400	0.4409	0.4460	0.4459	0.4410	0.4457
SVMMAP	0.5229	0.4909	0.4663	0.4625	0.4516	0.4411	0.4335	0.4332	0.4304	0.4319	0.4453
AdaRank-NDCG	0.5330	0.4922	0.4790	0.4688	0.4673	0.4597	0.4596	0.4575	0.4541	0.4496	**0.4498**
FSMRank	0.5394	**0.5149**	**0.5013**	**0.4916**	**0.4824**	**0.4757**	**0.4706**	**0.4666**	**0.4634**	**0.4613**	**0.4498**
HP2004											
RankSVM-primal	0.5733	0.6867	0.7129	0.7413	0.7528	0.7683	0.7706	0.7706	0.7720	0.7720	0.6712
RankSVM-struct	0.5867	0.6733	0.7248	0.7465	0.7523	0.7652	0.7652	0.7666	0.7666	0.7666	0.6784
RankBoost	0.5067	0.6600	0.6989	0.7039	0.7211	0.7263	0.7263	0.7344	0.7428	0.7428	0.6251
ListNet	0.6000	0.6867	0.7213	0.7618	0.7694	0.7797	0.7845	0.7845	0.7845	0.7845	0.6899
SVMMAP	**0.6267**	0.7400	0.7536	0.7953	0.8011	0.8062	0.8062	0.8062	0.8062	0.8062	0.7176
AdaRank-NDCG	0.5867	0.7333	0.7512	0.7862	0.7920	0.7971	0.7971	0.8016	0.8037	0.8057	0.6914
FSMRank	0.6133	**0.7933**	**0.8070**	**0.8187**	**0.8187**	**0.8255**	**0.8302**	**0.8383**	**0.8383**	**0.8383**	**0.7205**
NP2004											
RankSVM-primal	**0.5600**	0.7000	0.7236	0.7662	0.7719	0.7816	0.7864	0.7908	0.7950	0.7950	0.6755
RankSVM-struct	**0.5600**	0.7000	0.7321	0.7746	0.7746	0.7843	0.7890	0.7935	0.7977	0.7977	0.6771
RankBoost	0.4267	0.5533	0.6274	0.6433	0.6512	0.6667	0.6810	0.6854	0.6854	0.6914	0.5640
ListNet	0.5333	0.7267	0.7587	0.7879	0.7965	0.8037	0.8084	0.8128	0.8128	0.8128	0.6720
SVMMAP	0.5200	0.7000	0.7489	0.7847	0.7869	0.7947	0.7994	0.8039	0.8039	0.8079	0.6620
AdaRank-NDCG	0.5067	0.6133	0.6722	0.7122	0.7122	0.7225	0.7273	0.7384	0.7384	0.7384	0.6269
FSMRank	0.5467	**0.7800**	**0.7784**	**0.7943**	**0.8000**	**0.8072**	**0.8190**	**0.8279**	**0.8279**	**0.8279**	**0.6837**
MQ2008											
RankSVM-primal	0.3725	0.4065	0.4333	0.4566	0.4765	0.4893	0.4940	0.4604	0.2263	0.2309	0.4744
RankSVM-struct	0.3627	0.3985	0.4286	0.4509	0.4695	0.4851	0.4905	0.4564	0.2239	0.2279	0.4696
RankBoost	**0.3856**	0.3993	0.4288	0.4479	0.4666	0.4816	0.4898	0.4568	0.2214	0.2255	0.4775
ListNet	0.3754	0.4112	0.4324	0.4568	0.4747	0.4894	0.4978	0.4630	0.2265	0.2303	0.4775
SVMMAP	0.3516	0.3826	0.4102	0.4308	0.4515	0.4656	0.4758	0.4435	0.2133	0.2179	0.4569
AdaRank-NDCG	0.3826	**0.4211**	**0.4420**	**0.4653**	**0.4821**	**0.4948**	0.4993	**0.4636**	0.2270	0.2307	**0.4824**
FSMRank	0.3686	0.4126	0.4399	0.4604	0.4791	0.4940	**0.4994**	0.4633	**0.2281**	**0.2327**	0.4771

why FSMRank outperforms other algorithms on OHSUMED, HP2004, and NP2004 but performs competitively on MQ2008.

