# Extreme Learning to Rank via Low Rank Assumption

**Minhao Cheng** [1] **Ian Davidson** [1] **Cho-Jui Hsieh** [1 2]

## Abstract

We consider the setting where we wish to perform ranking for hundreds of thousands of users which is common in recommender systems and web search ranking. Learning a single ranking function is unlikely to capture the variability across all users while learning a ranking function for each person is time-consuming and requires large amounts of data from each user. To address this situation, we propose a Factorization RankSVM algorithm which learns a series of $k$ basic ranking functions and then constructs for each user a local ranking function that is a combination of them. We develop a fast algorithm to reduce the time complexity of gradient descent solver by exploiting the low-rank structure, and the resulting algorithm is much faster than existing methods. Furthermore, we prove that the generalization error of the proposed method can be significantly better than training individual RankSVMs. Finally, we present some interesting patterns in the principal ranking functions learned by our algorithms.

## 1. Introduction

Learning a ranking function based on pairwise comparisons has been studied extensively in recent years, with many successful applications in building search engines and other information retrieval tasks. Given a set of training instances with features $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ and pairwise comparisons, the goal is to find the optimal decision function $f(\cdot)$ such that $f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$ if $i$ is preferred over $j$. This is usually referred to as a learning-to-rank problem, and several algorithms have been proposed, including RankSVM (Herbrich et al., 1999), gradient boosting decision tree (Li et al., 2007), and many others (Cao et al., 2007; Yue et al., 2007; Negahban et al., 2012; Wauthier et al., 2013).

However, in many modern applications, a single global ranking is not sufficient to represent the variety of individual users preferences. For example, in movie ranking systems, it is preferable to learn an individual ranking function for each user since users' tastes can vary largely. The issue also arises in many other applications such as product recommendation, and personalized web search ranking.

Motivated by these real scenarios, we consider the problem of learning *hundreds of thousands* of ranking functions jointly, one for each user. Our target problem is different from collaborative ranking and BPR (Rendle et al., 2009; Wu et al., 2017a) since they only aim to recover ranking over existing items without using item features, while our goal is to obtain the ranking functions (taking item features as input) that can generalize to unseen items. This is also different from existing work on learning multiple ranking functions (i.e. (Qian et al., 2014)) because in that setting they are learning only several ranking functions. Here we focus on problems where the number of ranking functions $T$ is very large (e.g.,100K) but the amount of data to learn each ranking function is limited. The naive extensions of learning to rank algorithms fail since the training time grows dramatically, and also due to the over-fitting problem because of insufficient number of pairs for training.

To resolve this dilemma, we propose the Factorization RankSVM model for learning multiple ranking functions jointly. The main idea is to assume the $T$ ranking functions can be represented by a dictionary of $k$ ranking functions with $k \ll T$. In the linear RankSVM case, this assumption implies a low-rank structure when we stack all the $T$ linear hyper-planes together into a matrix. By exploiting this low rank structure, our algorithm can be efficient for both time and sample complexity.

Our contributions can be summarized as follows:

- We propose the Factorization RankSVM model for learning a large number of different ranking functions on different sets of data simultaneously. By exploiting the low-rank structure, we show that the gradient can be calculated very efficiently, and the resulting algorithm can scale to problems with large number of tasks.
- We derive the generalization error bound of our model,

[1] Department of Computer Science, University of California Davis, USA. [2] Department of Statistics, University of California Davis, USA . Correspondence to: Minhao Cheng <mhcheng@ucdavis.edu>.

*Proceedings of the $35^{th}$ International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

showing that by training all the $T$ tasks jointly, the sample complexity is much better than training individual rankSVMs under the low rank assumption.

- We conduct experiments on real world datasets, showing that the algorithm achieves higher accuracy and faster training time compared with state-of-the-art methods. This is a critical result as it shows the low rank ranking conjecture that underlies our research does occur.
- We further visualize the basic ranking functions learned by our algorithm, which has some interesting and meaningful patterns.

## 2. Related Work

**Learning to rank.** Given a set of pairwise comparisons between instances and the feature vectors associated with each instance, the goal of learning to rank is to discover the ranking function. There are three main categories of learning to rank algorithms: pointwise (Li et al., 2007), listwise (Cao et al., 2007; Yue et al., 2007), and pairwise methods (Herbrich et al., 2000; Cao et al., 2006). In this paper, we mainly focus on pairwise methods, which process a pair of documents or entries at a time. Among all the pairwise methods, rankSVM is a very popular one, so we choose it as our basic model.

We focus on the problem of solving $T$ learning-to-rank problems jointly when the problems share the same feature space and there is some hidden correlation between tasks. Obviously, we could apply existing learning-to-rank algorithms to solve each task independently, but this approach has several major drawbacks, as will be discussed in next section.

**Collaborative filtering and matrix factorization** Low-rank approximation has been widely used in matrix completion and collaborative filtering (Koren et al., 2009), and there are several extensions for matrix completion (Weimer et al., 2007). However, these methods cannot be applied in our setting, since our predictions are based on item features, and the corresponding items may not even appear in the training data. To conduct prediction based on item features, the inductive matrix factorization model has been recently proposed in (Jain & Dhillon, 2013; Xu et al., 2013), and factorization machine (Rendle, 2010) also uses a similar model. However, this model only allows input to be user-item ratings, not the pairwise comparisons used in our problem. In the experiments, we observe that even if the rating data is available, our model still outperforms inductive matrix completion significantly.

