Preference Completion: Large-scale Collaborative Ranking from Pairwise Comparisons

Dohyung Park DHPARK@UTEXAS.EDU
Joe Neeman JOENEEMAN@GMAIL.COM
Jin Zhang ZJ@UTEXAS.EDU
Sujay Sanghavi SANGHAVI@MAIL.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
The University of Texas at Austin

Abstract

In this paper we consider the collaborative ranking setting: a pool of users each provides a small number of pairwise preferences between d possible items; from these we need to predict each users preferences for items they have not yet seen. We do so by fitting a rank r score matrix to the pairwise data, and provide two main contributions:
(a) we show that an algorithm based on convex optimization provides good generalization guarantees once each user provides as few as $O(r \log^2 d)$ pairwise comparisons – essentially matching the sample complexity required in the related matrix completion setting (which uses actual numerical as opposed to pairwise information), and
(b) we develop a large-scale non-convex implementation, which we call AltSVM, that trains a factored form of the matrix via alternating minimization (which we show reduces to alternating SVM problems), and scales and parallelizes very well to large problem settings. It also outperforms common baselines on many moderately large popular collaborative filtering datasets in both NDCG and in other measures of ranking performance.

1. Introduction

This paper considers the following recommendation system problem: given a set of items, a set of users, and non-numerical *pairwise comparison* data, find the underlying preference ordering of the users. In particular, we are interested in the setting where data is of the form "user i preferes item j over item k", for different ordered user-item-item triples i, j, k. Pairwise preference data is wide-spread; indeed, almost any setting where a user is presented with a menu of options – and chooses one of them – can be considered to be providing a pairwise preference between the chosen item and every other item that is presented.

Crucially, we are interested in the collaborative filtering setting, where *(a)* on the one hand the number of such pairwise preferences we have for any one user is woefully insufficient to infer anything for that user in isolation; and *(b)* on the other hand, we aim for *personalization*, i.e. for every user to possibly have different inferred preferences from every other. To reconcile these two requirements, our method relates the preferences of users to each other via a low-rank matrix, which we (implicitly) assume governs the observed preferences. Essentially, we fit a low-rank users $\times$ items *score matrix* X to pairwise comparison data by trying to ensure that $X_{ij} - X_{ik}$ is positive when user i prefers item j to item k.

Our contributions: We present two algorithms to infer the score matrix X from training data; once inferred, this can be used for predicting future preferences. While there has been some recent work on fitting low-rank score matrices to pairwise preference data (which we review and compare to below), in this paper we present the following two contributions:
(a) *A statistical analysis for the convex relaxation:* we bound the *generalization error* of the solution to our convex program. Essentially, we show that the minimizer of the empirical loss also almost minimizes the true expected loss. We also give a lower bound showing that our error rate is sharp up to logarithmic factors.
(b) *A large-scale non-convex implementation:* We provide a non-convex algorithm that we call Alternating Support

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Vector Machine (AltSVM). This non-convex algorithm is more practical than the convex program in a large-scale setting; it explicitly parameterizes the low-rank matrix in factored form and minimizes the hinge loss. Crucially, each step in this algorithm can be formulated as a standard SVM that updates one of the two factors; the algorithm proceeds by alternating updates to both factors. We apply a stochastic version of dual coordinate descent (Hsieh et al., 2008; Shalev-Shwartz & Zhang, 2013) with lock-free parallelization. This exploits the problem structure and ensures it parallelizes well. We show that our algorithm outperforms several existing collaborative ranking algorithms in both speed and prediction accuracy, and it achieves significant speedups as the number of cores increases.

1.1. Related Work

Ranking/learning preferences is a classical problem that has been considered in a large amount of work. There are many different settings for this problem, which we discuss below.

Learning to Rank The main problem in this community has been to estimate a ranking function from given feature vectors and relevance scores. Depending on its application, a feature vector may correpond to a user-item pair or a single item. While there have been algorithms that use pairwise comparisons (Herbrich et al., 2000; Joachims, 2002) of the training samples, our setting is different in that our data consists *only* of pairwise comparisons. We refer the reader to the survey (Liu, 2009).

