Modeling the Parameter Interactions in Ranking SVM with Low-Rank Approximation

Jun Xu, *Member, IEEE*, Wei Zeng, Yanyan Lan, *Member, IEEE*, Jiafeng Guo, *Member, IEEE*, and Xueqi Cheng, *Member, IEEE*

Abstract—Ranking SVM, which formalizes the problem of learning a ranking model as that of learning a binary SVM on preference pairs of documents, is a state-of-the-art ranking model in information retrieval. The dual form solution of a linear Ranking SVM model can be written as a linear combination of the preference pairs, i.e., $\boldsymbol{w} = \sum_{(i,j)} \alpha_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)$, where α_{ij} denotes the Lagrange parameters associated with each preference pair (i, j). It is observed that there exist obvious interactions among the document pairs because two preference pairs could share a same document as their items, e.g., preference pairs (d_1, d_2) and (d_1, d_3) share the document d_1. Thus it is natural to ask if there also exist interactions over the model parameters α_{ij}, which may be leveraged to construct better ranking models. This paper aims to answer the question. We empirically found that there exists a low-rank structure over the rearranged Ranking SVM model parameters α_{ij}, which indicates that the interactions do exist. Based on the discovery, we made modifications on the original Ranking SVM model by explicitly applying low-rank constraints to the Lagrange parameters, achieving two novel algorithms called Factorized Ranking SVM and Regularized Ranking SVM, respectively. Specifically, in Factorized Ranking SVM each parameter α_{ij} is decomposed as a product of two low-dimensional vectors, i.e., $\alpha_{ij} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$, where vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ correspond to document i and j, respectively; In Regularized Ranking SVM, a nuclear norm is applied to the rearranged parameters matrix for controlling its rank. Experimental results on three LETOR datasets show that both of the proposed methods can outperform state-of-the-art learning to rank models including the conventional Ranking SVM.

Index Terms—Learning to rank, ranking SVM, parameter interactions, low-rank approximation

1 Introduction

In recent years 'learning to rank' [1], [2] has gained increasing attention in both the fields of information retrieval and machine learning. When applied to document retrieval, learning to rank becomes a task as follows. In training, a ranking model is constructed with data consisting of queries, their corresponding retrieved documents, and relevance levels given by humans. In ranking, given a new query, the corresponding retrieved documents are sorted by using the trained ranking model. Among the different learning to rank models, the pairwise approaches [3], [4] have been widely used. Pairwise learning to rank formalizes the problem of learning a ranking model as that of binary classification over the document preference pairs. Each document preference pair consists of two retrieved documents in which the first document is more relevant than the second one w.r.t. the query.

Representative pairwise learning to rank models include Ranking SVM [4], RankNet [5], RankBoost [6], and IR-SVM [7] etc. For example, Ranking SVM evolves from the popular support vector machines (SVM) [8] for classification problems. In training, Ranking SVM first constructs the preference pairs of the documents based on their relevance labels (or click-through data [4]). Then, a binary classification SVM model is learned based on the preference pairs to capture the differences between documents with different relevance labels. In ranking, each document is assigned a relevance score based on the learned ranking model. According to the representer theorem [9] in machine learning, the Ranking SVM model can be written in their dual forms. Specifically, the solutions of the linear Ranking SVM optimization problem can be written as linear combinations of the training preference pairs, i.e., $\boldsymbol{w} = \sum_{(i,j)} \alpha_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)$, where $\boldsymbol{w}$ is the model parameter vector, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the feature vectors of the first and second document in the pair (i, j), and $\alpha_{ij} \in \mathbb{R}$ is the corresponding dual model parameter (Lagrange multiplier).[1]

In machine learning, it is usually supposed that the training instances are independent and identically distributed (I.I.D.). In Ranking SVM, however, the I.I.D. assumption is violated. It is easy to show that the training instances (the constructed document preference pairs) have strong interactions. This is because two preference pairs could share a same document as their preferred or unpreferred documents. For example, given a query q and three retrieved documents d_1, d_2 and d_3 whose relevance labels are 'relevant', 'irrelevant', and 'irrelevant', respectively. Two preference pairs (d_1, d_2) and (d_1, d_3) will be

- *J. Xu is with the Beijing Key Laboratory of Big Data Management and Analysis Methods, School of Information, Renmin University of China. E-mail: junxu@ruc.edu.cn.*
- *W. Zeng, Y. Lan, J. Guo, and X. Cheng are with the CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences. E-mail: zengwei@software.ict.ac.cn, {lanyanyan, guojiafeng, cxq}@ict.ac.cn.*

Manuscript received 24 Nov. 2016; revised 13 Mar. 2018; accepted 20 June 2018. Date of publication 28 June 2018; date of current version 26 Apr. 2019. (Corresponding author: Jun Xu.)
Recommended for acceptance by F. Silvestri.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2851257

1. In dual form Ranking SVM α_{ij} is a Lagrange multiplier corresponds to one document preference pair (i, j).

generated by Ranking SVM as the training instances for learning the binary SVM model. It is obvious that these two preference pairs have some interactions as both of them contains the document d_1 as the preferred document.

In the dual form solution of Ranking SVM model, each preference pair (i, j) is associated with a Lagrange multiplier α_{ij}. Therefore, it is natural to ask the following two questions: 1) *whether there also exist interactions among these Lagrange multipliers*; 2) If the answer is yes, *how to utilize the interactions to improve the Ranking SVM algorithm?*

This paper tries to answer the first question by analyzing the model parameters of a trained Ranking SVM model. Specifically, we analyzed the Lagrange multipliers of a trained Ranking SVM model as follows: first, we made an arrangement of the Lagrange multipliers and constructed a block diagonal matrix $\mathbf{A}$, where $A(i, j) = \alpha_{ij}$ if α_{ij} appears in the Ranking SVM model and zero otherwise. In the matrix $\mathbf{A}$, one block corresponds to the set of the preference pairs generated for one query. Then, we performed singular value decomposition (SVD) [10] on each block of $\mathbf{A}$ and sorted the eigenvalues in descending order. Finally, for each query, we gradually removed the dimensions with small eigenvalues, yielding a low-rank approximation of $\mathbf{A}$ and keeping most of the energy. We found that for all the queries, we just need 40 percent dimensions to capture 90 percent energy. If we want to capture 100 percent energy, however, almost all the queries need all of the dimensions. The phenomenon indicates that there exists a low-rank structure in the matrix $\mathbf{A}$. We conclude that there exist interactions among the Lagrange multipliers in Ranking SVM models.

To answer the second question, we propose to explicitly modeling the parameter interactions in the training process of Ranking SVM, based on the above discovery. Specifically, two approaches are proposed in the paper, called Factorized Ranking SVM and Regularized Ranking SVM, respectively. In Factorized Ranking SVM, each parameter α_{ij} in the Ranking SVM dual form objective function is factorized as a dot product of two K-dimensional latent vectors, i.e., $\alpha_{ij} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$, where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ correspond to the first document and second document in the preference pair, respectively. In this way, the rank of each block of matrix $\mathbf{A}$ would be less than K. The learning of the ranking model, then, becomes optimizing the factorized dual form objective function with respect to the latent vectors. In Regularized Ranking SVM, a nuclear norm regularizer is applied to matrix $\mathbf{A}$ and combined with the dual form Ranking SVM objective function, for controlling the rank of matrix $\mathbf{A}$. The learning of the ranking model, thus, becomes optimizing a dual form objective function regularized with a nuclear norm regularizer over $\mathbf{A}$.

We tested the performances of the proposed Factorized Ranking SVM and Regularized Ranking SVM models on the basis of LETOR datasets of OHSUMED, MQ2007, and MQ2008. Experimental results showed that both Factorized Ranking SVM and Regularized Ranking SVM can outperform several state-of-the-art baseline methods including Ranking SVM, in term of the popularly used evaluation measures of MAP and NDCG. Experimental results also showed that the Factorized Ranking SVM and Regularized Ranking SVM algorithms can achieve more improvements on the tasks in which the generated preference pairs have more interactions, i.e., each query has more labeled documents and generates more preference pairs.

Contributions of the paper include: 1) The paper investigated the impacts of the interactions among the document preference pairs generated for training the Ranking SVM models. A low-rank structure among the rearranged Ranking SVM Lagrange multipliers was observed and verified empirically; 2) Based on the observation, we proposed two approaches to directly modeling the parameter interactions, achieving two novel pairwise learning to rank models, referred to as Factorized Ranking SVM and Regularized Ranking SVM, respectively; 3) The effectiveness of the proposed approaches is tested on public available benchmark datasets.

The observations and the approaches to modeling parameter interactions (factorizing or regularizing the rearranged dual form parameters) are quite general. They can also be used to model the low rank structure in other pairwise learning to rank models.

The rest of the paper are organized as follows. After introducing the related work in Section 2, we present the pairwise learning to rank and one of the representative pairwise learning to rank model Ranking SVM in Section 3. In Section 4 we analyze the phenomenon of the parameter interactions in the trained Ranking SVM model. Based on the analysis, we proposed Factorized Ranking SVM and Regularized Ranking SVM in Section 5. Section 6 presents the experimental results and the conclusions are drawn in Section 7.

2 Related Work

2.1 Learning to Rank for Information Retrieval

Learning to rank has become one of the most active research topics in relevance ranking for IR [1]. State-of-the-art learning to rank models can be categorized into pointwise methods (e.g., PRank [11] and other models [12], [13]), pairwise methods (e.g., Ranking SVM [3], [4], RankNet [5], RankBoost [6], and IRSVM [7], [14]), and listwise methods (e.g., ListNet [15], AdaRank [16], LambdaMART [17] , XGBoost [18], PermuRank [19], and DSSVM [20]).