**Bayesian Personalized Ranking** (Rendle et al., 2009) proposes Bayesian Personalized Ranking(BPR) method to solve personalized ranking task. However, there are several major differences with our work. First, our target problem is different from BPR. We consider problems given both pairwise comparisons and "explicit" item features, and the goal is to learn the personalized ranking "functions" that can generalize to unseen items as long as we know their features. In comparison, the BPR does not take item features into account, and the goal of BPR is to recover the ranking among existing items. Also, the ranking cannot generalize to unseen items. Moreover, BPR considers implicit (0/1) feedback instead of explicit feedback.

**Collaborative Ranking** is another line of research that incorporates ranking loss in collaborative filtering. (Park et al., 2015; Weimer et al., 2007; Wu et al., 2017a) combines the ranking loss with matrix completion model, and (Yun et al., 2014) also uses a low-rank model with ranking loss given a binary observed matrix. However, similar to matrix completion and BPR, these collaborative ranking approaches do not use the item features. So they are not able to predict the preferences for unseen items. Also in this category, (Barjasteh et al., 2015) uses a trace norm to constraint ranking function, which is similar with our idea. However, they use implicit feedback which will lose certain information.

**Multi-task Learning:** has been extensively studied, especially in computer vision application. To model the shared information across tasks, a low-rank structure is widely assumed (Chen et al., 2012; 2009). (Hwang et al., 2011; Su et al., 2015) takes the attributes correlation as low-rank embeddings to learn SVM. However, our approach of learning basic ranking functions has not been discussed in the literature.

A summary of the differences between our algorithm with others are showed in Table 1.

## 3. Problem Setting

Our goal is to learn multiple ranking functions together, one for each user. Assume there are in total $T$ ranking functions to be learned (each one can be viewed as a task), and we are given pairwise comparisons for these ranking functions among $n$ items with features $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$. For each task $i$, the pairwise comparisons are denoted as $\Omega_i = \{(j,k)\}$, where $(j,k) \in \Omega_i$ means task $i$ compares item $j$ with $k$, and $y_{ijk} \in \{+1, -1\}$ is the observed outcome. For convenience, we use $\Omega$ to denote the union of all $\Omega_i$. Given these pairwise comparisons, we aim to learn a set of linear ranking functions $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T \in \mathbb{R}^d$ such that

$$\text{sign}(\boldsymbol{w}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_k)) \approx y_{ijk}, \;\; \forall (j,k) \in \Omega_i, \;\; \forall i = 1, \ldots, T$$

The only assumption we make for these $T$ ranking tasks is that the items involved in each task share the same feature space with $d$ features. Note that our algorithm allows each task has non-overlapping items—in that case we can still gather all the items together, and define $\Omega_i$ to be the comparisons within each task's own item block.

*Table 1.* The comparisons between different algorithms. RankSVM (Herbrich et al., 1999) is the algorithm for learning to rank. IMC (Jain & Dhillon, 2013; Rendle, 2010) stands for inductive matrix completion (a special case of factorization machine); CR (Park et al., 2015; Weimer et al., 2007; Wu et al., 2017a) stands for collaborative ranking; MF (Koren et al., 2009) stands for matrix factorization; BPR(Rendle et al., 2009) stands for Bayesian personalized ranking. S2COR (Barjasteh et al., 2015) stands for semi-supervised collaborative ranking.

| | Ours | rankSVM | IMC | CR | MF | BPR | S2COR |
|---|---|---|---|---|---|---|---|
| Using Item features | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Predict on new item | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Multiple users | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Pairwise data | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| Explicit feeback | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ |

This model can be easily deployed into recommendation systems where each user $i$ has a corresponding ranking function and the items could be movies, music, goods etc. Then the objective of the task is to learn a ranking function for each user $i$. Note that after obtaining $\boldsymbol{w}_i$ for each $i$, we can predict the preference for any pairs of items $\boldsymbol{x}_j, \boldsymbol{x}_k$ even when they are "*unseen items*" that are not in the training set. And most collaborative filtering approaches such as matrix completion cannot solve this problem. We are able to predict preferences on unseen items because we try to learn ranking functions based on features instead of just completing the rating matrix over "seen" items.

**Naive approaches:** For a single ranking function, (Herbrich et al., 1999) proposes the following RankSVM algorithm:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{(i,j,k)\in\Omega} \xi_{ijk}^2$$
$$\text{s.t. } y_{ijk}\boldsymbol{w}^T(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 1 - \xi_{ijk}, \quad \xi_{ijk} \geq 0. \ \forall i, j, k.$$

Here we use $L2$ hinge loss in our model, however it could be extended to $L1$ loss as well. We can take RankSVM into multiple-user case by simply assuming that all ranking functions share a common $\boldsymbol{w}$. We denote this method as RankSVM Jointly. (Evgeniou & Pontil, 2004) provides a variation by assuming each ranking function to be $\boldsymbol{w}_i = \boldsymbol{w} + \boldsymbol{v}_i$, where $\boldsymbol{w}$ is the centralized model and $\boldsymbol{v}_i$ is the task-dependent variance. However, this algorithm follows the strong assumption that $T$ ranking functions $\{\boldsymbol{w}_i\}_{i=1}^{d_1}$ are all close to a single base function $\boldsymbol{w}$. We call this algorithm RankSVM Var. This assumption is not always true in practice so that it will cause the model to under-fit training data (see our experimental results).