One ranking with pairwise comparisions In a single-user model, we are asked to learn a single ranking given pairwise comparisons. Jamieson & Nowak (2011a) and Ailon (2011) consider an active query model with noiseless responses; Jamieson & Nowak (2011b) give an algorithm for exactly recovering the true ranking under a low-rank assumption similar to ours, while Ailon (2011) approximately recovers the true ranking without such an assumption. Wauthier et al. (2013) and Negahban et al. (2012) learn a ranking from noisy pairwise comparisions; Negahban et al. (2012) consider a Bradley-Terry-Luce model similar to ours and attempt to learn an underlying score vector, while Wauthier et al. (2013) get by without structure assumptions, but only attempt to learn the ranking itself. Hajek et al. (2014) considered a problem to learn a single ranking given a more generalized partial rankings from the Plackett-Luce model and provided a minimax-optimal algorithm.

Many rankings with pairwise comparisions Given multiple users with different rankings, one could of course attempt to learn their rankings by simply applying an algorithm from the previous section to each user individually. However, it is more efficient – both statistically and computationally – to postulate some global structure and use it to relate the many users' rankings. This is the same idea that has been applied so successfully in collaborative filtering. Rendle et al. (2009) and Liu et al. (2009) were the first to take this approach. They modeled the observations as coming from a BTL model with low-rank structure (i.e., very similar to our model) and gave algorithms for learning the model parameters. Yi et al. (2013) took a purely optimization-based approach. Rather than assuming a probabilistic model, they minimized a convex objective using the hinge loss on a low-rank matrix. In a slightly different model, Hu et al. (2008) and Shi et al. (2013) consider the problem of learning from latent feedback. Recently, Lu & Negahban (2014) analyzed an algorithm which is very similar to ours for the Bradley-Terry-Luce model independently from our work.

Many rankings with 1-bit ratings Instead of moving to pairwise comparisons, some work has suggested avoiding the difficulties of numerical ratings by instead asking users to give 1-bit ratings to items; that is, each user only indicates whether they like or dislike an item. In this setting, the work of Davenport et al. (2014) is most closely related to ours, in that they assume an underlying low-rank structure and give an algorithm based on convex optimization. Also, our theoretical analysis owes a lot to their work. Xu et al. (2013) consider a slightly different goal: rather than attempting to recover the preferences of each user, they try to cluster similar users and similar items together. Yun et al. (2014) proposed an optimization problem motivated from robust binary classification and used stochastic gradient descent to solve the problem in a large-scale setting.

Many rankings with numerical ratings The goal in this setting is the same as ours, except that the data is in the form of numerical ratings instead of pairwise comparisons. Weimer et al. (2007) attempted to directly optimize Normalized Discounted Cumulative Gain (NDCG), a widely used performance measure for ranking problems. Balakrishnan & Chopra (2012), and Volkovs & Zemel (2012) converted this problem into a learning-to-rank problem and solved it using the existing algorithms. While these works considered the low-rank matrix model, different models are proposed by Weston et al. (2012) and Lee et al. (2014). Weston et al. (2012) proposed a tensor model to rank items for different queries and users, and (Lee et al., 2014) proposed a weighted sum of low-rank matrix models.

2. Empirical Risk Minimization (ERM)

Let us first formulate the problem mathematically. The task is to estimate rankings of multiple users on multiple items.

We denote the numbers of users by d_1, and the number of items by d_2. We are given a set of triples $\Omega \subset [d_1] \times [d_2] \times [d_2]$, where the preference of user i between items j and k is observed if $(i, j, k) \in \Omega$. The observed comparison is then given by $\{Y_{ijk} \in \{1, -1\} : (i, j, k) \in \Omega\}$ where $Y_{ijk} = 1$ if user i prefers item j over item k, and $Y_{ijk} = -1$ otherwise. Let $\Omega_i = \{(j, k) : (i, j, k) \in \Omega\}$ denote the set of item pairs that user i has compared.

We predict rankings for multiple users by estimating a score matrix $X \in \mathbb{R}^{d_1 \times d_2}$ such that $X_{ij} > X_{ik}$ means that user i prefers item j over item k. Then the sorting order for each row provides the predicted ranking for the corresponding user.

We propose (as have others) that X is low-rank or close to low-rank, the intuition being that each user bases their preferences on a small set of features that are common among all the items. Then the empirical risk minimization (ERM) framework can naturally be formulated as

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})) \qquad (1) \\ \text{subject to} \quad & \operatorname{rank}(X) \le r \end{aligned}$$

where $\mathcal{L}(\cdot)$ is a monotonically non-increasing loss function which induces $X_{ij} > X_{ik}$ if $Y_{ijk} = 1$, and $X_{ij} < X_{ik}$ otherwise. (e.g., hinge loss, logistic regression loss, etc.)

Solving (1) is NP-hard because of the rank constraint. As a first alternative, we propose a straightforward convex relaxation.