In this paper we focus on the pairwise methods which formalize the problem of learning a model for ranking documents as a problem of learning a binary classification model over the document preference pairs. Thus, the pairwise approaches first generate a set of preference pairs based on the labeled data, which is then used as the training data for learning the binary classification model. In this way, the pairwise approaches focus on predicting the relative order between two documents. Different binary classification models including SVM, Boosting, and Neural Networks can be applied here, achieves the pairwise learning to rank algorithms of Ranking SVM [3], [4], RankBoot [6], RankNet [5]. Among these algorithms, Ranking SVM [3], which applies binary SVM to solve the binary classification problem, is a representative pairwise learning to rank method. To make the algorithm more suitable to real-world IR applications, Joachims [4] proposed to train Ranking SVM with users' clickthrough data from search engines. Cao et al. [7] adapted Ranking SVM to document retrieval by modifying the loss function so that the training can focus on the top ranked documents.

A natural concern on the pairwise learning to rank is that the generated document preference pairs are not independent

and identically distributed, which violates the basic assumption of classification. In this paper, we found that these non-I.I.D. training pairs lead to strong interactions among the Lagrange multipliers in the Ranking SVM model, which can be utilized to design better pairwise learning to rank algorithms.

In recent years, learning to rank has been applied to the ranking tasks other than the relevance ranking. For example, a number of learning to rank algorithms has been proposed for search result diversification in which both the relevance and document novelty is considered simultaneously [21], [22], [23]. Qiang et al. [24] proposed Ranking FM to rank tweets in microblog retrieval. In Ranking FM, the interactions between features are modeled with factorization machine [25].

2.2 Low-Rank Approximation of a Matrix

In machine learning, it is common to approximate a set of parameters of interest as low-rank matrices. It has achieved great success in various tasks including dictionary learning [26], topic modeling [27], [28], [29], and low-rank matrix recovery and completion [30]. In general there exist two main approaches: rank minimization and matrix factorization.

A direct approach to approximating an input matrix with a low-rank one is to minimize the rank of the matrix with certain constraints that make the estimated matrix consistent with the original matrix. However, the rank minimization problem is combinatorial and known to be NP hard [31]. Convex relaxation is often used to make the minimization tractable. The most popular choice is to replace rank with the nuclear norm, which is defined as the sum of all of the singular values. For example, in [32], the low-rank matrix approximation problem is formalized as minimizing an objective function which consists of the F-norm term and the regularization term. The F-norm term measures the difference between the recovered matrix and the input matrix. The regularization term is defined as a nuclear norm over the recovered matrix. One of the most popular methods for solving this problem is in the class of iterative shrinkage-thresholding algorithm (ISTA) [33], where each iteration involves a gradient step followed by a shrinkage/soft-threshold step. A Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [34] is proposed, which preserves the computational simplicity of ISTA, but with a global rate of convergence which is proven to be significantly better, both theoretically and practically.

Another approach to modeling the low-rank structure is matrix factorization which decomposes the input matrix as a product of two (or more) matrices. Using matrix factorization to model a low-rank matrix is based on the fact that the rank of the product matrix cannot be greater than the ranks of any factor matrices. Therefore the problem of modeling a low rank matrix can be converted to estimating two factor matrices. A popular approach to solving the problem is to minimizing the objective function over of of the two factor matrices alternately by fixing the other one. Matrix factorization has been widely used in variant tasks such as collaborative filtering [35] and topic modeling [27], [29].

In this paper, we utilize both of the rank minimization and matrix factorization approaches to modeling the low rank structure over the Lagrange multipliers, which capture the parameter interactions in the Ranking SVM model effectively.

3 Ranking SVM

In this section, we introduce the problem formulation of pairwise learning to rank and the representative pairwise learning to algorithm Ranking SVM.

3.1 Pairwise Learning to Rank

The problem of learning to rank for information retrieval can be formulated as follows. In retrieval (testing), given a query the retrieval system returns a ranking list of documents in descending order of the relevance scores. The relevance scores are calculated with a ranking function (model). In learning (training), a number of queries and their corresponding retrieved documents are given. Furthermore, the relevance levels of the documents with respect to the queries are also provided. The relevance levels are represented as ranks (i.e., categories in a total order). The objective of learning is to construct a ranking function which achieves the best results in ranking of the training data in the sense of minimization of a loss function.

Suppose we are given a set of training label-query-document tuples $\{(y_i, q_i, \boldsymbol{x}_i)\}_{i=1}^{N}$, where N is the number of training tuples, $y_i \in \{r_1, \cdots, r_\ell\}$ is the relevance label for the ith query-document pair, ℓ is the number of relevance levels, $q_i \in Q$ is the query for the ith query-document pair and Q is the set of queries in training data. There exists a total order between the relevance labels $r_\ell \succ r_{\ell-1}, \ldots, \succ r_1$, where '$\succ$' denotes a preference relationship. $\boldsymbol{x}_i \in \mathcal{X}$ is the feature vector encoding the ith query-document pair (q_i, d_i), i.e., the ranking features $\boldsymbol{x}_i = \boldsymbol{\phi}(q_i, d_i)$, where $\boldsymbol{\phi}$ is the feature function describing the relationship between the query q_i and the document d_i. The objective of learning is to create a ranking function $f : \mathcal{X} \mapsto R$ such that for each query the documents in its corresponding document list can be assigned relevance scores using the function and then be ranked according to the scores.

In learning, pairwise learning to rank formalizes the problem of learning a ranking model as the problem of learning a binary classification model over preference document pairs. Thus, the first step of pairwise learning to rank is to generate a set of preference pairs based on the original training data. The set of preference pairs is defined as

$$P \equiv \{(i, j) | q_i = q_j, y_i \succ y_j\}.$$

From the definition, we can see that two documents form a preference pair only if they are retrieved by one query and are labeled with different relevance levels. The second step of pairwise learning to rank is to define the pairwise object function based on the preference pairs. Similar to the loss functions in conventional classification, the pairwise loss function consists of the empirical loss on the training data and the penalty over the ranking model

$$\min_{f \in \mathcal{F}} L^{\text{pairwise}} = \sum_{(i,j) \in P} l(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)) + \Omega(f), \tag{1}$$

where $\mathcal{F}$ is the set of possible ranking models, $l(\cdot)$ is the loss for each of the training preference pair which takes the difference of the ranking scores between the first document and the second document as input, and Ω is the regularizer for controlling the complexity of ranking model f.

Minimizing the loss function achieves the pairwise learning to rank model. Different choices of the ranking model f, the pairwise loss l, the regularizer Ω, and the optimization techniques lead to different pairwise learning to rank models.

In ranking, given a user query q the retrieve system returns the retrieved candidate documents. We assign a score to each of the documents using $f(\boldsymbol{x})$ and sort the candidate documents based on their scores.

Note that the pairwise approach does not focus on accurately predicting the relevance levels of the query-document pairs. Instead, it cares about the relative order between two retrieved documents given the query. In this sense, the learning cares more about the relative differences of the training pairs (i.e., $f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)$) than the absolute value of documents (i.e., $f(\boldsymbol{x}_i)$). The property makes the pairwise learning to rank is closer to the concept of "ranking" than the pointwise approach which directly considers the relevance labels as categories.

The binary classification used in the pairwise learning to rank is actually different to the classification used in conventional machine learning, since it operates on every two documents under investigation. A natural concern is that document preference pairs are not I.I.D., which violates the basic assumption of classification. In practice, the problem is usually ignored and people directly apply the binary classification technologies for learning the ranking model. In theory, to the best of our knowledge, there exists very few work to investigate the fundamental problem.

3.2 The Ranking SVM Model

The Ranking SVM model adopt the hinge loss as its pairwise loss and use the large margin binary classifier of SVM to solve the binary classification problem.

In this paper we consider the linear Ranking SVM model in which the ranking model f is assumed to be a linear function. That is, the ranking model f takes a linear form $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$, where $\boldsymbol{w}$ denotes the weights for features and $\langle \cdot, \cdot \rangle$ stands for the dot product. Note that compared with the conventional linear classification model, $f(\boldsymbol{x})$ contains no bias term b because in pairwise ranking we only care about the relative differences of two documents in a preference pair. The bias term in the ranking model does not affect the ranking of the documents and thus can be ignored.

Further assuming that the regularizer term takes the ℓ_2-norm, the optimization problem of Eq. (1) becomes

$$\min_{\boldsymbol{w}} \sum_{(i,j)\in P} l(\langle \boldsymbol{w}, \boldsymbol{x_i} - \boldsymbol{x_j} \rangle) + \lambda \cdot \|\boldsymbol{w}\|^2, \tag{2}$$

where $\|\cdot\|^2$ is the ℓ_2-norm and $\lambda > 0$ is the trade-off parameter. Assumes that the pairwise loss l takes the hinge function, i.e., $l(x) = [1 - x]_+$ where $[x]_+ = \max(0, x)$. Thus, the optimization problem can be written as

$$\min_{\boldsymbol{w}} L^{\text{RSVM}} = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{(i,j)\in P} \left[1 - \langle \boldsymbol{w}, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle\right]_+, \tag{3}$$

where $C > 0$ is a trade-off parameter which plays a similar role as λ in Eq. (2). In Ranking SVM, it is common to solve the dual problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} - \boldsymbol{e}^T \boldsymbol{\alpha}, \\ &\text{s.t.} \qquad 0 \le \boldsymbol{\alpha}_{ij} \le C, \forall (i,j) \in P, \end{aligned} \tag{4}$$

where $\boldsymbol{\alpha} \in \mathcal{R}^{|P|}$ is a vector of Lagrange multipliers indexed by the preference pairs in P, $\boldsymbol{e} \in \mathcal{R}^{|P|}$ is the vector of ones, $\mathbf{M}$ is a $|P| \times |P|$ kernel matrix over the preference pairs, and the entries of $\mathbf{M}$ are defined as

$$M_{(i,j),(u,v)} = \langle \boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_u - \boldsymbol{x}_v \rangle, \tag{5}$$

for all $((i,j),(u,v)) \in P \times P$.

In ranking, the learned ranking model $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$ assigns a relevance score for each test query-document pair $\boldsymbol{x}$. The documents retrieved by the same query are sorted in decending order according to the relevance scores.