On the other hand, we can treat every user separately, which means we train every ranking function $\boldsymbol{w}_i$ independently by solving the following problem for every $i = 1, \ldots, T$:

$$\min_{\boldsymbol{w}_i} \frac{1}{2}\|\boldsymbol{w}_i\|^2 + C \sum_{(j,k)\in\Omega_i} \xi_{ijk}^2$$
$$\text{s.t. } \ y_{ijk}\boldsymbol{w}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 1 - \xi_{ijk}, \xi_{ijk} \geq 0, \forall (j,k) \in \Omega_i$$

We call this method as RankSVM Separately. It is obvious that this model has more freedom to fit the training data. However, due to the limited number of observed pairs $\Omega_i$ per user, each $\boldsymbol{w}_i$ has poor prediction quality due to over-fitting. We will analyze the sample complexity of RankSVM Separately in Section 4, and experimental results in Section 5 also support our analysis.

## 4. Proposed Algorithm

Our low rank personalized ranking conjecture assumes that all the $T$ ranking functions can be well-approximated by a linear combination of $k$ basic ranking functions, where $k \ll T$. This makes sense in many real applications; for example, in personalized recommender systems, there are group of users who have similar preferences. Let $\{\boldsymbol{u}_j\}_{j=1}^k$ to be the basic (linear) ranking functions, we can linearly combine weight then using $v_i$ to obtain a ranking function for user $i$ as follows: $\boldsymbol{w}_i = \sum_{j=1}^k v_{ij}\boldsymbol{u}_j$ for all $i$. This can be written as $W = UV^T$, where columns of $W, U$ are $\boldsymbol{w}_i$ and $\boldsymbol{u}_j$ respectively, and $V$ is the coefficients. Therefore, $W$ will be a rank-$k$ matrix, which leads to the following nuclear norm regularized problem to enforce the low-rankness of $W$:

$$\min_{W\in\mathbb{R}^{d\times T}} \ \|W\|_* + C \sum_{(i,j,k)\in\Omega} \xi_{ijk}^2$$
$$\text{s.t.} \quad \boldsymbol{y}_{ijk}\boldsymbol{w}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 1 - \xi_{ijk},$$
$$\xi_{ijk} \geq 0, \ \forall (i,j,k) \in \Omega.$$

where $||\cdot||_*$ is the nuclear norm of matrix, defined by summation of singular values. We could use some recent developed nuclear norm solvers to solve (4) (see (Cai et al., 2010; Hsieh & Olsen, 2014)).

While the nuclear norm regularized formulation is statistically near optimal for recovering the underlying low-rank model, it cannot be efficiently solved since there are $dT$ parameters in the problem. Therefore, we solve the following equivalent non-convex formulation:

$$\begin{aligned} \min_{U,V} \quad & C \sum_{(i,j,k)\in\Omega} \xi_{ijk}^2 + \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) \\ \text{s.t.} \quad & y_{ijk}\bar{\boldsymbol{v}}_i^T U^T(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 1 - \xi_{ijk}, \\ & \xi_{ijk} \geq 0, \forall (i,j,k) \in \Omega. \end{aligned} \tag{1}$$

where we replace the nuclear norm regularization in Equation (4) using the property $\|W\|_* = \min_{W=UV^T} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$, $U \in \mathbb{R}^{d\times k}$, $V \in \mathbb{R}^{T\times k}$, and $\bar{\boldsymbol{v}}_i^T$ is the $i$-th row of $V$. With this non-convex relaxation, there are only $(d+T)k$ parameters involved. So it is preferred over the convex form.

However, developing a fast solver for (1) is still nontrivial. Although RankSVM is often solved in the dual space using stochastic dual coordinate ascent (SDCA) (Shalev-Shwartz & Zhang, 2013), in our case, it is not suitable because there are $|\Omega|k$ dual variables, where each corresponds to one constraint. So applying a dual coordinate ascent will take $O(|\Omega k|)$ time complexity (to go through all dual variables) and the same order of memory complexity to store all of them. Therefore, we solve the problem in the primal space using alternating minimization. Instead of solving the constrained form, we solve the following equivalent unconstrained problem:

$$\min_{U,V} \left\{ C \sum_{(i,j,k)\in\Omega} \max(0, 1 - y_{ijk}\bar{\boldsymbol{v}}_i^T U^T(\boldsymbol{x}_j - \boldsymbol{x}_k))^2 + \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) \right\} := f(U,V). \tag{2}$$

Following the alternating minimization scheme, our algorithm iteratively updates one of $U, V$ while keeping the other one fixed. When updating $U$ with $V$ fixed, the subproblem becomes:

$$U = \operatorname*{argmin}_{U\in\mathbb{R}^{d\times k}} C \sum_{(i,j,k)\in\Omega} \max(0, 1 - y_{ijk}\bar{\boldsymbol{v}}_i^T U^T(\boldsymbol{x}_j - \boldsymbol{x}_k))^2 + \frac{1}{2}\|U\|_F^2. \tag{3}$$

To solve the problem in the primal space, the main bottleneck is the gradient computation when we apply gradient descent. The gradient can be written as

$$\nabla_U f(U,V) = U + \sum_{i=1}^{T} \sum_{(j,k)\in\Omega_i} -2Cy_{ijk} \max\left(0, 1 - y_{ijk}\bar{\boldsymbol{v}}_i^T U^T(\boldsymbol{x}_j - \boldsymbol{x}_k)\right)(\boldsymbol{x}_j - \boldsymbol{x}_k)\bar{\boldsymbol{v}}_i^T \tag{4}$$

Computing (4) naively takes $O(|\Omega|kd)$ time, since we need to go through the summation, and each term requires $O(kd)$ computing time for computing $(\boldsymbol{x}_j - \boldsymbol{x}_k)\bar{\boldsymbol{v}}_i^T$. However, by re-organizing the computation using a book-keeping technique, we are able to do this in $O(T\bar{n}k + dkn + |\Omega|)$ time, where $\bar{n}$ is the average number of ratings per user. The details are presented in Algorithm 1.