3) Effects of the Importance and Similarity Information: We perform experiments to observe the effects of the importance and similarity information.

To test the effect of the importance information, we first set $\lambda_1 = 0$ in (2) to exclude the similarity information. Denote S^1 as a vector with all 1s, and S^{PC} as a vector with S_i^{PC} being $|\mathrm{PC}(\boldsymbol{h}_i, \boldsymbol{Y})|$. On each dataset, we learn ranking models with $s = S^1$ (without importance) and $s = S^{\mathrm{PC}}$ (with importance), respectively. As shown in Table III, the results indicates that the importance information of the features can help to improve the ranking performance.

Similarly, to test the effect of the similarity information, we set $\lambda_2 = 0$ in (2) to exclude the importance information. Denote I as an identity matrix, and A^{PC} as a matrix with $A_{i,j}^{\mathrm{PC}} = |PC(\boldsymbol{h}_i, \boldsymbol{h}_j)|$. The learned ranking models with $A = A^{\mathrm{PC}}$ (with similarity) do not show significant improvement compared to the learned models with $A = I$ (without similarity).

TABLE III
EFFECT OF IMPORTANCE INFORMATION. HERE WE USE $N@m$ AS THE ABBREVIATION OF NDCG@m

	N@1	N@3	N@5	N@10	MAP
OHSUMED					
s=S^1	0.5486	0.4899	0.4729	0.4523	**0.4471**
s=S^{PC}	**0.5550**	**0.5081**	**0.4835**	**0.4591**	**0.4471**
HP2004					
s=S^1	**0.6133**	0.7796	0.8070	0.8159	**0.7135**
s=S^{PC}	**0.6133**	**0.7870**	**0.8102**	**0.8188**	0.7132
NP2004					
s=S^1	0.5200	0.7454	0.7768	0.8079	0.6603
s=S^{PC}	**0.5600**	**0.7686**	**0.7866**	**0.8113**	**0.6840**

In summary, the weighted ℓ_1 norm with importance information is the main source that leads to the superior performance of FSMRank.

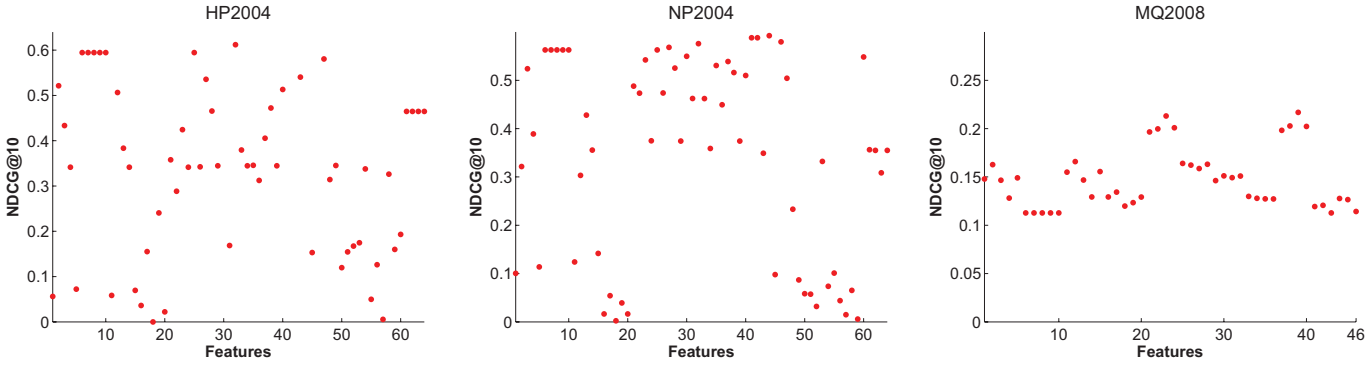


Fig. 4. Comparison of the features in HP2004, NP2004, and MQ2008. The dots are the NDCG@10 values of 64 features on each dataset.