It can be shown that the optimal solution of Eq. (3) $\boldsymbol{w}$ can be represented as its dual form [36]

$$\boldsymbol{w} = \sum_{(i,j)\in P} \alpha_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{6}$$

where α_{ij} is the Lagrange multiplier corresponds to preference pair (i,j).

4 Analysis of Parameter Interactions in Ranking SVM

Ranking SVM formalizes the problem of learning a model for ranking documents as learning a binary classification model over the document preference pairs and trains a binary classification SVM model based on the generated pairs. The formulation assumes that the generated document preference pairs can be used as training data for estimating the model parameters. In machine learning it is usually assumed that all of the training instances are I.I.D. It is obvious, however, that the assumption is violated in Ranking SVM: the document preference pairs generated in Ranking SVM have strong interactions and cannot be I.I.D. This is because two preference pairs may share a common document as their preferred or/and not preferred documents.

For example, assuming that there exists a training query q which retrieves three documents d_1, d_2, and d_3. These three documents are labeled as 'relevant', 'irrelevant', and 'irrelevant', respectively. According to the rules for generating the preference pairs, two preference pairs (d_1, d_2) and (d_1, d_3) will be generated by Ranking SVM as the training instances. It is obvious that these two preference pairs cannot be I.I.D. and have some interactions, as both of them contains the document d_1 as their preferred document.

From the dual form of the ranking model Eq. (6), we know that each of the model parameter α_{ij} is associated with a preference pair (i,j). It is natural to ask whether there also exist interaction among the Ranking SVM Lagrange multipliers α_{ij}'s.

To answer the question, we propose to empirically analyze the interactions between model parameters based on SVD [10]. Specifically, we trained a Ranking SVM model based on one fold of the OHSUMED dataset [37]. Then, we extracted the Lagrange multipliers $\boldsymbol{\alpha}$ from the learned Ranking SVM model and rearranged the parameters as a matrix $\mathbf{A} \in \mathcal{R}^{N \times N}$. The entries of matrix $\mathbf{A}$ are defined as

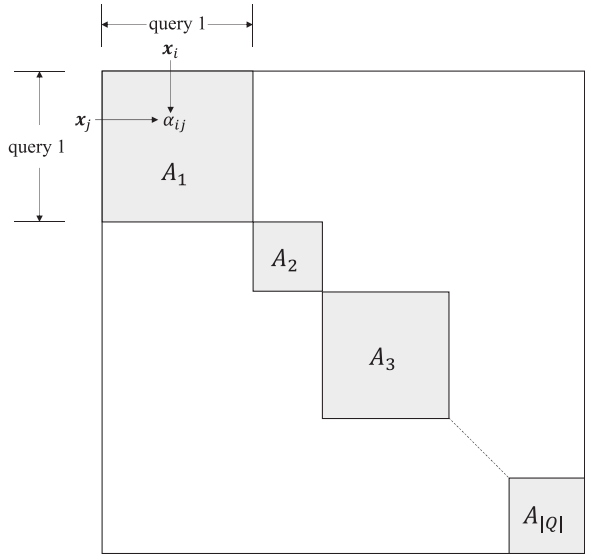


Fig. 1. Re-arrangement of the model parameters $\boldsymbol{\alpha} \in \Re^{|P|}$ as a matrix $\mathbf{A} \in \mathcal{R}^{N \times N}$. $\mathbf{A}$ is block diagonal and each block $\mathbf{A}_l$ corresponds to the Lagrange multipliers related to a training query.

$$A(i,j) = \begin{cases} \alpha_{ij} & (i,j) \in P \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

for all $1 \leq i \leq N$ and $1 \leq j \leq N$.

Note that $\mathbf{A}$ is a block diagonal matrix, which consists a set of sub-matrices $\mathbf{A}_l (l = 1, \ldots, |Q|$ where Q is the set of unique queries in training set), and each block $\mathbf{A}_l$ corresponds to a training query, as shown in Fig. 1. This is because in Ranking SVM the preference pairs reflect the relative relevance given a specific query. Thus, the preference pairs are generated within the documents retrieved by one query.

In the analysis, we observed that the interactions in the training preference pairs make the parameter matrix $\mathbf{A}$ (also the sub-matrices $\mathbf{A}_l$'s) tends to be low-rank. To show this, for each of the sub-matrix $\mathbf{A}_l$ (corresponds to a query), we performed the SVD on it. Let

$$\mathbf{A}_l = \sum_{i=1}^{r_l} \sigma_{li} \mathbf{u}_{li} \mathbf{v}_{li}^T,$$

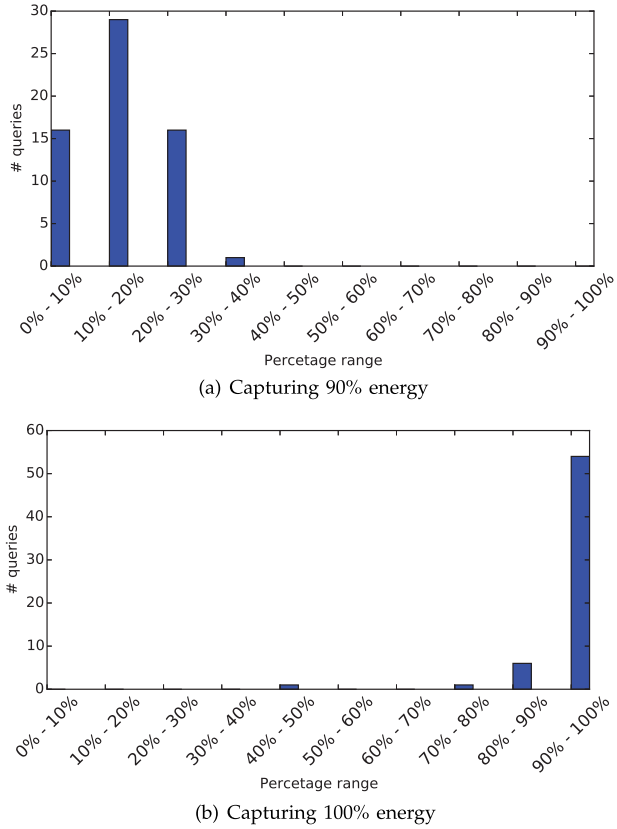
be the SVD of $\mathbf{A}_l$, where σ_{li}, $\mathbf{u}_{li}$, and $\mathbf{v}_{li}$ are the ith eigenvalue, the ith left eigenvector, and the ith right eigenvector of matrix $\mathbf{A}_l$, respectively. Assuming that the eigenvalues are sorted in descending order: $\sigma_{l1} \geq \sigma_{l2} \geq \cdots \geq \sigma_{lr_l}$. For $k \in \{1, 2, \ldots, r_l\}$, let

$$\mathbf{A}_l^k = \sum_{i=1}^{k} \sigma_{li} \mathbf{u}_{li} \mathbf{v}_{li}^T,$$

be the sum truncated after k terms. It is clear that $\mathbf{A}_l^k$ has rank k and keeps only the energy related to the top k eigenvalues. Thus, the percentage of the energy kept in the truncated sum $\mathbf{A}_l^k$ is

$$PE_l(k) = \frac{\sum_{i=1}^{k} \sigma_{li}^2}{\sum_{i=1}^{r_l} \sigma_{li}^2} \times 100\%.$$

With the definition of $PE_l(k)$, we can empirically find the smallest dimensions k that can keep at least $t\%$ of the energy in $\mathbf{A}_l$, denoted as k_l



(a) Capturing 90% energy

(b) Capturing 100% energy

Fig. 2. Distribution of Dim_t over percentage ranges when $t\%$ is set to 90 percent (a) or 100 percent (b).

$$k_l = \min\{k | PE_l(k) \geq t\%\}.$$

For all of the training queries in Q and the corresponding sub-matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{|Q|}$, given the energy threshold $t\%$, we can find $|Q|$ values $\{k_1, \ldots, k_{|Q|}\}$. Intuitively, these values reflect the minimal number of dimensions needed for keeping the $t\% (t < 100)$ energy for each query. Small values indicates the low-rank structure.

The dimensions of the sub-matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{|Q|}$ could be very different. For example, some popular queries can retrieve a lot of related documents for labeling and some rare queries have a very small number of related documents. To making the values in $\{k_1, \ldots, k_{|Q|}\}$ comparable, we normalize the k_l's with the corresponding matrix sizes and get $|Q|$ ratios

$$Dim_t = \left\{ \frac{k_1}{\text{size}(\mathbf{A}_1)}, \ldots, \frac{k_{|Q|}}{\text{size}(\mathbf{A}_{|Q|})} \right\},$$

where the operator $\text{size}(\mathbf{A}_l)$ returns the dimensions of the input matrix $\mathbf{A}_l$. Please note that $\mathbf{A}_l$ is a square matrix and $\text{size}(\mathbf{A}_l)$ only returns the number of columns (or rows) of the input matrix.

The distribution of the values in Dim_t can be used to indicate the rank structure of the $|Q|$ sub-matrices. For example, if most of the ratios are small (e.g., less than 50 percent) for a large energy threshold (e.g., $r\% = 90\%$), it indicates that there exists low-rank structures over most of the sub-matrices.

Fig. 2a plots the distribution of Dim_t and the energy threshold parameter $t\%$ is set to 90 percent. In Fig. 2a, the

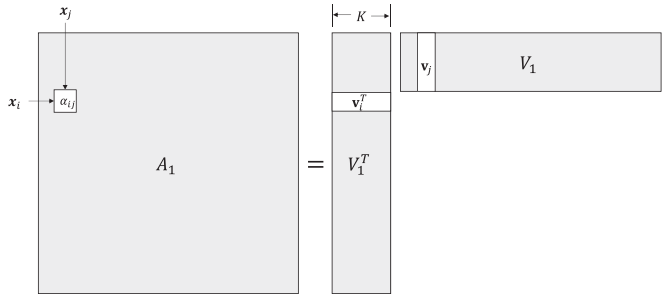


Fig. 3. Each sub-matrix of $\mathbf{A}_l$ is factorized into $\mathbf{V}_l^T \times \mathbf{V}_l$.

x-axis indicates the range of the values and y-axis is the amount of elements in Dim_t that falls into the range. For example, the bar denoted with $0\% - 10\%$ takes the value 16, which means that there are 16 elements in Dim_t whose values are between 0 and 10 percent. From the statistics shown in Fig. 2a, we can see that for most of the sub-matrices, less than 30 percent of the original dimensions are needed for capturing 90 percent of the total energy. Since these sub-matrices are constructed based on the Ranking SVM's Lagrange multipliers $\boldsymbol{\alpha}$, we can conclude that there exists very strong low-rank structures over the ranking SVM parameters $\boldsymbol{\alpha}$.