**Algorithm 1** Factorization RankSVM: Computing $\nabla_U f(U,V)$

```
Input: $X, V, \Omega, Y$
Compute $\boldsymbol{p}_j = U^T\boldsymbol{x}_j$ and set $\boldsymbol{z}_j = \mathbf{0}$ for all $j = 1, \ldots, n$
for $i = 1, 2, \ldots, T$ do
    Compute $\boldsymbol{q}_j = \bar{\boldsymbol{v}}_i^T\boldsymbol{p}_j$ for all $j \in \Omega_i$
    Set $s_j = 0$ for all $j \in \Omega_i$
    for $(j,k) \in \Omega_i$ do
        $s_j \leftarrow s_j - 2Cy_{ijk}\max(0, 1 - y_{ijk}(q_j - q_k))$
        $s_k \leftarrow s_k + 2Cy_{ijk}\max(0, 1 - y_{ijk}(q_j - q_k))$
    end for
    $\boldsymbol{z}_j \leftarrow \boldsymbol{z}_j + s_j\boldsymbol{v}_i$ for all $j \in \Omega_i$
end for
for $j = 1, 2, \ldots, n$ do
    $\nabla_U f(U,V) \leftarrow \nabla_U f(U,V) + \boldsymbol{x}_j\boldsymbol{z}_j^T$
end for
Output $\nabla_U f(U,V) + U$
```

For updating $V$, the objective function (1) can be decomposed into $T$ subproblems:

$$\bar{\boldsymbol{v}}_i = \operatorname*{argmin}_{\bar{\boldsymbol{v}}_i\in\mathbb{R}^k} C \sum_{(j,k)\in\Omega_i} \max(0, 1 - y_{ijk}\bar{\boldsymbol{v}}_i^T U^T(\boldsymbol{x}_j - \boldsymbol{x}_k))^2 + \frac{1}{2}\|\boldsymbol{v}_i\|_2^2, \tag{5}$$

where each of them is just an RankSVM problem that can be easily solved by gradient descent or Newton method (Chapelle & Keerthi, 2010). The details are omitted here, but the time complexity for this part is $O(T\bar{n}k + dkn + |\Omega|)$, which is exactly the same with the $U$ part.

To sum up, our algorithm has an overall time complexity $O(T\bar{n}k + dkn + |\Omega|)$ per iteration, which is quite small because the dominated term $|\Omega|$ (number of pairs) is separated from rest of the terms. Also, $k$ (rank) and $\bar{n}$ (averaged items involves in a ranking task) are usually small. Furthermore, we could adapt Newton method proposed by (Wu et al., 2017b) to further speed up the optimization. As a result, we are able to scaleto very large datasets.

## 5. Sample Complexity Analysis

Now we analyze the sample complexity of the proposed model. If we keep growing $T$ (number of ranking functions), under the low-rank assumption $\mathcal{W} = O(T^{1/2})$, the samples needed for Factorization RankSVM to achieve the same $\epsilon$-error is approximately $O(T^{1/2})$, which is much better than training $T$ individual RankSVMs which requires $O(T)$ samples. Detailed proofs can be found in the appendix.

**Sample complexity of our model**

Assume we observe a set of $(i, j, k)$ pairs and comparison results $y_{ijk} \in \{+1, -1\}$ from a fixed but unknown distribution. To recover the underlying model, we proposed to solve (4), and it is equivalent to the constraint form:

$$\begin{aligned} \hat{W} = &\arg\min_{W\in\mathbb{R}^{d\times T}} \sum_{(i,j,k)\in\Omega} \ell((I_i^T W^T(\boldsymbol{x}_j - \boldsymbol{x}_k)), y_{ijk}), \\ &\text{s.t.} \quad \|W\|_* \le \mathcal{W}, \end{aligned} \tag{6}$$

where $I \in \mathbb{R}^{T\times T}$ is the indicator matrix, each column $I_i$ is $[0, 0, ..., 1, 0, 0]$ where the $i$-th element equals to one. Without loss of generality, we assume $\|\boldsymbol{x}_j\| \le 1$ for all $j$. The prediction function we want to learn is

$$f_W(i, j, k) = I_i^T W^T(\boldsymbol{x}_j - \boldsymbol{x}_k) = \langle W, (\boldsymbol{x}_j - \boldsymbol{x}_k)I_i^T\rangle,$$

and in our formulation (6), we search within the function class $F_W := \{f_W : \|W\|_* \le \mathcal{W}\}$.

The quality of any ranking function $f_W$ can be measured by the following expected ranking error (where $\mathbf{1}(\cdot)$ is the indicator function):

$$R(f) := \mathbb{E}_{i,j,k}[\mathbf{1}\big(\text{sign}(f(i, j, k)) \ne \text{sign}(y_{ijk})\big)]. \tag{7}$$

We denote $R^* = \min_f R(f)$ to be the optimal risk we can get. Since optimizing 0/1 loss is hard, our algorithm uses a convex surrogate loss $\ell$, and the following concepts of $\ell$-risk will be used in our analysis:

- Expected $\ell$-risk: $R_\ell(f) = \mathbb{E}_{i,j,k}[\ell(f(i, j, k), y_{ijk})]$
- Empirical $\ell$-risk: $\hat{R}_\ell(f) = \frac{1}{m}\sum_{(i,j,k)\in\Omega} \ell(f(i, j, k), y_{ijk})$

We begin with the following lemma to bound the expected $\ell$-risk:

**Lemma 1** (Bound on Expected $\ell$-risk (Bartlett & Mendelson, 2002))**.** *Assume $\ell(\cdot,\cdot)$ is a loss function upper bounded by $\mathcal{B}$ and with Lipschitz constant $L_\ell$ with respect to its first argument. Let $\mathfrak{R}(F_W)$ be the model complexity of the function class $F_W$ (w.r.t $\Omega$ and associated with $\ell$) defined as:*

$$\mathfrak{R}(F_W) = \mathbb{E}_\sigma[\sup_{f\in F_W} \frac{1}{m} \sum_{(i,j,k)\in\Omega} \sigma_\alpha \ell(f(i, j, k), y_{ijk})], \tag{8}$$

*where each $\sigma_\alpha$ takes values $\{\pm 1\}$ with equal probability. Then with probability at least $1 - \delta$, for all $f \in F_W$, we have*

$$R_\ell(f) \le \hat{R}_\ell(f) + 2\mathbb{E}_\Omega[\mathfrak{R}(F_W)] + \mathcal{B}\sqrt{\frac{\log\frac{1}{\delta}}{2m}}.$$

To achieve an upper bound for $R_\ell(f)$, we derive a bound of the Radamacker complexity $\mathbb{E}_\Omega[\mathfrak{R}(F_W)]$:

**Lemma 2.** *The model complexity of* (6) *can be upper bounded by:*

$$\mathbb{E}_\Omega[\mathfrak{R}(F_W)] \le \min\left\{2L_\ell\mathcal{W}\sqrt{\frac{\log 2d}{m}}, \sqrt{\frac{9L_\ell\mathcal{B}C\mathcal{W}(\sqrt{T}+n)}{m}}\right\}, \tag{9}$$

*where $L_\ell$ is the Lipchitz constant of loss function and $C$ is a universal constant.*

With the above lemma, we now derive the following theorem to bound the expected ranking error:

**Theorem 1.** *With probability $1 - \delta$, the expected error of the optimal solution of our model* (6) *is:*

$$\begin{aligned} R(f_{\hat{W}}) - R^* \le &O(\hat{R}_\ell(f_{\hat{W}}) - R^*_\ell) + O(\mathcal{B}\sqrt{\frac{\log(1/\delta)}{m}}) \\ &+O(\frac{\min(\sqrt{\mathcal{W}\mathcal{B}(\sqrt{T}+n)}, \mathcal{W}\log d)}{\sqrt{m}}), \end{aligned} \tag{10}$$

*where $R^* = \inf_f R(f)$ and $R^*_\ell := \inf_f R_\ell(f)$.*

Note that all the hidden constants can be found in the appendix. In general, the first term on the right hand side will be small since $f_{\hat{W}}$ minimizes the empirical error. This is a standard generalization error bound (as shown in (Kakade et al., 2009)) that works for any distribution of $y_{ijk}$.

If we further assume the $y_{ijk}$ is generated from an unseen groudtruth $W^*$ with $\|\mathcal{W}^*\|_* \le \mathcal{W}$, then the following theorem shows that the error is small when $m$ goes to infinity:

**Lemma 3.** *If the observed $y_{ijk} = \boldsymbol{x}_j^T\boldsymbol{w}_i^* - \boldsymbol{x}_k^T\boldsymbol{w}_i^*$ for all $i, j, k$, and loss function satisfies $\ell(a, b) = 0$ if sign$(a)$ = sign$(b)$, then we have $R(f_{\hat{W}}) \le O(\frac{\min(\sqrt{\mathcal{W}\mathcal{B}(\sqrt{T}+n)}, \mathcal{W}\log d)}{\sqrt{m}}) + O(\mathcal{B}\sqrt{\frac{\log(1/\delta)}{m}})$.*

Note that the loss $\ell(a, y) = \max(-ay, 0)^2$ satisfies the assumption of Lemma 3, but in practice adding a margin will improve the performance (using $\ell(a, y) = \max(1-ay, 0)^2$). From Theorem 1 and Lemma 3, we can conclude that the error of our model decreases roughly with $1/\sqrt{m}$ ($m$ is number of samples), and increases with $\sqrt{\mathcal{W}}$ (nuclear norm of the underlying model).

**Comparison with** RANKSVM SEPARATELY**.**

Training $T$ independent RankSVMs separately can also achieve arbitrary small $\epsilon$ error under similar condition, so the main question is whether our model can reduce the number of samples $m$ needed. In RANKSVM SEPARATELY, it is equivalent to solving problem (6) with the constraint replaced by $\|\boldsymbol{w}_i\| \le w$ for all $i$. Assume there are $m/T$ pairs per ranking function, then we can prove the following sample complexity based on standard analysis from (Kakade et al., 2009):

**Lemma 4.** *Under the same condition of Lemma 3, the* RANKSVM SEPARATELY *solution $\tilde{f}$ satisfies*

$$R(\tilde{f}) = O(\frac{w}{\sqrt{m/T}}) + O(\mathcal{B}\sqrt{\frac{\log(1/\delta)}{m/T}}).$$

Note that we assume $\mathcal{W} := \|W^*\|_*$ and $w := \max_i \|W^*_{:,i}\|$ where $W^*$ is the underlying matrix. Clearly, if the nuclear norm $\mathcal{W}$ is small constant, our sample complexity (Lemma 3) is much better than the bound for RANKSVM SEPARATELY (Lemma 4), since our dependency to $T$ is $O(T^{1/4})$ while it is $O(\sqrt{T})$ for rankSVM. Moreover, in another setting (see, for example, (Shamir & Shalev-Shwartz, 2014)), if each element of $W^*$ is bounded and rank of $W^*$ is a constant, $\mathcal{W} = O(\sqrt{Td})$ and $w = O(\sqrt{d})$, our bound in Lemma 3 is still better than Lemma 4. Although a better sample complexity upper bound doesnt directly imply our method is always better, however, by obtaining a smaller Rademacher complexity, it is clear that our formulation has benefits to achieve a tighter upper bounds, which leads to better performance in practice.