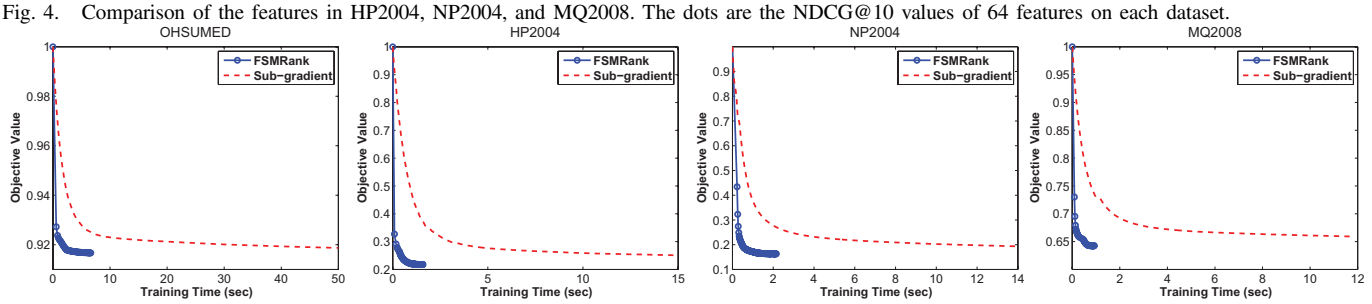


Fig. 5. Comparison results of training time on four datasets from LETOR Distribution.

E. The Convergence Speed of FSMRank

Theorem 2 indicates a fast convergence rate of the proposed FSMRank. To verify the fast convergence speed of FSMRank, we evaluate its training time on LETOR datasets. Note that the optimization objective in (2) is convex but not differentiable everywhere due to the weighted ℓ_1 norm. A popular algorithm to solve convex but nondifferentiable objectives like (2) is subgradient descent, which has a convergence rate of $O(1/\sqrt{T})$ where T is the number of iterations. Hence, we implement a subgradient algorithm for (2) and compare its training time to FSMRank. As shown in Fig. 5, the proposed FSMRank converges consistently faster than the subgradient descent baseline on all datasets. This empirically justifies the superior convergence rate of FSMRank in Theorem 2.

V. Conclusion

In this paper, we developed FSMRank, a new feature selection method for ranking. FSMRank minimizes the ranking errors and simultaneously conducts feature selection. It provides a flexible framework where we can easily incorporate various importance/similarity measures of the features. We derived an accelerated gradient algorithm for FSMRank, with a fast convergence rate $O(\frac{1}{T^2})$. We further provided a generalization bound for the proposed optimization problem using the Rademacher complexities. Extensive evaluation results demonstrated the effectiveness of FSMRank.

Appendix I
Proof of Theorem 1

Proof: Since $\ell(\boldsymbol{w})$ has second-order derivatives, with Lemma 1.2.2 in [15], the Lipschitz constant L is upper bounded by $||\ell''(\boldsymbol{w})||$. Hence, we only need to prove that $||\ell''(\boldsymbol{w})|| \leq 4d + 2\lambda d$.

Let

$$sv = \{((X_i^{(q_k)}, Y_i^{(q_k)}), (X_j^{(q_k)}, Y_j^{(q_k)})), 1 - y_{i,j}^{(q_k)} \\ \langle \boldsymbol{w}, \boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)} \rangle > 0\}$$

be the set of support vectors. The second-order derivative of $\ell(\boldsymbol{w})$ is

$$\ell''(\boldsymbol{w}) = \frac{2}{p} \sum_{\text{sv}} (\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})^T (\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)}) + \lambda A. \quad \text{(I.1)}$$

Proving $||\ell''(\boldsymbol{w})|| \leq M$ is equivalent to proving $\langle \ell''(\boldsymbol{w})\boldsymbol{s}, \boldsymbol{s} \rangle \leq M \sum_{i=1}^{2d} (s_i)^2$ for all $\boldsymbol{s} \in \mathbb{R}^{2d}$, where M is an upper bound constant and $2d$ is the dimension of $\boldsymbol{w}$.