We noticed that although these sub-matrices contains a low rank structure, but most of them are nearly full rank matrices, as shown in Fig. 2b.

We also conducted the analysis on the Ranking SVM models trained based on other datasets and similar results were observed. The results indicate that it is a common phenomenon that the conventional Ranking SVM model parameters have interactions, which leads to a low-rank structure upon the rearranged parameter matrix. It is an interesting and important question whether the phenomenon can be utilized to improve the conventional Ranking SVM.

5 Ranking SVM with Low-Rank Approximation

Based on the analysis above, we tried to improve the Ranking SVM algorithm through explicitly modeling the low-rank structure in the rearranged Lagrange multipliers. Specifically, we propose to approximate the parameter matrix $\mathbf{A}$ in Ranking SVM with its low-rank approximations. Both the matrix factorization approach and the direct ranking minimization approach can be used here, lead to the new algorithms called Factorized Ranking SVM and Regularized Ranking SVM, respectively.

5.1 Factorized Ranking SVM

Factorization based approaches can be used for modeling the parameter interactions in Ranking SVM. In this section we show a method of factorizing the Lagrange multipliers in Ranking SVM, achieving Factorized Ranking SVM.

5.1.1 Factorizing the Parameter Matrix $\mathbf{A}$

In Factorized Ranking SVM, the low-rank structure of each block matrix $\mathbf{A}_l$ in $\mathbf{A}$ is explicitly modeled with the product of a K-dimensional latent matrix and its transpose, i.e., $\mathbf{A}_l = \mathbf{V}_l^T \times \mathbf{V}_l$ where $\mathbf{V}_l$ is the corresponding latent matrix, as shown in Fig. 3. Here $\mathbf{V}_l$ consists of the K-dimensional latent vectors correspond to the parameters α_{ij}'s in $\mathbf{A}_l$. In this way, the rank of each block matrix $\mathbf{A}_l$ will be less than or equals to K. Usually, K is set to a small number to make sure that each block matrix is low rank.

More specifically, each of the model parameter α_{ij} is assumed to be a product of two K-dimensional vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$

$$\alpha_{ij} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle, \tag{8}$$

for all $(i, j) \in P$. Note that in the approach, if two model parameters α_{ij} and α_{ik} whose corresponding preference pairs share document d_i as their preferred document, the same vector $\boldsymbol{v}_i$ will be used for representing both of these two parameters, i.e., $\alpha_{ij} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$ and $\alpha_{ik} = \langle \boldsymbol{v}_i, \boldsymbol{v}_k \rangle$.

Replacing the $\boldsymbol{\alpha}$ in the dual form Ranking SVM objective function (4) with Eq. (8), we get the optimization problem of Factorized Ranking SVM

$$\min_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N} \frac{1}{2} \begin{bmatrix} \vdots \\ \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \\ \vdots \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \vdots \\ \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \\ \vdots \end{bmatrix} + \boldsymbol{e}^T \begin{bmatrix} \vdots \\ \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \\ \vdots \end{bmatrix}$$
$$s.t. \quad 0 \leq \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \leq C \quad \forall (i, j) \in P.$$

Replacing the elements in the kernel matrix $\mathbf{M}$ with their definitions in Eq. (5), it is easy to show that above optimization problem is equivalent to

$$\min_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N} \frac{1}{2} \left\| \sum_{(i,j)\in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|^2 + \sum_{(i,j)\in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \tag{9}$$
$$s.t. \quad 0 \leq \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \leq C \quad \forall (i, j) \in P.$$

It is difficult to optimize the Problem (9) directly because of the complex constraints $0 \leq \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \leq C, \forall (i, j) \in P$. In this paper we resort to a new problem in which the constraints are simpler. More specifically, the constraints that are defined on the dot products of the pairs are replaced with the box constraints that are directly defined on each of the parameters v_{ik}'s

$$\min_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N} \frac{1}{2} \left\| \sum_{(i,j)\in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|^2 + \sum_{(i,j)\in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \tag{10}$$
$$s.t. \quad 0 \leq v_{ik} \leq \sqrt{\frac{C}{K}} \quad \forall i, k,$$

where v_{ik} is the kth element of the parameter vector $\boldsymbol{v}_i$ and K is the length of $\boldsymbol{v}_j$. Please note that a feasible region defined by the constraints in Problem (10) is a convex subset of the feasible region in Problem (9). The new constraints are stricter while simpler than the original ones.

5.1.2 Optimization

A number of methods can be used for optimizing Problem (10). In this paper, we adopt the popularly used projected gradients method [38]. Projected gradient methods optimize an unconstrained and uncontinuous loss function through updates the parameters based on the continuously

differentiable part and then make an euclidean projection to the feasible region.

Let's define an indicator function that penalizes the point out of the feasible region: $1_{\mathcal{C}}(\boldsymbol{v}) = \begin{cases} +\infty & \boldsymbol{v} \in \mathcal{C}, \\ 0 & \text{otherwise.} \end{cases}$ where $\mathcal{C}$ is a convex set which defines the feasible results for the problem. In our problem, $\mathcal{C}$ is defined by the box constraints: $\mathcal{C} = \left\{ \boldsymbol{v} \middle| \forall k \in \{1, \ldots, K\} : 0 \leq v_k \leq \sqrt{\frac{C}{K}} \right\}$. With the definition of $1_{\mathcal{C}}(\boldsymbol{v})$, the constrained Problem (10) can be written as an unconstrained one

$$\min_{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N} \frac{1}{2} \left\| \sum_{(i,j) \in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|^2 + \sum_{(i,j) \in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle + \sum_{i=1}^{N} 1_{\mathcal{C}}(\boldsymbol{v}_i),$$

where

$$f(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N) = \frac{1}{2} \left\| \sum_{(i,j) \in P} \langle \boldsymbol{v}_i \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|^2 + \sum_{(i,j) \in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle,$$

is the continuous differentiable part of the objective function and $\sum_{i=1}^{N} 1_{\mathcal{C}}(\boldsymbol{v}_i)$ is the un-differentiable part of the objective function.

In the projected gradient method, we compute the gradient of the differentiable part to update the parameters with a stochastic manner at the query level. After updating the parameters, we make an euclidean projection to the convex set $\mathcal{C}$ to make sure that the solution is in the feasible region. Specifically, given a query q and the corresponding preference pairs $P_q = \{(i,j) | q_i = q_j = q, y_i \succ y_j\}$. The gradient of $\boldsymbol{v}_i$ can be calculated as

$$\frac{\partial f}{\partial \boldsymbol{v}_i} = \begin{cases} \boldsymbol{0} & \forall j : (i,j) \notin P_q \wedge (j,i) \notin P_q \\ \boldsymbol{a} + \boldsymbol{b} & \text{otherwise,} \end{cases} \tag{11}$$

where $\boldsymbol{0}$ is the vector of zeros and

$$\begin{aligned} \boldsymbol{a} &= \sum_{i:(i,j) \in P_q} \boldsymbol{v}_j \left(\sum_{(u,v) \in P} \langle \boldsymbol{v}_u, \boldsymbol{v}_v \rangle \langle \boldsymbol{x}_u - \boldsymbol{x}_v, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle - C \right) \\ &= \sum_{i:(i,j) \in P_q} \boldsymbol{v}_j \big(\langle \boldsymbol{\beta}, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle - C \big), \\ \boldsymbol{b} &= \sum_{j:(j,i) \in P_q} \boldsymbol{v}_j \left(\sum_{(u,v) \in P} \langle \boldsymbol{v}_u, \boldsymbol{v}_v \rangle \langle \boldsymbol{x}_u - \boldsymbol{x}_v, \boldsymbol{x}_j - \boldsymbol{x}_i \rangle - C \right) \\ &= \sum_{j:(j,i) \in P_q} \boldsymbol{v}_j \big(\langle \boldsymbol{\beta}, \boldsymbol{x}_j - \boldsymbol{x}_i \rangle - C \big), \end{aligned}$$

where $\boldsymbol{\beta} = \sum_{(k,l) \in P} \langle \boldsymbol{v}_k, \boldsymbol{v}_l \rangle (\boldsymbol{x}_k - \boldsymbol{x}_l)$. Here we use the intermediate variable $\boldsymbol{\beta}$ because it can be updated incrementally during the optimization, which avoids the algorithm to go through all of the preference pairs when calculating the gradients. For further details, please see the lines 6, 10, and 11 of Algorithm 1.

Thus in the updating phase, the parameters $\boldsymbol{v}_i$ are updated as $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \eta \frac{\partial f}{\partial \boldsymbol{v}_i}$, for all $i = 1, \ldots, N$, and η is the step size.

In the projection phase, the updated solution is projected to the feasible region $\mathcal{C}$: $\boldsymbol{v}_i \leftarrow P_{\mathcal{C}}(\boldsymbol{v}_i)$, where $P_{\mathcal{C}} : \mathcal{R}^n \rightarrow \mathcal{C}$ is the euclidean projection on the convex set $\mathcal{C}$, which is defined as

$$P_{\mathcal{C}}(\boldsymbol{v_i})_k = \begin{cases} 0 & v_{ik} \leq 0, \\ v_{ik} & 0 \leq v_{ik} \leq \sqrt{\frac{C}{K}}, \\ \sqrt{\frac{C}{K}} & v_{ik} \geq \sqrt{\frac{C}{K}}, \end{cases} \tag{12}$$

for all $k = 1, \ldots, K$.