3. Convex Relaxation

Our first method is the convex relaxation of (1), which involves a nuclear norm constraint.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})) \qquad (2) \\ \text{subject to} \quad & \|X\|_* \le \sqrt{\lambda d_1 d_2} \end{aligned}$$

Here, for any matrix X, the nuclear/trace norm $\|X\|_*$ denotes the sum of its singular values; it is a well-recognized convex surrogate for low-rank structure (most famously in matrix completion).

The only parameter of this algorithm is λ, which governs the trade-off between better optimizing the likelihood of the observed data, and the strictness in imposing approximate low-rank structure. Since we motivated our algorithm with the assumption that X has low rank, we should point out how our algorithm's parameter λ compares to the rank: note that if X is a $d_1 \times d_2$ rank-r matrix whose largest absolute entry is bounded by C then $\|X\|_* \le \sqrt{r}\|X\|_F \le C\sqrt{r d_1 d_2}$. In other words, λ is a parameter that takes into account both the rank of X and the size of its elements, and it is roughly proportional to the rank.

3.1. Analytic results

We analyze (2) by assuming a standard model for pairwise comparisons. Then we provide a statistical guarantee of the method under the model.

Recall the classical Bradley-Terry-Luce model (Bradley & Terry, 1952; Luce, 1959) for pairwise preferences of a single user, which assumes that the probability of item j being preferred over k is given by a logistic of the difference of the underlying preference scores of the two items. For multiple users, we assume that there is some true score matrix $X^* \in \mathbb{R}^{d_1 \times d_2}$ and

$$\Pr(Y_{ijk} = 1) \;=\; \frac{\exp(X^*_{ij} - X^*_{ik})}{1 + \exp(X^*_{ij} - X^*_{ik})}.$$

Assume that each user-item-item triple (i, j, k) independently belongs to Ω with probability $p_{i,j,k}$, and let $m = \sum_{i,j,k} p_{i,j,k}$ be the expected size of Ω. We will assume that the $p_{i,j,k}$ are approximately balanced in the sense that no user-item pair is observed too frequently:

Assumption 3.1. *There is a constant $\kappa > 0$ such that for every i, j,*

$$\sum_k p_{i,j,k} \le \kappa \frac{m}{d_1 d_2}.$$

Note that if $\kappa = 1$ in Assumption 3.1 then the $p_{i,j,k}$ are all equal, meaning that each user-item-item triple has an equal chance to be observed.

In order to state our error bounds, we first introduce some notation: let $\mathbb{P}_X$ be the distribution of $\{Y_{i,j,k} : 1 \le i \le d_1, 1 \le j < k \le d_2\}$ (i.e. the complete distribution of all pairwise preferences, even those that are not observed).

Our main upper bound shows that if m is sufficiently large then our algorithm finds a solution with almost minimal risk. Given a loss function $\mathcal{L}$, define the expected risk of X by

$$R(X) = \frac{1}{d_1 d_2^2} \sum_{i=1}^{d_1} \sum_{j,k=1}^{d_2} \mathbb{E}_{X^*} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})),$$

where the expectation is with respect to the distribution parametrized by the true parameters X^*.

Theorem 3.1. *Suppose that $\mathcal{L}$ is 1-Lipschitz, and let Y and Ω be distributed as $\mathbb{P}_{X^*}$ for some $d_1 \times d_2$ matrix X^*. Under Assumption 3.1,*

$$\mathbb{E}R(\hat{X}) \le \inf_{\{X : \|X\|_* \le \sqrt{\lambda d_1 d_2}\}} \mathbb{E}R(X) + C\kappa\sqrt{\frac{\lambda(d_1 + d_2)}{m}} \log(d_1 + d_2),$$

where C is a universal constant.

We recall that the parameter λ is related to rank in that if X is a $d_1 \times d_2$ rank-r matrix whose largest absolute entry is bounded by C then $\|X\|_* \leq \sqrt{r}\|X\|_F \leq C\sqrt{rd_1d_2}$. In other words, λ is a parameter that takes into account both the rank of X^* and the size of its elements, and it is roughly proportional to the rank. In particular, Theorem 3.1 shows that once we observe $m \sim r(d_1+d_2)\log^2(d_1+d_2)$ pairwise comparisons, then we can accurately estimate the probability of any user preferring any item over any other. In other words, we need to observe about $r(1+d_2/d_1)\log^2(d_1+d_2)$ comparisons per user, which is substantially less than the $rd_2\log(d_2)$ comparisons that we would have required if each user were modelled in isolation. Moreover, our lower bound (below) shows that at least $r(1+d_2/d_1)$ comparisons per user are required, which is only a logarithmic factor from the upper bound.