For all $\boldsymbol{s} \in \mathbb{R}^{2d}$ and $\boldsymbol{x} = \boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)}$, we have

$$\langle \boldsymbol{x}^T \boldsymbol{x} \boldsymbol{s}, \boldsymbol{s} \rangle = \left(\sum_{j=1}^{2d} \boldsymbol{s}_j \boldsymbol{x}_j \right)^2 \leq \left(\sum_{j=1}^{2d} ||\boldsymbol{x}||_\infty |\boldsymbol{s}_j| \right)^2 \\ \leq 2d (||\boldsymbol{x}||_\infty)^2 \sum_{j=1}^{2d} (\boldsymbol{s}_j)^2 \quad \text{(I.2)}$$

where the first inequality follows from $\boldsymbol{x}_j \leq ||\boldsymbol{x}||_\infty$, and the second inequality follows from $||s||_1^2 \leq 2d||s||_2^2$ by the Cauchy inequality.

Similarly, for all $\boldsymbol{s} \in \mathbb{R}^{2d}$, we have

$$\langle As, s \rangle = s'As = \sum_{i=1}^{2d} s_i \left(\sum_{j=1}^{2d} (s_j A_{ji}) \right) \\ \leq \sum_{i=1}^{2d} \sum_{j=1}^{2d} |s_i||s_j| = \left(\sum_{i=1}^{2d} |s_i| \right)^2 \leq 2d \sum_{i=1}^{2d} (s_i)^2 \quad \text{(I.3)}$$

where the first inequality follows from the fact $A_{j,i} \in [0, 1]$ for all i, j, and the second inequality follows from $||s||_1^2 \leq 2d||s||_2^2$ by the Cauchy inequality.

Equipped with (I.2) and (I.3), we have

$$\begin{aligned}||\ell''(\boldsymbol{w})|| &= ||\frac{2}{p}\sum_{\text{sv}}(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})^T(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)}) + \lambda A|| \\ &\leq \frac{2}{p}\sum_{\text{sv}}||(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})^T(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})|| + \lambda||A|| \\ &\leq \frac{2}{p}\sum_{P}||(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})^T(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})|| + \lambda||A|| \\ &\leq \frac{4d}{p}\sum_{P}||(\boldsymbol{X}_i^{(q_k)} - \boldsymbol{X}_j^{(q_k)})||_\infty^2 + \lambda 2d \\ &\leq \frac{4d}{p} * p + 2\lambda d = 4d + 2\lambda d. \end{aligned} \tag{I.4}$$

Hence, the Lipschitz constant L of $\ell(\cdot)$ is less than or equal to $4d + 2\lambda d$. ∎

Appendix II
Proof of Theorem 3

The proof of Theorem 3 uses the following lemmas from [20] and [21] (using our notations).

Lemma 1: This is presented in [20, Theorem 3]. Assume a function $\hat{\ell}(\boldsymbol{w}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is Lipschitz (with respect to $\boldsymbol{w}$) with Lipschitz constant L_0 and that $\hat{\ell}$ is bounded by $c_{\hat{\ell}}$. For any $\theta > 0$ and with probability at least $1 - \theta$ simultaneously for all $\boldsymbol{w} \in \mathcal{W}$, we have that

$$\mathcal{L}(\boldsymbol{w}) \leq \overline{\mathcal{L}}(\boldsymbol{w}) + 2L_0\mathcal{R}_p(\mathcal{W}) + c_{\hat{\ell}}\sqrt{\frac{log(1/\theta)}{2p}} \tag{II.5}$$

where $\mathcal{R}_p(\mathcal{W})$ is the Rademacher complexity of a function class $\mathcal{W}$, and p is the sample size. $\mathcal{L}(\boldsymbol{w}) = \mathbb{E}[\hat{\ell}(\boldsymbol{w}^T x, y)]$ is the expectation of $\hat{\ell} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and $\overline{\mathcal{L}}(\boldsymbol{w}) = \frac{1}{p}\sum_{i=1}^p \hat{\ell}(\boldsymbol{w}^T x_i, y_i)$ is the empirical loss.