Algorithm 1 illustrates the optimization algorithm. Please note that in Algorithm 1, we update the $\boldsymbol{\beta}$ incrementally after changing the parameters $\boldsymbol{v}$'s related to each query. The operation avoids the algorithm to scan all of the preference pairs in P at each iteration.

Algorithm 1. Factorized Ranking SVM

Input: training data $D = \{y_i, q_i, \boldsymbol{x}_i\}_{i=1}^{N}$, learning rate $\eta > 0$, number of hidden dimensions K, parameter C, and number of iterations T

Output: model parameters $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$

1: $P \leftarrow \bigcup_{q=1}^{|Q|} P_q = \bigcup_{q=1}^{|Q|} \{(i,j) | q_i = q_j = q, y_i \succ y_j\}$, where Q is the set of training queries in D.
2: $(\boldsymbol{v}_1^{(0)}, \ldots, \boldsymbol{v}_N^{(0)}) \leftarrow$ random values in $\left[0, \sqrt{\frac{C}{K}}\right]$
3: $\boldsymbol{\beta} = \sum_{(i,j) \in P} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j)$
4: **for** $t = 1$ to T **do**
5: **for all** $q \in Q$ **do**
6: $\boldsymbol{\beta}_q = \sum_{(i,j) \in P_q} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j)$
7: **for all** $i \in \{i | \exists j : (i,j) \in P_q \vee (j,i) \in P_q\}$ **do**
8: $\boldsymbol{v}_i \leftarrow P_{\mathcal{C}}\left(\boldsymbol{v}_i + \eta \frac{\partial f}{\partial \boldsymbol{v}_i} \right)$ {Eqs. (11) and (12)}
9: **end for**
10: $\boldsymbol{\beta}_q' = \sum_{(i,j) \in P_q} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle (\boldsymbol{x}_i - \boldsymbol{x}_j)$
11: $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \boldsymbol{\beta}_q + \boldsymbol{\beta}_q'$ {update $\boldsymbol{\beta}$ incrementally}
12: **end for**
13: **end for**
14: **return** $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$

5.2 Regularized Ranking SVM

Direct rank minimization approaches can also be used for modeling the parameter interactions in Ranking SVM. In this section we show a method of using the nuclear norm regularizer for the task, achieving Regularized Ranking SVM.

5.2.1 Ranking SVM with Nuclear Norm Regularizer

The low-rank structure of parameter matrix $\mathbf{A}$ can be modeled via directly minimizing the rank of $\mathbf{A}$. Specifically, given the dual form objective function of Ranking SVM, the low-rank structure of matrix $\mathbf{A}$ can be modeled through applying a rank regularizer to $\mathbf{A}$, as shown in the follow optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} - \boldsymbol{e}^T \boldsymbol{\alpha} + \gamma \cdot \operatorname{rank}(\mathbf{A}) \\ s.t. \quad & 0 \leq \alpha_{i,j} \leq C, \forall (i,j) \in P, \end{aligned} \tag{13}$$

where the matrix $\mathbf{A}$ is achieved by reshaping the vector $\boldsymbol{\alpha}$, as shown in Eq. (7), $\operatorname{rank}(\mathbf{A}) \in Z^+ \cup \{0\}$ is the rank operator of matrix $\mathbf{A}$, and $\gamma \geq 0$ is the tradeoff parameter.

It is well known that directly minimizing the rank of a matrix is NP-hard [31]. Convex relaxation is often used to replace rank with the nuclear norm [39] which is defined as $\|\mathbf{A}\|_* = \sum_{i=1}^{r} \sigma_i$, where $\sigma_1, \sigma_2, \ldots, \sigma_r$ are the singular values of $\mathbf{A}$ and r is the rank of $\mathbf{A}$. It can be proved that the nuclear norm $\|\mathbf{A}\|_*$ is the convex envelope of rank($\mathbf{A}$)[39].[2]

To describe the relationship between the Lagrange multipliers vector $\boldsymbol{\alpha}$ and the matrix $\mathbf{A}$, we define two indicator matrices, $\mathbf{I}_1 \in \mathcal{R}^{N\times P}$ and $\mathbf{I}_2 \in \mathcal{R}^{N\times P}$. Each row of $\mathbf{I}_1$ (and $\mathbf{I}_2$) corresponds a document and each column corresponds a preference pair in P. The entries of matrix $\mathbf{I}_1$ are set as: the entry of ith row and (i,j)th column is set to one if there exists a preference pair (i,j) in P, otherwise zero. Similarly, the entries of matrix $\mathbf{I}_2$ are set as: the entry of jth row and (i,j)th column is set to one if there exists a preference pair (i,j) in P, otherwise zero. Please note that each column $\mathbf{I}_1$ and $\mathbf{I}_2$ has only one nonzero entry.

It is easy to show that the relationship between $\boldsymbol{\alpha}$ and $\mathbf{A}$ can be described as $\boldsymbol{\alpha} = \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e$, where $e \in \mathcal{R}^{|P|}$ is a vector of ones. Thus, the optimization Problem (13) can be written as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T M\boldsymbol{\alpha} - e^T\boldsymbol{\alpha} + \gamma\|\mathbf{A}\|_* + 1_{\mathcal{C}}(\boldsymbol{\alpha}) \\ s.t. \quad & \boldsymbol{\alpha} = \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e, \end{aligned} \tag{14}$$

where $1_{\mathcal{C}}(\boldsymbol{\alpha}) = \begin{cases} +\infty & \boldsymbol{\alpha} \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$ and $\mathcal{C} = \{\boldsymbol{\alpha} | 0 \le \alpha_{ij} \le C, \forall i,j\}$.

Problem (14) is not smooth because the indicator function $1_{\mathcal{C}}(\boldsymbol{\alpha})$ and the nuclear norm regularization $\|A\|_*$ are not smooth, which makes it hard to directly apply the general optimization methods such as gradient descent to perform the optimization.

5.2.2 Optimization

In this paper, we adopt the proximal gradients method for optimizing the Problem (14). In the method, the parameters $\boldsymbol{\alpha}$ and $\mathbf{A}$ are decoupled as two sets of parameters during the optimization process. A new regularizer over both $\boldsymbol{\alpha}$ and $\mathbf{A}$ is used to avoid the values of $\boldsymbol{\alpha}$ being too different from the values of $\mathbf{A}$. Thus, the optimization process can be decomposed as alternatively optimizing two sub optimization problems, which can be solved with standard proximal optimization methods efficiently.

Specifically, we approximate the Problem (14) by relaxing the constraint $\boldsymbol{\alpha} = \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e$ as a regularizer $\|\boldsymbol{\alpha} - \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e\|^2$ in the object function. Thus, we get a new optimization problem in which the parameters $\boldsymbol{\alpha}$ and $\mathbf{A}$ are decoupled

$$\min_{\boldsymbol{\alpha},\boldsymbol{A}} \varphi(\boldsymbol{\alpha}) + \gamma\|\mathbf{A}\|_* + 1_{\mathcal{C}}(\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha} - \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e\|^2, \tag{15}$$

where $\varphi(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T M\boldsymbol{\alpha} - e^T\boldsymbol{\alpha}$ is the smooth part of the loss function, and $\lambda > 0$ is a parameter for controlling the difference between $\boldsymbol{\alpha}$ and $\mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e$. Please note that in the new problem we optimize the objective function with respect to both $\boldsymbol{\alpha}$ and $\mathbf{A}$, which means $\boldsymbol{\alpha}$ and $\mathbf{A}$ are considered as two sets of parameters.

The alternative optimization method is adopted for conducting the optimization, in which the parameters $\boldsymbol{\alpha}$ and $\mathbf{A}$ are updated alternatively. Specifically, in the tth iteration, the algorithm first finds the optimal solution of $\boldsymbol{\alpha}$ (denoted as $\boldsymbol{\alpha}^{(t+1)}$), by fixing the values in matrix $\mathbf{A}$ (denoted as $\mathbf{A}^{(t)}$). That is, we solve the problem

$$\boldsymbol{\alpha}^{(t+1)} = \arg\min_{\boldsymbol{\alpha}} \varphi(\boldsymbol{\alpha}) + 1_{\mathcal{C}}(\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha} - \mathbf{I}_1^T\mathbf{A}^{(t)}\mathbf{I}_2 e\|^2. \tag{16}$$

Then, the algorithm finds the optimal solution of $\mathbf{A}$ (denoted as $\mathbf{A}^{(t+1)}$), by fixing the values in vector $\boldsymbol{\alpha}$ (denoted as $\boldsymbol{\alpha}^{(t+1)}$). That is, we solve the problem of

$$\mathbf{A}^{(t+1)} = \arg\min_{\mathbf{A}} \gamma\|\mathbf{A}\|_* + \lambda\|\boldsymbol{\alpha}^{(t+1)} - \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e\|^2. \tag{17}$$

The procedure is repeated until convergence.

- *Updating* $\boldsymbol{\alpha}$

The objective function of Problem (16) can be decomposed as the sum of the smooth convex function $g(\boldsymbol{\alpha})$ and a nonsmooth convex function $1_{\mathcal{C}}(\boldsymbol{\alpha})$, where $g(\boldsymbol{\alpha}) = \varphi(\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha} - \mathbf{I}_1^T\mathbf{A}^{(t)}\mathbf{I}_2 e\|^2$.