Theorem 3.2. *Suppose that $\mathcal{L}'(0) < 0$. Let $\mathcal{A}$ be any algorithm that receives $\{Y_{i,j,k} : (i,j,k) \in \Omega\}$ as input and produces $\hat{X}$ as output. For any $\lambda \geq 1$ and $m \geq d_1+d_2$, there exists X^* with $\|X^*\|_* \leq \sqrt{\lambda d_1 d_2}$ such that when Y and Ω are distributed according to $\mathbb{P}_{X^*}$ then with probability at least $\frac{1}{2}$,*

$$\mathbb{E}R(\hat{X}) \geq R(X^*) + c\min\left\{1, \sqrt{\frac{\lambda(d_1+d_2)}{m}}\right\},$$

where $c > 0$ is a constant depending only on $\mathcal{L}$.

Together, Theorems 3.1 and 3.2 show that (up to logarithmic factors) if X^* has rank r then about $r(1+d_2/d_1)$ comparisons per user are necessary and sufficient for learning the users' preferences.

3.1.1. Maximum likelihood estimation of X^*

By specializing the loss function $\mathcal{L}$, Theorem 3.1 has a simple corollary for maximum-likelihood estimation of X^*. Recall that if μ and ν are two probability distributions on a finite set S the the Kullback-Leibler divergence between them is

$$D(\mu\|\nu) = \sum_{s\in S}\mu(s)\log\frac{\mu(s)}{\nu(s)},$$

under the convention that $0\log 0 = 0$. We recall that although $D(\cdot\|\cdot)$ is not a metric it is always non-negative, and that $D(\mu\|\nu) = 0$ implies $\mu = \nu$.

Corollary 3.3. *Let Y and Ω be distributed as $\mathbb{P}_{X^*}$ for some $d_1 \times d_2$ matrix X^*. Define the loss function $\mathcal{L}$ by $\mathcal{L}(z) = \log(1+\exp(z)) - z$. Under Assumption 3.1,*

$$\frac{1}{d_1d_2^2}\sup_{\{X:\|X\|_*\leq\sqrt{\lambda d_1 d_2}\}} D(\mathbb{P}_{X^*}\|\mathbb{P}_{\hat{X}}) - D(\mathbb{P}_{X^*}\|\mathbb{P}_X) \leq C\kappa\sqrt{\frac{\lambda(d_1+d_2)}{m}}\log(d_1+d_2),$$

where C is a universal constant.

Note that the loss function in Corollary 3.3 is exactly the negative logarithm of the logistic function, and so $\hat{X}$ in Corollary 3.3 is the maximum-likelihood estimate for X^*. Thus, Corollary 3.3 shows that the distribution induced by the maximum-likelihood estimator is close to the true distribution in Kullback-Leibler divergence.

4. Large-scale Non-convex Implementation

While the convex relaxation is statistically near optimal, it is not ideal for large-scale datasets because it requires the solution of a convex program with $d_1 \times d_2$ variables. In this section we develop a non-convex variant which both scales and parallelizes very well, and has better empirical performance as compared to several existing empirical baseline methods.

Our approach is based on the following steps:

1. We represent the low-rank matrix in explicit factored form $X = UV^\top$ and replace the regularizer appropriately. This results in a non-convex optimization problem in $U \in \mathbb{R}^{d_1\times r}$ and $V \in \mathbb{R}^{d_2\times r}$, where r is the rank parameter.
2. We solve the non-convex problem by alternating between updating U while keeping V fixed, and vice versa. With the hinge loss (which we found works best in experiments), each of these becomes an SVM problem - hence we call our algorithm AltSVM.
3. The problem is of course not symmetric in U and V because users rank items but not vice versa. For the U update, each user vector naturally decouples and can be done in parallel (and in fact just reduces to the case of rankSVM (Joachims, 2002)).
4. For the V update, we show that this can *also* be made into an SVM problem; however it involves coupling of all item vectors, and all user ratings. We employ several tricks (detailed below) to speed up and effectively parallelize this step.