Lemma 2 (Theorem 9 in [21]): Let $\hat{\ell}(\boldsymbol{w})$ be a β-strongly convex function with respect to a norm $||\cdot||$ on $\mathcal{W}$ such that $\inf_{\boldsymbol{w}=0}\hat{\ell}(\boldsymbol{w}) = 0$. Let $\mathcal{X} = \{x : ||x||_* \leq X_{\max}\}$ and $\mathcal{W} = \{\boldsymbol{w} : \hat{\ell}(\boldsymbol{w}) \leq \hat{\ell}_{\max}\}$, where $||.||_*$ denotes the Fenchel dual norm of $||.||$. Consider the class of linear functions

$$\mathcal{W} = \{x \mapsto \boldsymbol{w}^T x : \boldsymbol{w} \in \mathcal{W}\}.$$

Then we have

$$\mathcal{R}_p(\mathcal{W}) \leq X_{\max}\sqrt{\frac{2\hat{\ell}_{\max}}{\beta p}}.$$

Proof: Let $\hat{\ell}(\boldsymbol{w}) = \ell(\boldsymbol{w}) - 1$, where $\ell(\boldsymbol{w})$ is defined in (4). Obviously, $\inf_{\boldsymbol{w}=0}\hat{\ell}(\boldsymbol{w}) = 0$ for any $x,\ y$.

First, we will prove

$$\hat{\ell}(\boldsymbol{w}) \leq (1 + C)^2 + \frac{\lambda_1}{2}C^2 - 1. \tag{II.6}$$

Since each attribute in any $X_i^{(q)}$ is scaled in $[0, 1]$ (see Section III-A), we have

$$|y_{i,j}^{(q)}(\boldsymbol{w}^T X_i^{(q)} - \boldsymbol{w}^T X_j^{(q)})| = |\boldsymbol{w}^T X_i^{(q)} - \boldsymbol{w}^T X_j^{(q)}| \leq ||\boldsymbol{w}||_\infty.$$

Under the constraints $||\boldsymbol{w} \oslash \boldsymbol{s}||_1 \leq C$ and $\boldsymbol{s}_i \leq 1$ for all i, we have

$$||\boldsymbol{w}||_\infty \leq C. \tag{II.7}$$

Hence, $|y_{i,j}^{(q)}(\boldsymbol{w}^T X_i^{(q)} - \boldsymbol{w}^T X_j^{(q)})| \leq C$. For any object pair $(X_i^{(q)}, X_j^{(q)})$, we have $max(0, 1 - y_{i,j}^{(q)}(\boldsymbol{w}^T X_i^{(q)} - \boldsymbol{w}^T X_j^{(q)}))^2 \leq (1 + C)^2$. Therefore, we have

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{p}\sum_P \max(0, 1 - y_{i,j}^{(q)}(\boldsymbol{w}^T X_i^{(q)} - \boldsymbol{w}^T X_j^{(q)}))^2 \\ &\leq (1 + C)^2 \end{aligned} \tag{II.8}$$

where $f(\boldsymbol{w})$ is defined in (2).

Moreover, since $||\boldsymbol{w}||_\infty \leq C$ (II.7) and $A_{ij} \in [0, 1]$, we have

$$\boldsymbol{w}^T A\boldsymbol{w} \leq C^2. \tag{II.9}$$

With (II.8) and (II.9), we have $\hat{\ell}(\boldsymbol{w}) \leq (1 + C)^2 + \frac{\lambda_1}{2}C^2 - 1$.

Second, we will prove that the Rademacher bound with respect to $\hat{\ell}(\boldsymbol{w})$

$$\mathcal{R}_p(W) \leq \left[2C\sqrt{\frac{2d}{p}} + \left(C^2 + \frac{\lambda_1}{2}\right)\sqrt{\frac{2log(4d)}{p}}\right]. \tag{II.10}$$