The projected gradient method can be used here for conduct the optimization. The gradient of the smooth part of the loss function $g(\boldsymbol{\alpha})$ is $\frac{\partial g}{\partial \boldsymbol{\alpha}} = \mathbf{M}\boldsymbol{\alpha} - e^T - 2\lambda(\boldsymbol{\alpha} - \mathbf{I}_1^T\mathbf{A}^{(t)}\mathbf{I}_2 e)$. Thus, the updating rule for $\boldsymbol{\alpha}$ can be written as

$$\boldsymbol{\alpha}^{(t+1)} \leftarrow \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{\alpha}^{(t)} + \eta\frac{\partial g}{\partial \boldsymbol{\alpha}}\right), \tag{18}$$

where $\eta > 0$ is the step size and $\mathcal{P}_{\mathcal{C}}$ is the euclidean projection onto the set $\mathcal{C} = \{\boldsymbol{\alpha} | 0 \le \alpha_{ij} \le C, \forall i,j\}$ which is defined by the box constraints.

- *Updating* $\mathbf{A}$

It has been extensively studied in the literature for optimizing the problems similar to Problem (17). The solution can be given by SVD, as shown by the following theorem:

Theorem 1 [40]. *For any $\rho \ge 0$ and matrix $\mathbf{X}$, the solution of the following problem is given by*

$$\mathcal{D}_\rho(\mathbf{X}) = \arg\min_{\mathbf{A}} \rho\|\mathbf{A}\|_* + \frac{1}{2}\|\mathbf{X} - \mathbf{A}\|_F^2,$$

where $\mathcal{D}_\rho(\mathbf{X})$ is the singular value thresholding operator which is defined as $\mathcal{D}_\rho(\mathbf{X}) = \mathbf{U}\mathcal{D}_\rho(\Sigma)\mathbf{V}^T$, where $\mathbf{U}$, $\mathbf{V}$ and Σ are given by the SVD of the input matrix $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ and $\mathcal{D}_\rho(\Sigma) = \mathrm{diag}(\{[\sigma_i - \rho]_+\})$ is the singular value shrinkage operator. Here σ_i is the ith singular value in Σ and $[x]_+ = \max(0, x)$ is the hinge function.

Based on Theorem (1), it is easy to show that the solution to Problem (17) can be written as

$$\begin{aligned} \mathbf{A}^{(t+1)} &= \arg\min_{\mathbf{A}} \gamma\|\mathbf{A}\|_* + \lambda\|\boldsymbol{\alpha}^{(t+1)} - \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e\|^2 \\ &= \arg\min_{\mathbf{A}} \frac{\gamma}{2\lambda}\|\mathbf{A}\|_* + \frac{1}{2}\|\mathcal{P}(\boldsymbol{\alpha}^{(t+1)}) - \mathbf{A}\|_F^2 \\ &= \mathcal{D}_{\frac{\gamma}{2\lambda}}(\mathcal{P}(\boldsymbol{\alpha}^{(t+1)})), \end{aligned} \tag{19}$$

where the operator $\mathcal{P}$ is for rearranging the vector $\boldsymbol{\alpha}^{(t+1)}$ to a matrix according to Eq. (7).[3]

2. This is because the nuclear norm ball $\{X : \|X\|_* \le 1\}$ is the convex hull of the set of rank-one matrices with spectral norm bounded by one.

3. Note $\|\boldsymbol{\alpha}^{(t+1)} - \mathbf{I}_1^T\mathbf{A}\mathbf{I}_2 e\|^2 = \|\mathcal{P}(\boldsymbol{\alpha}^{(t+1)}) - \mathbf{A}\|_F^2$ because the elements in $\boldsymbol{\alpha}^{(t+1)}$ correspond one to one with the elements in $\mathbf{A}$.

TABLE 1
Statistics on OHSUMED, MQ2007, MQ2008, and MLSR-WEB10K

Data Set	#labeled docs	#queries	#preference pairs	# average shared docs among preference pairs per query
OHSUMED	16,140	106	582,588	30.6
MQ2007	69,623	1692	404,467	5.75
MQ2008	15,211	784	80,925	2.45
MLSR-WEB10K	1,200,192	10,000	52,639,827	32.18

Algorithm 2. Regularized Ranking SVM

Input: training data $\{y_i, q_i, \boldsymbol{x}_i\}_{i=1}^{N}$, the parameters C, γ, λ, η, T, and T'
Output: model parameters $\boldsymbol{w}$

$P \leftarrow \bigcup_{q=1}^{|Q|} P_q = \bigcup_{q=1}^{|Q|} \{(i,j) | q_i = q_j = q, y_i \succ y_j\}$, where Q is the set of training queries in D.
$\boldsymbol{\alpha}^{(0)} \leftarrow$ random values in $[0, C]$
$\mathbf{A}^{(0)} \leftarrow \mathcal{P}(\boldsymbol{\alpha}^{(0)})$ {rearrange $\boldsymbol{\alpha}$ to $\mathbf{A}$}
for $t = 1$ to T **do**
  for $t = 1$ to T' **do**
    $\boldsymbol{\alpha}^{(t)} \leftarrow \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{\alpha}^{(t-1)} + \eta \frac{\partial g}{\partial \boldsymbol{\alpha}}\right)$ {Eq. (18)}
  end for
  $\mathbf{A}^{(t)} \leftarrow \mathcal{D}_{\frac{\gamma}{2\lambda}}(\mathcal{P}(\boldsymbol{\alpha}^{(t)}))$ {Update $\mathbf{A}$, Eq. (19)}
end for
return $\boldsymbol{w} = \sum_{(i,j)\in P} \boldsymbol{\alpha}_{ij}^{(T)} (\boldsymbol{x}_i - \boldsymbol{x}_j)$

Algorithm 2 shows the optimization procedure of the Regularized Ranking SVM algorithm.

5.3 Discussions

Factorized Ranking SVM and Regularized Ranking SVM model the parameter interactions in the original Ranking SVM from different aspects, which makes them perform differently in real world tasks.

Factorized Ranking SVM makes use of a factorization-based method for approximating the parameter matrix. Similar methods have been widely used in many real applications, mainly due to its computational convenience and promising performances. One main limitation of Factorized Ranking SVM is that the underlying rank K needs to be predefined. It is challenging for users to estimate an optimal rank value in advance, especially in noisy cases. Moreover, in real learning to rank tasks, different queries could retrieve very different number of documents for labeling. Also, the different label distributions will generate different number of preference pairs. Ideally, we need to set different rank values for different training queries (correspond to different diagonal matrices). However, this is obviously unpractical in real learning to rank tasks. From this viewpoint, it seems very difficult for us to find an optimal global K for all training queries.

Regularized Ranking SVM makes use of nuclear norm for controlling the rank of the parameter matrix, which is a 'softer' approach than the factorization based methods. In experiments, we also observed that Regularized Ranking SVM will approximate the diagonal matrices $\mathbf{A}_l$'s with different ranks. The diagonal matrices with more dimensions (more preference pairs for the query) tend to be approximated with higher ranks.

Theoretically, it has been proved that the nuclear norm is a tight convex upper bound over the rank operator [40]. Thus, the loss function of Regularized Ranking SVM is convex and has global optimality, which makes Regularized Ranking SVM generally converging fast and achieving stable performances. In our experiments, we also found that Regularized Ranking SVM often converges to its optimal solution in a few iterations. One limitation of Regularized Ranking SVM is the requirement of repeated SVD computation, which is time consuming and unaffordable in large-scale tasks. Though many efforts have been made towards accelerating SVD computation, the computational efficiency is still a big issue for the real application of Regularized Ranking SVM.

The parameter interaction is a common phenomenon in pairwise learning to rank. In this paper we use the widely used Ranking SVM algorithm as an example and showed two approaches to modeling the parameter interactions. The techniques is not limited to Ranking SVM. They can be easily adapted and applied to model the parameter interactions in other pairwise learning to rank algorithms.

6 Experiments

6.1 Experiment Settings

We conducted experiments to test the performances of the proposed Factorized Ranking SVM and Regularized Ranking SVM based on LETOR benchmark datasets [37] and the Microsoft Learning to Rank dataset, including OHSUMED, MQ2007, MQ2008, and MSLR-WEB10K. Each dataset consists of queries, corresponding retrieved documents, and human judged labels. Statistics on the datasets are given in Table 1. The number of preference pairs for each dataset is also shown in the table.

Following the configuration of these datasets [41], we conducted 5-fold cross-validation experiments for choosing the hyper parameters. That is, each dataset was randomly spit into five even subsets, denoted as S1, S2, S3, S4, and S5. At each trail of the cross-validation, three subsets (e.g., S1, S2, and S3) were combined and used for training the standard model, one subset (e.g., S4) was used for validation, and one subset (e.g., S5) was used for testing. The process was repeated 5 times and the results reported were the average over the five folds. In all of the experiments, the standard features provided by LETOR and MSLR-WEB10K were used. As for evaluation measures, mean average precision (MAP) and normalized discounted cumulative gain (NDCG) [42] at position of 1, 3, and 5 were used in our experiments.