The non-convex problem can be written as

$$\min_{U,V}\sum_{(i,j,k)\in\Omega}\mathcal{L}(Y_{ijk}\cdot u_i^\top(v_j - v_k)) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{3}$$

where we replace the nuclear norm regularizer using the property $\|X\|_* = \min_{X=UV^\top} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$ (Srebro et al., 2004). $u_i^\top$ and $v_i^\top$ denote the ith rows of U and V, respectively. While this is a non-convex algorithm for which it is hard to find the global optimum, it is computationally more efficient since only $(d_1 + d_2)r$ variables are involved. We propose to use L2 hinge loss, i.e., $\mathcal{L}(x) = \max(0, 1 - x)^2$.

In the alternating minimization of (3), the subproblem for U is to solve

$$U \leftarrow \arg\min_{U \in \mathbb{R}^{d_1 \times r}} \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk} \cdot u_i^\top (v_j - v_j)) + \frac{\lambda}{2}\|U\|_F^2, \tag{4}$$

while V is fixed. This can be decomposed into n independent problems for u_i's where each solves for

$$u_i \leftarrow \arg\min_{u \in \mathbb{R}^r} \frac{\lambda}{2}\|u\|_2^2 + \sum_{(j,k)\in\Omega_i} \mathcal{L}(Y_{ijk} \cdot u^\top (v_j - v_k). \tag{5}$$

This part is in general a small-scale problem as the dimension is r, and the sample size is $|\Omega_i|$ for each user i.

On the other hand, solving for V with fixed U can be written as

$$V \leftarrow \arg\min_{V \in \mathbb{R}^{d_2 \times r}} \left\{ \frac{\lambda}{2}\|V\|_F^2 + \sum_{(i,j,k)\in\Omega} \mathcal{L}(\langle V, A^{(u,i,j)} \rangle) \right\} \tag{6}$$

where $A^{(i,j,k)} \in \mathbb{R}^{d_2 \times r}$ is such that the lth row of $A^{(i,j,k)}$ is $Y_{ijk} \cdot u_i^\top$ if $l = j$, $-Y_{ijk} \cdot u_i^\top$ if $l = k$, and 0 otherwise. It is a much larger SVM problem than (5) as the dimension is $d_2 r$ and the sample size is $|\Omega|$.

We note that the feature matrices $\{A^{(i,j,k)} : (i,j,k) \in \Omega\}$ are highly sparse since in each feature matrix only $2r$ out of the $d_2 r$ elements are nonzero. This motivates us to apply the stochastic dual coordinate descent algorithm (Hsieh et al., 2008; Shalev-Shwartz & Zhang, 2013), which not only converges fast but also takes advantages of feature sparsity in linear SVMs. Each coordinate descent step takes $O(r)$ computation, and iterations over $|\Omega|$ coordinates provide linear convergence (Shalev-Shwartz & Zhang, 2013).

Now we describe the dual problems of our two subproblems explicitly. Let $\alpha \in \mathbb{R}^{|\Omega_i|}$ denote the dual vector for (5), in which each coordinate is denoted by α_{ijk} where $(j,k) \in \Omega_i$. Then the dual problem of (5) is to solve

$$\min_{\alpha \in \mathbb{R}^{|\Omega_i|}, \alpha \geq 0} \frac{1}{2} \left\| \sum_{(j,k)\in\Omega_i} \alpha_{ijk} Y_{ijk} (v_j - v_k) \right\|_2^2 + \frac{1}{\lambda} \sum_{(j,k)\in\Omega_i} \mathcal{L}^*(-\lambda \alpha_{ijk}) \tag{7}$$

where $\mathcal{L}^*(z)$ is the convex conjugate of $\mathcal{L}$. At each coordinate descent step for α_{ijk}, we find the value of α_{ijk} minimizing (7) while all the other variables are fixed. If we maintain $u_i = \sum_{(j,k)\in\Omega_i} \alpha_{ijk} Y_{ijk} (v_j - v_k)$, then the coordinate descent step is simply to find δ^* minimizing

$$\frac{1}{2}\|u_i + \delta^* Y_{ijk}(v_j - v_k)\|_2^2 + \frac{1}{\lambda}\mathcal{L}^*(-\lambda(\alpha_{ijk} + \delta^*)) \tag{8}$$

and update $\alpha_{ijk} \leftarrow \alpha_{ijk} + \delta^*$.