We define an object pair matrix $K \in \mathbb{R}^{p\times 2d}$ that each pair $((X_i^{(q_k)}, Y_i^{(q_k)}), (X_j^{(q_k)}, Y_j^{(q_k)})) \in P$ corresponds to a row $K_l = y_{i,j}^{(q_k)}(X_i^{(q_k)} - X_j^{(q_k)})$ in K. We rewrite $\hat{\ell}(\boldsymbol{w})$ as

$$\begin{aligned}\hat{\ell}(\boldsymbol{w}) &= \frac{1}{p}\sum_{l=1}^p max(0, 1 - K_l\boldsymbol{w})^2 + \frac{\lambda_1}{2}\boldsymbol{w}^T A\boldsymbol{w} - 1 \\ &\leq \frac{1}{p}||e - K\boldsymbol{w}||_2^2 + \frac{\lambda_1}{2}\boldsymbol{w}^T A\boldsymbol{w} - 1 \\ &= -\frac{2}{p}\langle\boldsymbol{w}, e^T K\rangle + \frac{1}{p}\langle\boldsymbol{w}\boldsymbol{w}^T, K^T K\rangle + \frac{\lambda_1}{2}\langle\boldsymbol{w}\boldsymbol{w}^T, A\rangle \end{aligned} \tag{II.11}$$

where e is a vector with all 1s, and $\langle\cdot,\cdot\rangle$ denotes the inner product on vectors/matrices.

We define $\mathcal{F}_1 = \{(x, y) \mapsto \frac{2}{p}\langle\boldsymbol{w}, e^T K\rangle\}$, $\mathcal{F}_2 = \{(x, y) \mapsto \frac{1}{p}\langle\boldsymbol{w}\boldsymbol{w}^T, K^T K\rangle\}$ and $\mathcal{F}_3 = \{(x, y) \mapsto \frac{\lambda_1}{2}\langle\boldsymbol{w}\boldsymbol{w}^T, A\rangle\}$. $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ corresponds to the three terms in (II.11).

For $\mathcal{F}_1$, we define $h_1(\boldsymbol{w}) = 12/||\boldsymbol{w}||_2^2$, which is 1-strongly convex w.r.t $||\cdot||_2$ and its fenchel dual norm is also $||\cdot||_2$. With the fact that each attribute in any $X_i^{(q)}$ is scaled to $[0, 1]$, we can verify $||e^T K||_2 \leq \sqrt{2d}p$. With $s_i \in [0, 1]$ for all i and (II.7), we have

$$\begin{aligned} h_1(\boldsymbol{w}) = 12/||w||_2^2 &= ||s||_\infty^2 12/(||w/||s||_\infty^2||_2^2) \leq 12/ \\ &||s||_\infty^2||w/||s||_\infty^2||_1^2 \leq 12/||s||_\infty^2||\boldsymbol{w} \oslash \boldsymbol{s}||_1^2 \leq 1/2C^2. \end{aligned}$$

Using Lemma 2, we have $\mathcal{R}_p(\mathcal{F}_1) \leq 2C\sqrt{2d/p}$.

For $\mathcal{F}_2$, we have $||\boldsymbol{w}\boldsymbol{w}^T||_{1,1} = (\sum_{i=1}^{2d}\boldsymbol{w}_i)^2 \leq C^2$, and $||K^T K||_{\infty,\infty} \leq p$. We define $h_2(\boldsymbol{w}\boldsymbol{w}^T) = \sum_{i,j}((\boldsymbol{w}\boldsymbol{w}^T)_{i,j}/C^2)log((\boldsymbol{w}\boldsymbol{w}^T)_{i,j}/(C^2\mu_{i,j}))$, where μ is a probability distribution with $\sum_{i,j}\mu_{i,j} = 1$ and $\mu_{i,j} > 0$. Note that $h_2(\boldsymbol{w}\boldsymbol{w}^T)$ is the entropy function which is $1/C^4$-strongly convex with respect to $||.||_{1,1}$. By applying Lemma 2, we have$\mathcal{R}_p(\mathcal{F}_2) \leq C^2\sqrt{2log(4d)/p}$.

For $\mathcal{F}_3$, we have $||\boldsymbol{w}\boldsymbol{w}^T||_{1,1} \leq C^2$ and $||A||_{\infty,\infty} \leq 1$. Recall that $h_2(\boldsymbol{w}\boldsymbol{w}^T) = \sum_{i,j}((\boldsymbol{w}\boldsymbol{w}^T)_{i,j}/C^2)log((\boldsymbol{w}\boldsymbol{w}^T)_{i,j}/(C^2\mu_{i,j}))$ is $1/C^4$-strongly convex with respect to $||.||_{1,1}$. By applying Lemma 2, we obtain $\mathcal{R}_p(\mathcal{F}_3) \leq \frac{\lambda_1}{2}C^2\sqrt{2log(4d)/p}$.