We compared the proposed Factorized Ranking SVM (Fac-RSVM) and Regularized Ranking SVM (Reg-RSVM) with several state-of-the-art baseline methods, including the

TABLE 2
Ranking Accuracies on Dataset OHSUMED

Method	MAP	NDCG@1	NDCG@3	NDCG@5
RankNet	0.404	0.4007	0.3616	0.3388
ListNet	0.4443	0.5134	0.4664	0.4530
RSVM	0.4427	0.5289	0.4553	0.4392
LambdaMART	0.4096	0.5166	0.4443	0.4366
XGBoost	0.4301	0.5325	0.4758	0.4485
Fac-RSVM	**0.4518***†	**0.5671***†	**0.5113***†	**0.4889***†
($\pm$std)	($\pm$0.057)	($\pm$0.100)	($\pm$0.087)	($\pm$0.045)
Reg-RSVM	0.4480*†	0.5553*†	0.4946*†	0.4723*†
($\pm$std)	($\pm$0.063)	($\pm$0.063)	($\pm$0.089)	($\pm$0.072)

TABLE 3
Ranking Accuracies on Dataset MQ2007

Method	MAP	NDCG@1	NDCG@3	NDCG@5
RankNet	0.4184	0.3527	0.3599	0.3660
ListNet	0.4466	0.3897	0.3897	0.3956
RSVM	0.4442	0.3821	0.3796	0.3890
LambdaMART	**0.4680**	**0.4134**	**0.4201**	0.4201
XGBoost	0.4678	0.4097	0.4191	**0.4237**
Fac-RSVM	0.4645*†	0.4098*†	0.4103*†	0.4179*†
($\pm$std)	($\pm$0.017)	($\pm$0.028)	($\pm$0.036)	($\pm$0.029)
Reg-RSVM	0.4500*†	0.3728†	0.3871†	0.3947†
($\pm$std)	($\pm$0.015)	($\pm$0.023)	($\pm$0.026)	($\pm$0.024)

TABLE 4
Ranking Accuracies on Dataset MQ2008

Method	MAP	NDCG@1	NDCG@3	NDCG@5
RankNet	0.4522	0.3410	0.3991	0.4500
ListNet	0.4415	0.3244	0.3916	0.4396
RSVM	0.4713	0.3686	0.4277	0.4730
LambdaMART	0.4731	0.3622	0.4299	0.4714
XGBoost	**0.4790**	**0.3839**	0.4293	**0.4764**
Fac-RSVM	0.4723†	0.3690†	**0.4361***†	**0.4764**†
($\pm$std)	($\pm$0.043)	($\pm$0.050)	($\pm$0.046)	($\pm$0.048)
Reg-RSVM	0.4755*†	0.3690†	0.4315*†	0.4719†
($\pm$std)	($\pm$0.039)	($\pm$0.034)	($\pm$0.034)	($\pm$0.060)

TABLE 5
Ranking Accuracies on Dataset MSLR-WEB10K

Method	MAP	NDCG@1	NDCG@3	NDCG@5
RankNet	0.5739	0.2540	0.2755	0.2890
ListNet	0.5698	0.3047	0.3224	0.3348
RSVM	N/A	N/A	N/A	N/A
LambdaMART	0.6153	0.3623	0.3773	0.3889
XGBoost	**0.6249**	**0.4202**	**0.4230**	**0.4309**
Fac-RSVM	0.5875†	0.2891†	0.3205†	0.3219†
($\pm$std)	($\pm$0.003)	($\pm$0.006)	($\pm$0.003)	($\pm$0.003)
Reg-RSVM	0.5819†	0.2651†	0.2932†	0.3051†
($\pm$std)	($\pm$0.002)	($\pm$0.005)	($\pm$0.004)	($\pm$0.006)

conventional Ranking SVM (RSVM) [3], [4], RankNet [5], ListNet [15], LambdaMART [17], and XGBoost [18]. For Ranking SVM, we used the implementation released in [43][4] in all of the experiments. For RankNet, ListNet, and LambdaMART, we used the implementations in RankLib.[5] For XGBoost, we used the public available package.[6]

Factorized Ranking SVM and Regularized Ranking SVM have some parameters to tune, e.g., the number of hidden dimensions K, the tradeoff parameters C, γ, and λ, the number of iterations T, and the learning rate parameter η. A grid search strategy combined with manual search [44] is performed for optimizing the parameters. The parameters were respectively tuned in ranges of $K \in [1, 20], C \in [0.1, 10], \gamma \in [0.1, 50], \lambda \in [0.1, 10], \eta \in [0.001, 0.1]$, and $T \in [100, 5000]$.

6.2 Experimental Results

The experimental results on OHSUMED, MQ2007, MQ2008, and MSLR-WEB10K are reported in Tables 2, 3, 4, and 5, respectively. Boldface indicates the highest score among all runs. For Fac-RSVM and Reg-RSVM, the standard deviations are reported.

From the results, we can see that both Fac-RSVM and Reg-RSVM can outperform the linear baselines, including RankNet, ListNet and RSVM, in all of the datasets in terms of all of the evaluation measures (except Reg-RSVM on MQ2007 in terms of NDCG, and Reg-RSVM on MQ2008 in terms of NDCG@5). The results indicate that Fac-RSVM and Reg-RSVM can improve the learning to rank baselines through modeling the parameter interactions. For the non-linear baseline, including LambaMART and XGBoost, our model gained a better performance on the OHSUMED dataset achieved comparable results over the MQ2007 and MQ2008 dataset.

We also noticed that the improvements on OHSUMED are more significant than that of on MQ2007 and MQ2008. We analyzed the reasons through comparing the number of preference per query in these datasets. From the statistics shown in Table 1, we can see that compared with MQ2007 and MQ2008, the OHSUMED has much more labeled documents per query. On average one OHSUMED query has 152.3 (16,140 labeled documents for 106 queries) while one MQ2007 query and one MQ2008 query have only 41.2 and 19.4 labeled documents, respectively. The small number of labeled documents per query leads small number of preference pairs per query (the 3rd column in Table 1), which further results in a very small number of shared documents among the preference pairs (the 4th column in Table 1). Since the phenomenon of parameters interaction occurs only when one document is shared in different preference pairs, we can know that the parameter interaction in the Ranking SVM models trained on OHSUMED is much stronger than the models trained on MQ2007 and MQ2008. The analysis show that Fac-RSVM and Reg-RSVM can effectively improve the RSVM on the ranking tasks that each of the training query has a large number of labeled documents.

Table 5 reported experimental results on a larger dataset MSLR-WEB10K, which including 10,000 queries, 1,200,192 labeled documents and 52,639,827 preference pairs. The results of Ranking SVM is not available due to the large number of training queries. From the results, we can see that our

4. http://svmlight.joachims.org
5. http://pepple.cs.umass.edu/~vdang/ranklib.html
6. https://github.com/dmlc/XGBoost

TABLE 6
Ranks of Sampled $\mathbf{A}_l$'s in Fac-RSVM, Reg-RSVM, and RSVM Models, Each Corresponds to a Query

QID	#labeled docs/ #pairs	Rank (Fac-RSVM)	Rank (Reg-RSVM)	Rank (RSVM)
49	109/420	5	5	8
45	205/7482	5	6	92
35	320/27153	5	15	234

methods, both Fac-RSVM and Reg-RSVM, can outperform the pair-wise linear model RankNet, but are weaker than XGBoost and LambdaMART. The results indicate that LambdaMART takes the ranking evaluation measures into consideration (e.g., the calculation of λ in LambdaMART), which helps to boost the ranking performances. Another reason could be that LambdaMART and XGBoost used nonlinear ranking models (boosted trees), which enables them to capture complex ranking patterns, especially when training data is large.

We conducted t-tests on the improvements of Reg-RSVM and Fac-RSVM over the baselines. The results indicated that some of the improvements are significant (p-value < 0.05). As shown in Tables 2, 3, 4, and 5, '*' and '†' respectively indicate the improvements over RSVM and RankNet are significant. The improvements over LambdaMART and XGBoost are not significant in the experiments.

6.3 Comparison of Fac-RSVM and Reg-RSVM

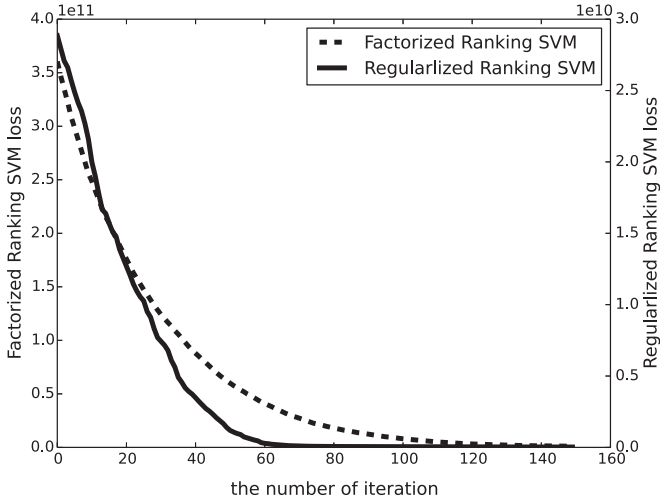
From the results reported in Tables 2, 3, 4, and 5 we can see that in general Fac-RSVM performed better than Reg-RSVM (except on MQ2008 in terms of MAP). The results indicate that the matrix factorization approach is more effective than the nuclear norm approach to modeling the low rank structure in the parameter matrix, though the matrix factorization approach has the limitation of assigning a common K to all of the sub-matrices.

For further analyzing Factorized Ranking SVM and Regularized Ranking SVM algorithms, we conducted more experiments to compare their performances, using the results on OHSUMED as examples.

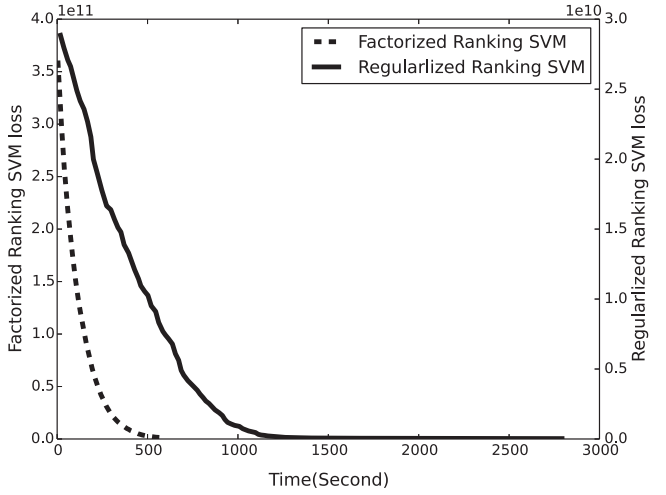
6.3.1 Controlling the Ranks

As have discussed in Section 5.3, Regularized Ranking SVM adopts a "softer" approach to controlling the ranks of each parameter matrix $\mathbf{A}_l$ than the Factorized Ranking SVM. Table 6 shows the ranks of three sampled $\mathbf{A}_l$'s in the Factorized Ranking SVM model, Regularized Ranking SVM model,[7] and the original Ranking SVM model. All of these models are trained on the first fold of the OHSUMED dataset. The number of the labeled documents as well as generated preference pairs are also shown. From the results, we can see that Regularized Ranking SVM assigned more rank to the queries that generated more preference pairs. The results indicate that Regularized Ranking SVM is suitable for the ranking tasks in which the training queries have very different number of labeled documents or preference pairs.