## 6. Experimental Results

In this section, we show our method outperforms other algorithms on both synthetic and real datasets. All the experiments are conducted on a server with an Intel E7-4820 CPU and 256G memory.

**Experimental Setting.** For each ranking task, we randomly split the items into training items and testing items. In the training phase, we use all the pairs between training items to train the model, and in the testing phase we evaluate the prediction accuracy for all the testing-testing item pairs and testing-training item pairs, which is similar with BPR (Rendle et al., 2009). The accuracy is defined to be the correctly predicted pairs divided by total number of predicted pairs.

We mainly compare our algorithm with RANKSVM JOINTLY (training a single rankSVM model), RANKSVM SEPARATELY (training an individual rankSVM model for each task), and RANKSVM VAR (the multi-task rankSVM model proposed in (Evgeniou & Pontil, 2004)). All the algorithm above are using square hinge loss in the experiments. We choose the best regularization parameter for each method by a validation set.

**Synthetic Data.** For synthetic dataset, we assume there are $1,000$ tasks, $10,000$ items and each item has $64$ features. The underlying ranking models are generated by $W^* = U^*(V^*)^T$, where $U^* \in \mathbb{R}^{64\times 20}, V^* \in \mathbb{R}^{1000,20}$, and $U, V \sim \mathcal{N}(0,1)$. The feature matrix is generated by $X \in \mathbb{R}^{64\times 10,000}$, $X \sim \mathcal{N}(0,1)$. We sample 800 pairs for each user as training data, with labels based on underlying rating $R = (W^*)^T X$.

*Table 2.* Comparisons on synthetic data. Ours-$k$ is (Factorization RankSVM) with rank $k$.

| | TIME/EPOCH | TRAIN ACC | TEST ACC |
|---|---|---|---|
| RANKSVM JOT | 0.08 | 0.52 | 0.508 |
| RANKSVM SEP | 0.13 | 0.999 | 0.808 |
| RANKSVM VAR | 0.18 | 0.964 | 0.719 |
| OURS-10 | 0.07 | 0.872 | 0.820 |
| OURS-20 | 0.10 | 0.999 | **0.964** |
| OURS-30 | 0.17 | 1.000 | 0.943 |

*Table 3.* Statistics of datasets. $d$ is the dimension of item features.

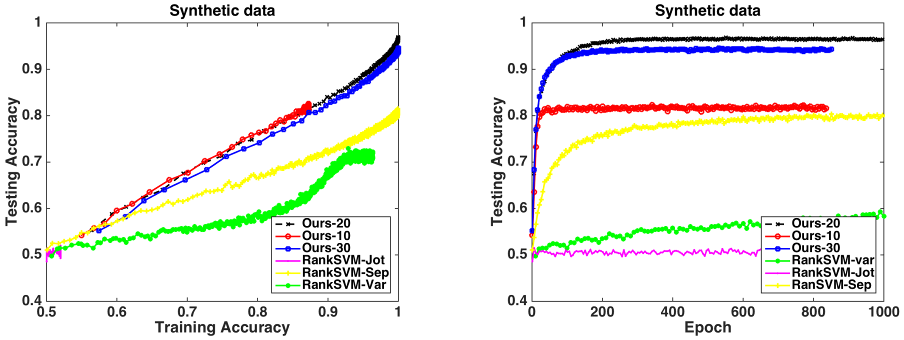
| DATASET | USERS | ITEMS | $d$ |
|---|---|---|---|
| SYNTHETIC | 1000 | 10000 | 64 |
| YAHOO! MOVIES | 7,642 | 106,954 | 194,697 |
| HETREC2011-2K | 2,133 | 10,197 | 173 |
| MOVIELENS 20M | 138,493 | 27,278 | 15,603 |

Table 2 shows that our algorithms outperform other rankSVM algorithms on synthetic datasets. Also, as showed in Figure 1, We observe that RANKSVM JOINTLY suffers from under-fitting (low training and test accuracy). On the other hand, RANKSVM SEPARATELY has the over-fitting problem (high training accuracy but low test accuracy) since it does not have enough samples for learning each individual task. Since the underlying $U, V$ have rank 20, our model with rank 20 performs the best. However, even if we choose rank to be 10 or 30, our model still significantly outperforms the other models.

**Real World Datasets.** We use recommender system as an application to compare our algorithm with other ranking algorithms. Each user is treated as a "ranking task", and the observed pairs are generated from training ratings. Note that we are also given item features $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, and the goal is to learn a personalized ranking model $\boldsymbol{w}_i$ for each user. The testing items are unseen in the training phase, which is different from classical matrix completion problem—the goal of classical matrix completion is to complete the matrix, while our goal is to learn the function that can generalize to unseen items. The only matrix completion work that can utilize the feature information to predict unseen items is inductive matrix completion (Jain & Dhillon, 2013) (IMC), which is a special case of factorization machine (Rendle, 2010). Although they do not allow pairwise comparisons as input, for the completeness of comparison, we still include them into comparison and give them the original rating data as input.

We choose three datasets in our real-world application experiments: (1) *Yahoo! Movies User Ratings and Descriptive Content Information V1_0*[1] (2) *HetRec2011-MovieLens-2K* (Cantador et al., 2011). (3) *MovieLens 20M Dataset* (Harper

[1] http://research.yahoo.com/Academic_Relations



(a) Testing accuracy versus training accuracy

(b) Testing accuracy versus epoch

*Figure 1.* Testing accuracy comparisons with synthetic data

& Konstan, 2016). For the first dataset, we use the title and abstract of each movie and combine them as the feature matrix $X$. For the second and third datasets, we take the genres information of each movie as features. See Table3 for more information.