The dual problem of (6) is to solve

$$\min_{\beta \in \mathbb{R}^{|\Omega|}, \beta \geq 0} \frac{1}{2} \left\| \sum_{(i,j,k)\in\Omega} \beta_{ijk} A^{(i,j,k)} \right\|_F^2 + \frac{1}{\lambda} \sum_{(i,j,k)\in\Omega} \mathcal{L}^*(-\lambda \beta_{ijk}) \tag{9}$$

where β is the dual vector for the subproblem (6). Similarly to α_{ijk}, the coordinate descent step for β_{ijk} is to replace β_{ijk} by $\beta_{ijk} + \delta^*$ where δ^* minimizes

$$\frac{1}{2}\left(\|v_j + \delta^* Y_{ijk} u_i\|_2^2 + \|v_k - \delta^* Y_{ijk} u_i\|_2^2\right) + \mathcal{L}^*(-\lambda(\beta_{ijk} + \delta^*)), \tag{10}$$

and maintain $V = \sum_{(i,j,k)\in\Omega} \beta_{ijk} Y_{ijk} A^{(i,j,k)}$.

The detailed description of AltSVM is presented in Algorithm 1. In each subproblem, we run the stochastic dual coordinate descent, in which a pairwise comparison $(i,j,k) \in \Omega$ is chosen uniformly at random, and the dual coordinate descent for α_{ijk} or β_{ijk} is computed. We note that each coordinate descent step takes the same $O(r)$ computational cost in both subproblems, while the subproblem sizes are much different.

4.1. Parallelization

For each subproblem, we parallelize the stochastic dual coordinate descent algorithm asynchronously without locking. Given T processors, each processor randomly sample a triple $(i,j,k) \in \Omega$ and update the corresponding dual variable and the user or item vectors. We note that this update is for a sparse subset of the parameters. In the user part, a coordinate descent step for one sample updates

Algorithm 1 Alternating Support Vector Machine (AltSVM)

Require: Ω, $\{Y_{ijk} : (i,j,k) \in \Omega\}$, and $\lambda \in \mathbb{R}^+$
Ensure: $U \in \mathbb{R}^{d_1 \times r}$, $V \in \mathbb{R}^{d_2 \times r}$
1: Initialize U, and set $\alpha, \beta \leftarrow 0 \in \mathbb{R}^{|\Omega|}$
2: **while** not converged **do**
3: $v_j \leftarrow \sum_{(i,j,k)\in\Omega} \beta_{ijk} Y_{ijk} u_i - \sum_{(i,k,j)\in\Omega} \beta_{ikj} Y_{ikj} u_i, \forall j \in [d_2]$
4: **for all threads** $t = 1, \ldots, T$ **in parallel do**
5: **for** $s = 1, \ldots, S$ **do**
6: Choose $(i,j,k) \in \Omega$ uniformly at random
7: Find δ^* minimizing (10).
8: $\beta_{ijk} \leftarrow \beta_{ijk} + \delta^*$
9: $v_j \leftarrow v_j + \delta^* Y_{ijk} u_i$
10: $v_k \leftarrow v_k - \delta^* Y_{ijk} u_i$
11: **end for**
12: **end for**
13: $u_i \leftarrow \sum_{(i,j,k)\in\Omega} \alpha_{ijk} Y_{ijk} (v_j - v_k), \forall i \in [d_1]$
14: **for all threads** $t = 1, \ldots, T$ **in parallel do**
15: **for** $s = 1, \ldots, S$ **do**
16: Choose $(i,j,k) \in \Omega$ uniformly at random.
17: Find δ^* minimizing (8).
18: $\alpha_{ijk} \leftarrow \alpha_{ijk} + \delta^*$
19: $u_i \leftarrow u_i + \delta^* Y_{ijk} (v_j - v_k)$
20: **end for**
21: **end for**
22: **end while**

only r out of the rd_1 variables. In the item part, one coordinate descent step for a sample update only $2r$ out of the rd_2 variables. This motivates us not to lock the variables when updated, so that we ignore the conflicts. This lock-free parallelism is shown to be effective in (Niu et al., 2011) for stochastic gradient descent (SGD) on the sum of sparse functions. Moreover, in (Hsieh et al., 2015), it is also shown that the stochastic dual coordinate descent scales well without locking. We implemented the algorithm using the OpenMP framework. In our implementations, we also parallelized steps 3 and 13 of Algorithm 1. We show in the next section that our proposed algorithm scales up favorably.

4.2. Remark on the implementation

In Algorithm 1, the subproblem for V comes first, and then it solves for the user vectors U. We empirically observed that this order gives better convergence on practical datasets. We also note that each subproblem reuses the dual variables in the previous outer iteration. When almost converged, the features (V for solving U, and U for solving V) do not change too much. By reusing the dual variables in the previous iteration we can start with a feasible solution close to the optimum.