Using the fact $\mathcal{R}_p(W) \leq \mathcal{R}_p(\mathcal{F}_1) + \mathcal{R}_p(\mathcal{F}_2) + \mathcal{R}_p(\mathcal{F}_3)$, we obtain the desired result in (II.10).

Using Theorem 1, we have $\ell(\boldsymbol{w})$ being Lipschitz with constant L_0. Hence, $\hat{\ell}(\boldsymbol{w})$ is also Lipschitz with L_0. Let $c_{\hat{\ell}} = (1 + C)^2 + \lambda_1/2C^2 - 1$. With (II.6), we have $\hat{\ell}(\boldsymbol{w})$ is bounded by $c_{\hat{\ell}}$. With (II.10), by applying Lemma 1 using $\hat{\ell}(\boldsymbol{w})$, we obtain the desired Rademacher bound in Theorem 3. This concludes the proof. ■

REFERENCES

[1] P. Li, C. Burges, Q. Wu, J. Platt, D. Koller, Y. Singer, and S. Roweis, "McRank: Learning to rank using multiple classification and gradient boosting," in *Advances in Neural Information Processing Systems*, vol. 20. Cambridge, MA, USA: MIT Press, 2007, pp. 897–904.

[2] O. Chapelle and S. Keerthi, "Efficient algorithms for ranking with SVMs," *Inf. Retr.*, vol. 13, no. 3, pp. 201–215, 2010.

[3] Y. Freund, R. Iyer, R. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, Nov. 2003.

[4] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 271–278.

[5] J. Xu and H. Li, "AdaRank: A boosting algorithm for information retrieval," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 391–398.

[6] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 129–136.

[7] F. Xia, T. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: Theory and algorithm," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1192–1199.

[8] X. Geng, T. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 407–414.

[9] K. Crammer and Y. Singer, "Pranking with ranking," in *Advances in Neural Information Processing Systems*, vol. 14. Cambridge, MA, USA: MIT Press, 2001, pp. 641–647.

[10] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 217–226.

[11] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1997, pp. 281–287.

[12] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. Cruz, "Quadratic programming feature selection," *J. Mach. Learn. Res.*, vol. 11, pp. 1491–1516, Apr. 2010.

[13] M. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.

[14] D. Stork, R. Duda, and P. Hart, *Pattern Classification*. New York, USA: Wiley, 2001.

[15] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. New York, USA: Springer-Verlag, 2004.

[16] Y. Nesterov, *Gradient Methods for Minimizing Composite Objective Function*. Louvain la Neuve, Belgium: CORE, 2007.

[17] A. Nemirovski, "Efficient methods in convex programming," Faculty Ind. Eng. Manage., Technion—The Israel Inst. Technol., Tech. Rep., 1994.

[18] J. Liu, J. Chen, and J. Ye, "Large-scale sparse logistic regression," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 547–556.

[19] S. Ji and J. Ye, "An accelerated gradient method for trace norm minimization," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 457–464.

[20] S. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Advances in Neural Information Processing Systems*, vol. 22. Cambridge, MA, USA: MIT Press, 2008, pp. 793–800.

[21] S. Kakade, S. Shalev-Shwartz, and A. Tewari, "On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization," unpublished.

[22] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, vol. 82. Reading, MA, USA: Addison-Wesley, 1999.

[23] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[24] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Halmilton, and G. Hullender, "Learning to rank using gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 89–96.

[25] T. Qin, T.Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retr. J.*, vol. 13, no. 4, pp. 346–374, 2010.

[26] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam, "OHSUMED: An interactive retrieval evaluation and new large test collection for research," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1994, pp. 192–201.

[27] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to Ad Hoc information retrieval," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2001, pp. 334–342.