The results for datasets (1) and (2) are presented in Table4. Clearly, our method outperforms other algorithms both in accuracy and in speed. Note that dataset (1) has dense features and dataset (2) has sparse features, and our algorithm performs well in both cases. For dataset (3), there are more than 100,000 ranking tasks and other algorithms take more than 1000 seconds per epoch. However, our algorithm only takes about 100 seconds per epoch, and converges to a solution with 63.4% testing accuracy.

We also plot the time vs accuracy curves in Figure Our algorithms consistently get better accuracy compared to all other methods. Note that sometimes RANKSVM JOINTLY is fast in the beginning, but eventually it cannot converge to a good solution.

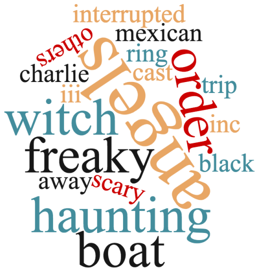


*Figure 2.* Visualization of the basic ranking function learned by our algorithm.

### 6.1. Feature Embedding

**Visualize basic ranking functions.** Finally, we visualize the basic ranking functions learned by our model. We take the *Yahoo! movie* dataset, where each feature corresponds to a word in movie title and abstract. We select a basic ranking function (a column of $U$) from our model and show the top 25 features with most positive weights and bottom 25 features with most negative weights in Figure 2, 3. The visualization of ranking function clearly demonstrates interesting common patterns of users' tastes.

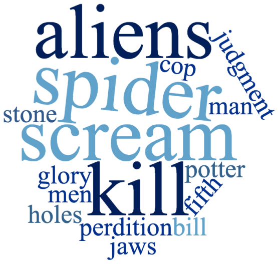


*Figure 3.* Visualization of the basic ranking function learned by our algorithm.

## 7. Conclusions

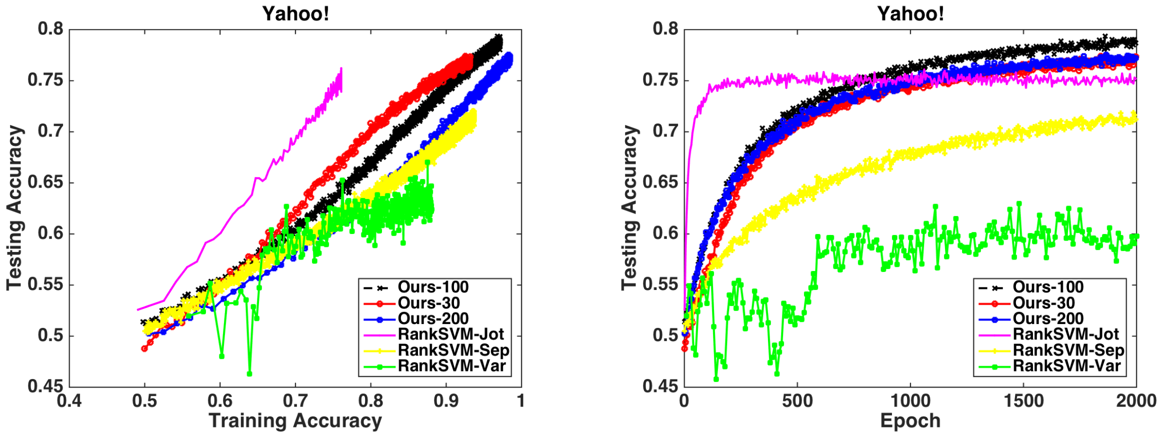
We propose a new algorithm for learning multiple ranking functions based on the combination of RankSVM and matrix factorization. We show that the model can be solved efficiently, has good statistical guarantee, and outperforms other methods on real datasets in both training time and prediction accuracy. Our algorithm can be used in many online personalized ranking systems. An interesting direction is to introduce non-linearity (e.g., neural networks) in the feature side of our model and learn $U, V$ with neural network weights jointly.

## Acknowledgments

Cho-Jui Hsieh and Minhao Cheng acknowledge the support of NSF via IIS-1719097 and the computing resources provided by Google cloud and Nvidia.

*Table 4.* The comparisons on real world recommender system datasets. Our algorithm is faster and more accurate than other ranking-based approaches. Since IMC is a totally different algorithm and has a different notion of epoch, we do not include its time per epoch here. Method-x means we use rank x for the method.