5. Experimental results

5.1. Pairwise data

We used the MovieLens 100k dataset, which contains 100,000 ratings given by 943 users on 1682 movies. The ratings are given as integers from one to five, but we converted them into preference data by declaring that a user preferred one movie to another if they gave it a higher rating (if two movies received the same rating, we treated it as though the user did not provide a preference). Then we held out 20% of the data as a test set.

We compared our algorithm to the following two:

- Bayesian Personalized Ranking (BPR) (Rendle et al., 2009): This algorithm is based on a similar model to ours, but a different optimization procedure (essentially, a variant of stochastic gradient descent).
- Matrix completion from pairwise differences : A standard matrix completion algorithm that observes – for various triples $(i,j,k) \in \Omega$ – the difference between user i's ratings for item j and item k. Note that this algorithm has an advantage over (2) because it sees the magnitude of this difference instead of only its sign. Nevertheless, the matrix completion algorithm does not perform any better than (2). A similar phenomenon was also observed in (Davenport et al., 2014).

We evaluate our performance by computing the proportion of pairwise comparisons in the test set $\mathcal{T}$ for which we correctly infer the user's preference.

$$(\text{Prediction error}) = \frac{1}{|\mathcal{T}|} \sum_{(i,j,k)\in\mathcal{T}, Y_{ijk}=1} \mathbb{I}(X_{ij} > X_{ik})$$

This is similar to the AUC statistic measured by Rendle et al. (Rendle et al., 2009), and if the data were fully observed then it would measure Kendall's distance between each user's true preferences and the learned ones. However, our main reason for choosing this measure of performance is that, as an average accuracy over all pairwise comparisons, it resembles the quantity that we study in our theoretical bounds.

Unsurprisingly, we were more accurate at correctly inferring strong preferences; therefore, we have also shown the accuracy obtained by only measuring performance on pairs whose rankings differ by two or more. Both the methods we considered do measurably better at predicting these orderings.

5.2. Large-scale experiments on rating data

Now we demonstrate that our algorithm performs well as a collaborative ranking method on rating data. We used

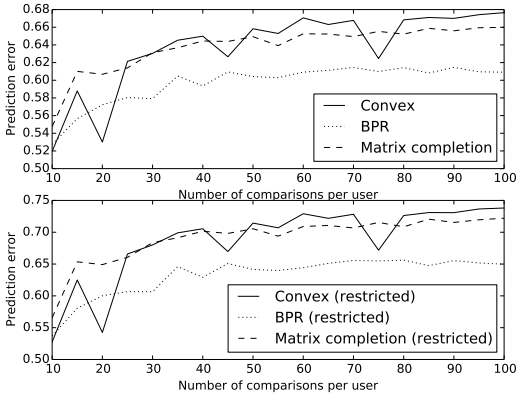


Figure 1. Prediction accuracy on the MovieLens 100k dataset, for different numbers of observed comparisions per user. For the "restricted" plots, only the pairs with a rating difference of two or more were used for evaluation.

the datasets specified in Table 1. Given a training set of ratings for each user, our algorithm will only use non-tying pairwise comparisons from the set, while other competing algorithms use the ratings themselves. Hence, they have more information than ours. The competing algorithms are those with publicly available codes provided by the authors.

- CofiRank (Weimer et al., 2007)[1] This algorithm uses alternating minimization to directly optimize NDCG.
- Local Collaborative Ranking (LCR) (Lee et al., 2014)[2] : The main idea is to predict preferences from the weighted sum of multiple low-rank matrices model.
- RobiRank (Yun et al., 2014)[3] : This algorithm uses stochastic gradient descent to optimize the loss function motivated from robust binary classification.
- Global Ranking : To see the effect of personalized ranking, we compare the results with a global ranking of the items. We fixed U to all ones and solved for V.

[1] http://www.cofirank.org, The dimension and the regularization parameter are set as suggested in the paper. For the rest of the parameters, we left them as provided.

[2] http://prea.gatech.edu, We run the code with each of the 48 sets of loss function and parameters given in the main code, and the best result is reported. We could not run this algorithm on the Netflix dataset due to time constraint.

[3] https://bitbucket.org/d_ijk_stra/robirank, We used the part for collaborative ranking from binary relevence score. We left the parameter settings as provide with the implementation.