[28] S. Robertson, "Overview of the okapi projects," *J. Document.*, vol. 53, no. 1, pp. 3–7, 1997.

[29] N. Craswell and D. Hawking, "Overview of the TREC-2004 web track," in *Proc. Text Retr. Conf.*, 2004, pp. 89–97.

[30] J. Allan, J. A. Aslam, V. Pavlu, E. Kanoulas, and B. Carterette, "Million query track 2008 overview," in *Proc. Text Retr. Conf.*, 2008, pp. 71–92.

[31] C. Li, L. Shao, C. S. Xu, and H. Lu, "Feature selection under learning to rank model for multimedia retrieval," in *Proc. 2nd Int. Conf. Internet Multimedia Comput. Service*, 2010, pp. 69–72.

[32] S. R. F. Da Silva, M. X. Ribeiro, J. E. S. Batista Neto, C. Traina, Jr., and A. J. M. Traina, "Improving the ranking quality of medical image retrieval using a genetic feature selection method," *Decision Support Syst.*, vol. 5, no. 4, pp. 810–820, 2011.

[33] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato, "Greedy and randomized feature selection for web search ranking," in *Proc. 11th Int. Conf. Comput. Inf. Technol.*, 2011, pp. 436–442.

[34] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "SortNet: Learning to rank by a neural preference function," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 22, no. 9, pp. 1368–1380, Sep. 2011.

[35] H. J. Lai, Y. Pan, C. Liu, L. Lin, and J. Wu, "Sparse learning-to-rank via an efficient primal-dual algorithm," *IEEE Trans. Comput.*, vol. 99, no. 99, p. 1, 2012.

[36] V. Dang and W. B. Croft, "Feature selection for document ranking using best first search and coordinate ascent," in *Proc. SIGIR Workshop Feature Generat. Sel. Inf. Retr.*, 2010, pp. 1–5.

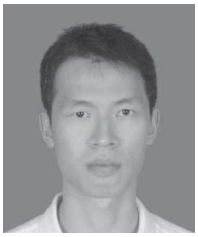
[37] S. Agarwal and P. Niyogi, "Generalization bounds for ranking algorithms via algorithmic stability," *J. Mach. Learn. Res.*, vol. 10, pp. 441–474, Feb. 2009.

[38] D. L. Donoho and Y. Tsaig, "Fast solution of ℓ_1 minimization problems when the solution may be sparse," *IEEE Trans. Inf. Theory*, vol. 54, no. 11, pp. 4789–4812, Nov. 2008.

[39] S. Perkins, K. Lacker, and J. Theiler, "Grafting: Fast, incremental feature selection by gradient descent in function space," *J. Mach. Learn. Res.*, vol. 3, nos. 7–8, pp. 1333–1356, 2003.

Hanjiang Lai received the B.S. degree from Sun Yat-Sen University, Kaohsiung, Taiwan, in 2009. He is currently pursuing the Ph.D. degree with the School of Information Science and Technology, Sun Yat-Sen University.

His current research interests include machine learning algorithms, optimization, and learning to rank.



Yan Pan received the B.S. and Ph.D. degrees in computer science from Sun Yat-Sen University, Kaohsiung, Taiwan, in 2002 and 2007, respectively.

He is currently an Assistant Professor with Sun Yat-Sen University. He has served as a reviewer for several conferences and journals. His current research interests include machine learning algorithms, learning to rank, and computer vision.

Yong Tang received the B.S. and M.S. degrees from Wuhan University, Wuhan, China, in 1985 and 1990, respectively, and the Ph.D. degrees from the University of Science and Technology of China, in 2001, all in computer science.

He is currently a Professor at the School of Computer Science, South China Normal University, Guangzhou, China. His current research interests include database and cooperative software, and temporal information processing. He has published more than 30 research and development projects, and has authored or co-authored more than 100 publications.

Rong Yu received the Ph.D. degree from Tsinghua University, Beijing, China, in 2007.

He was with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. In 2010, he joined the Institute of Intelligent Information Processing, Guangdong University of Technology, Guangdong, China. His current research interests include wireless communications and networking, including cognitive radio, wireless sensor networks, and home networking.