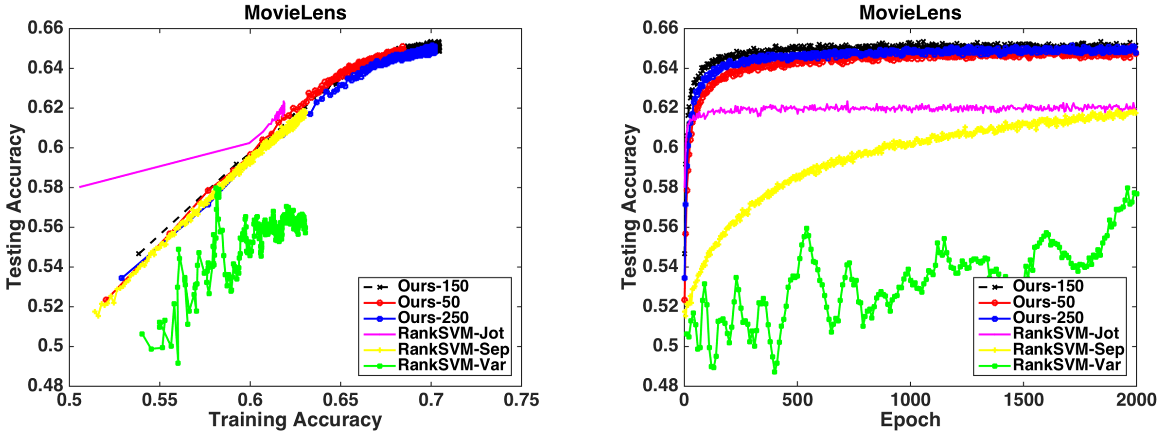
| Yahoo-music | | | | MovieLens | | | |
|---|---|---|---|---|---|---|---|
| Method | Time per epoch | Train Acc | Test Acc | Method | Time per epoch | Train Acc | Test Acc |
| RankSVM jot | 85 | 0.761 | 0.750 | RankSVM jot | 7.1 | 0.618 | 0.619 |
| RankSVM sep | 130 | 0.999 | 0.715 | RankSVM sep | 8.0 | 0.630 | 0.617 |
| RankSVM var | 309 | 0.881 | 0.642 | RankSVM var | 50 | 0.629 | 0.565 |
| IMC-100 | - | 0.997 | 0.802 | IMC-150 | - | 0.535 | 0.532 |
| Ours-30 | **1.10** | 0.932 | 0.770 | Ours-50 | **7.0** | 0.684 | 0.648 |
| Ours-100 | 2.85 | 0.999 | **0.804** | Ours-150 | 7.5 | 0.704 | **0.651** |
| Ours-200 | 5.40 | 0.999 | 0.775 | Ours-250 | 8.7 | 0.706 | 0.650 |



(a) Testing accuracy versus training accuracy

(b) Testing accuracy versus epoch

*Figure 4.* Testing accuracy comparisons with Yahoo!



(a) Testing accuracy versus training accuracy

(b) Testing accuracy versus epoch

*Figure 5.* Testing accuracy comparisons with HetRec2011-MovieLens-2K

## References

Barjasteh, I., Forsati, R., Esfahanian, A.-H., and Radha, H. Semi-supervised collaborative ranking with push at top. *arXiv preprint arXiv:1511.05266*, 2015.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Cai, J.-F., Candès, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., and Hon, H.-W. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 186–193. ACM, 2006.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136. ACM, 2007.

Chapelle, O. and Keerthi, S. S. Efficient algorithms for ranking with svms. *Information Retrieval*, 13(3):201–215, 2010.

Chen, J., Tang, L., Liu, J., and Ye, J. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 137–144. ACM, 2009.

Chen, J., Liu, J., and Ye, J. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):22, 2012.

Evgeniou, T. and Pontil, M. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 109–117. ACM, 2004.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

Herbrich, R., Graepel, T., and Obermayer, K. Support vector learning for ordinal regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pp. 97–102. IET, 1999.

Herbrich, R., Graepel, T., and Obermayer, K. Large margin rank boundaries for ordinal regression. 2000.

Hsieh, C.-J. and Olsen, P. A. Nuclear norm minimization via active subspace selection. In *ICML*, 2014.

Hwang, S. J., Sha, F., and Grauman, K. Sharing features between objects and their attributes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1761–1768. IEEE, 2011.

Jain, P. and Dhillon, I. S. Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626*, 2013.

Kakade, S. M., Sridharan, K., and Tewari, A. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems*, pp. 793–800, 2009.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Li, P., Burges, C. J., Wu, Q., Platt, J., Koller, D., Singer, Y., and Roweis, S. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, volume 7, pp. 845–852, 2007.

Negahban, S., Oh, S., and Shah, D. Iterative ranking from pair-wise comparisons. In *Advances in Neural Information Processing Systems*, pp. 2474–2482, 2012.

Park, D., Neeman, J., Zhang, J., Sanghavi, S., and Dhillon, I. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *International Conference on Machine Learning*, pp. 1907–1916, 2015.

Qian, B., Wang, X., Cao, N., Jiang, Y.-G., and Davidson, I. Learning multiple relative attributes with humans in the loop. *IEEE Transactions on Image Processing*, 23(12): 5573–5585, 2014.

Rendle, S. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 995–1000. IEEE, 2010.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pp. 452–461. AUAI Press, 2009.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Shamir, O. and Shalev-Shwartz, S. Matrix completion with the trace norm: learning, bounding, and transducing. *Journal of Machine Learning Research*, 15(1):3401–3423, 2014.

Su, C., Yang, F., Zhang, S., Tian, Q., Davis, L. S., and Gao, W. Multi-task learning with low rank attribute embedding for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3739–3747, 2015.

Wauthier, F. L., Jordan, M. I., and Jojic, N. Efficient ranking from pairwise comparisons. *ICML (3)*, 28:109–117, 2013.

Weimer, M., Karatzoglou, A., Le, Q. V., and Smola, A. Maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, pp. 1–8, 2007.

Wu, L., Hsieh, C.-J., and Sharpnack, J. Large-scale collaborative ranking in near-linear time. In *KDD*, 2017a.

Wu, L., Hsieh, C.-J., and Sharpnack, J. Large-scale collaborative ranking in near-linear time. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 515–524. ACM, 2017b.

Xu, M., Jin, R., and Zhou, Z.-H. Speedup matrix completion with side information: Application to multi-label learning. In *Advances in Neural Information Processing Systems*, pp. 2301–2309, 2013.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 271–278. ACM, 2007.

Yun, H., Raman, P., and Vishwanathan, S. Ranking via robust binary classification. In *Advances in Neural Information Processing Systems*, pp. 2582–2590, 2014.