7. The ranks are calculated based on the results of singular value shrinkage operator at the final iteration (Line 9 of Algorithm 2).



(a) Losses w.r.t. iterations



(b) Losses w.r.t. time

Fig. 4. Learning curves of factorized ranking SVM and regularized ranking SVM.

6.3.2 Convergency

We also compared the convergency of the proposed Factorized Ranking SVM and Regularized Ranking SVM. Specifically, we recorded the values of the loss functions of these two algorithms at each of the training iteration. Fig. 4a plots the learning curves of Factorized Ranking SVM and Regularized Ranking SVM w.r.t. the iteration number. The experiments were also conducted on the first fold of the OHSUMED dataset. We can see that Regularized Ranking SVM converged after about 60 iterations while the Factorized Ranking SVM used more than 100 iterations before convergence. Note that in the figure we re-scaled the values of the two loss functions for fitting the two curves in the same figure. In Section 5.3, we discussed that the loss function of Regularized Ranking SVM is convex and has global optimality, which makes it converges fast in terms of the number of iterations. The experimental results verified the analysis.

One disadvantage of the Regularized Ranking SVM is the requirement of repeated SVD operations on $\mathbf{A}_l$'s at each iteration, which is time consuming. Fig. 4b showed the learning curve of these two algorithms w.r.t. the training time. The experiments were conducted in the same Linux server for

TABLE 7
Impact of K in Fac-RSVM

K	MAP	NDCG@1	NDCG@3	NDCG@5
1	0.3472	0.3939	**0.4250**	**0.3962**
3	0.3475	**0.4242**	0.4236	0.3877
5	**0.3486**	**0.4242**	0.4127	0.3860
10	0.3481	**0.4242**	0.4127	0.3801
20	0.3476	**0.4242**	0.4127	0.3877

making a fair comparison. From the results we can see that compared with Regularized Ranking SVM, Factorized Ranking SVM need less time to learn a ranking model, though it needed more iterations to converge.

6.3.3 Sensitivity of Hyper Parameters

We conducted experiments to analyze the sensitivity of the hyper parameters. The experiments were conducted based on the fold 1 of OHSUMED dataset. The key parameter K in Fac-RSVM, and λ and γ in Reg-RSVM were tuned.

The parameter K in Fac-RSVM determines the number of hidden dimensions. In the experiments, we set the parameter K with the values of $K \in \{1, 3, 5, 10, 20\}$. From the results reported in Table 7, we can see that the performances are stable. We can conclude that Fac-RSVM is robust to parameter K.

The parameters λ and γ in Reg-RSVM control the rank of the reshaped matrix A. We set these two parameters with the values of $\lambda \in \{0.1, 0.5, 1, 5, 10\}$ and $\gamma \in \{0.1, 0.5, 1, 5, 10, 50\}$, respectively. From the results shown in Table 8, we can see the best setting appear around at $\lambda = 0.1$ and $\gamma = 10$. Note that Reg-RSVM is robust to the parameter γ.

TABLE 8
Impact of γ and λ in Reg-RSVM

MAP (γ \ λ)	0.1	0.5	1	5	10
0.1	0.3530	0.3472	0.3491	0.3461	0.3417
0.5	0.3523	0.3416	0.3478	0.3461	0.3419
1	0.3489	0.3543	0.3491	0.3465	0.3418
5	0.3507	0.3540	0.3468	0.3458	0.3416
10	**0.3566**	0.3525	0.3490	0.3465	0.3414
50	0.3523	0.3512	0.3483	0.3459	0.3418

7 Conclusion

In this paper we investigated the phenomenon of parameter interactions in the pairwise learning to rank algorithm of Ranking SVM. We empirically found that there exists a low-rank structure among the Lagrange multipliers of the trained Ranking SVM models. Based on the discovery, we proposed to directly apply the low-rank constraint to the Lagrange multipliers of the Ranking SVM model, achieving two novel pairwise learning to rank algorithms, called Factorized Ranking SVM and Regularized Ranking SVM. Factorized Ranking SVM decomposes each Lagrange multiplier as a dot product of two low-dimensional vectors. Regularized Ranking SVM models the low ranking structure through adding a nuclear norm defined over the rearranged Lagrange multipliers. Efficient algorithms were developed to conduct the optimization problems. Experimental results based on publicly available benchmark datasets show that both Factorized Ranking SVM and Regularized Ranking SVM can outperform the state-of-the-art methods including Ranking SVM, RankNet, and ListNet.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (NSFC) under Grants No. 61872338, 61773362, 61425016, 61472401, 61722211, and the Youth Innovation Promotion Association CAS under Grants No. 20144310 and 2016102.

References

[1] H. Li, *Learning to Rank for Information Retrieval and Natural Language Processing*. San Mateo, CA, USA: Morgan & Claypool, 2011.
[2] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.
[3] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," in *Proc. Advances Large Margin Classifiers*, 2000, pp. 115–132.
[4] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 133–142.
[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 89–96.
[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, 2003.
[7] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking SVM to document retrieval," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 186–193.
[8] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.
[9] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, vol. 36, no. 3, pp. 1171–1220, 2008.
[10] G. Strang, *Introduction to Linear Algebra*, 4th ed. Cambridge, MA, USA: Wellesley-Cambridge, 2009.
[11] K. Crammer and Y. Singer, "Pranking with ranking," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 641–647.
[12] W. Chu and S. S. Keerthi, "New approaches to support vector ordinal regression," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 145–152.
[13] R. Nallapati, "Discriminative models for information retrieval," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2004, pp. 64–71.
[14] J. Xu, Y. Cao, H. Li, and Y. Huang, "Cost-sensitive learning of SVM for ranking," in *Proc. Eur. Conf. Mach. Learn.*, 2006, pp. 833–840.
[15] Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 129–136.
[16] J. Xu and H. Li, "AdaRank: A boosting algorithm for information retrieval," in *Proc. Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 391–398.
[17] C. J. C. Burges, "From RankNet to LambdaRank to LambdaMART: An overview," Jun. 2010, https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/
[18] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
[19] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma, "Directly optimizing evaluation measures in learning to rank," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 107–114.
[20] X. Zhao, X. Li, and Z. Zhang, "Joint structural learning to rank with deep linear feature learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2756–2769, Oct. 2015.
[21] Y. Yue and T. Joachims, "Predicting diverse subsets using structural SVMs," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1224–1231.
[22] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 784–791.

[23] Y. Zhu, Y. Lan, J. Guo, X. Cheng, and S. Niu, "Learning for search result diversification," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 293–302.
[24] R. Qiang, F. Liang, and J. Yang, "Exploiting ranking factorization machines for microblog retrieval," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 1783–1788.
[25] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.
[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 689–696.
[27] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 41, no. 6, pp. 391–407, 1990.
[28] Q. Wang, J. Xu, H. Li, and N. Craswell, "Regularized latent semantic indexing: A new approach to large-scale topic modeling," *ACM Trans. Inf. Syst.*, vol. 31, no. 1, pp. 5:1–5:44, 2013.
[29] Q. Wang, J. Xu, H. Li, and C. Nick, "Regularized latent semantic indexing," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 685–694.
[30] D. Achlioptas and F. Mcsherry, "Fast computation of low-rank matrix approximations," *J. ACM*, vol. 54, no. 2, Apr. 2007, Art. no. 9.
[31] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Comput. Sci.*, vol. 209, no. 1/2, pp. 237–260, 1998.
[32] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 2080–2088.
[33] I. Daubechies, M. Defrise, and C. De Mol , "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.
[34] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.
[35] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[36] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.
[37] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retrieval*, vol. 13, pp. 346–374, 2010.
[38] D. G. Luenberger, Y. Ye, et al., *Linear and Nonlinear Programming*, vol. 2. Berlin, Germany: Springer, 1984.
[39] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.
[40] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optimization*, vol. 20, no. 4, pp. 1956–1982, Mar. 2010.
[41] T. Qin and T.-Y. Liu, "Introducing LETOR 4.0 datasets," *CoRR*, vol. abs/1306.2597, 2013, http://arxiv.org/abs/1306.2597
[42] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2000, pp. 41–48.
[43] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184.
[44] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 473–480.

Jun Xu is a professor with the School of Information, Renmin University of China. His research interests focus on learning to rank and semantic matching in web search. He has published more than 50 papers in international conferences (e.g., SIGIR, WWW) and journals (e.g., the *ACM Transactions on Information Systems*, the *Journal of Machine Learning Research*). He serves as SPC for SIGIR, WWW, AAAI, and ACML, and an editorial board member for the *Journal of the Association for Information Science and Technology*. He has won the Best Paper Award in AIRS (2010) and Best Paper Runner-up in CIKM (2017). He is a member of the IEEE.

Wei Zeng is working toward the PhD degree in the Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS). Her major research interests focus on learning to rank for information retrieval.

Yanyan Lan is an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include machine learning and information retrieval. She has published more than 50 papers on ICML, NIPS, AAAI, SIGIR etc. She has won the best student paper in SIGIR 2012 and best paper runner-up in CIKM2017. She is a member of the IEEE.

Jiafeng Guo is a professor with the Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS). His major research interests include web search and data mining. He has published more than 50 papers in international journals and conferences, including the *IEEE Transactions on Knowledge and Data Engineering*, the *Physical Review E*, SIGIR etc. He has won the Best Paper Award in CIKM (2011) and Best Student Paper Award in SIGIR (2012). He is a member of the IEEE.

Xueqi Cheng is a professor with the Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS), and the director of the CAS Key Lab of Network Data Science and Technology. His main research interests include data science etc. He has published more than 100 publications in prestigious journals and conferences, including the *IEEE Transactions on Knowledge and Data Engineering*, the *Physical Review E*, SIGIR, WWW etc. He is a member of the IEEE.