	MovieLens1m	MovieLens10m	Netflix
Users	6,040	71,567	480,000
Items	3,900	10,681	17,000
Ratings	1,000,209	10,000,054	100,000,000

Table 1. Datasets to be used for simulation

The algorithms are compared in terms of two standard performance measures of ranking, which are NDCG and Precision@K. NDCG@K is the ranking measure for numerical ratings. NDCG@K for user i is defined as

$$\mathrm{NDCG@}K(i) = \frac{\mathrm{DCG@}K(i, \pi_i)}{\mathrm{DCG@}K(i, \pi_i^*)}$$

where

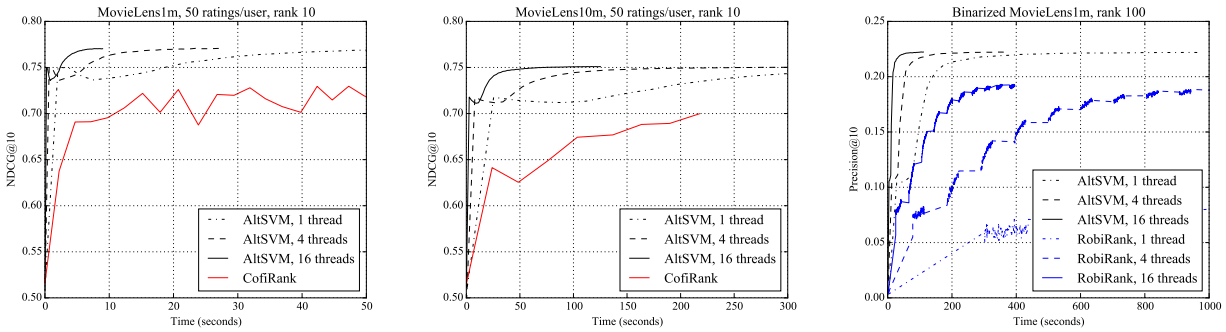
$$\mathrm{DCG@}K(i, \pi_i) = \sum_{k=1}^{K} \frac{2^{M_{i\pi_i(k)}} - 1}{\log_2(k+1)},$$

and $\pi_u(k)$ is the index of the kth ranked item of $\mathcal{T}_i$ in our prediction. M_{ij} is the true rating of item j by user i in the given dataset, and π_u^* is the permutation that maximizes DCG@K. This measure counts only the top K items in our predicted ranking and put more weights on the prediction of highly ranked items. We measured NDCG@10 in our experiments. Precision@K is the ranking measure for binary ratings. Precision@K for user i is defined as

$$\mathrm{Precision@}K(i) = \frac{1}{K} \sum_{j \in \mathcal{P}_K(i)} M_{ij}$$

where M_{ij} is the binary rating on item j by user i given in the dataset. This counts the number of relevant items in the predicted top K recommendation. These two measures are averaged over all of the users.

We first compare our algorithm with numerical rating based algorithms, CofiRank and LCR. We follow the standard setting that are used in the collaborative ranking literature (Weimer et al., 2007; Balakrishnan & Chopra, 2012; Volkovs & Zemel, 2012; Lee et al., 2014). For each user, we subsampled N ratings, used them for training, and took the rest of the ratings for test. The users with less than $N + 10$ ratings were dropped out. Table 2 compares AltSVM with numerical rating based algorithms. While $N = 20$ is too small so that a global ranking provides the best NDCG, our algorithm performs the best with larger N. We also ran our algorithm with subsampled pairwise comparions with the largest numerical gap (AltSVM-sub), which are as many as N for each user (the number of numerical ratings used in the other algorithms). Even with this, we could achieve better NDCG. We can also observe that the statistical performance is better with the hinge loss than with the logistic loss.



(a) (b) (c)

Figure 2. NDCG@10 and Precision@10 over time for different algorithms.

Datasets	N	AltSVM	AltSVM-sub	AltSVM-logistic	Global	CofiRank	LCR
MovieLens1m	20	0.7308	0.6998	0.7125	**0.7500**	0.7333	0.7007
	50	**0.7712**	0.7392	0.7141	0.7501	0.7441	0.7081
	100	**0.7902**	0.7508	0.7446	0.7482	0.7332	0.7151
MovieLens10m	20	0.7059	0.7053	0.7031	**0.7264**	0.7076	0.6977
	50	**0.7508**	0.7212	0.7115	0.7176	0.6977	0.6940
	100	**0.7692**	0.7248	0.7292	0.7101	0.6754	0.6899
Netflix	20	0.7132	0.6822	-	**0.7605**	0.6615	-
	50	**0.7642**	0.7111	-	**0.7640**	0.6527	-
	100	**0.8007**	0.7393	-	0.7656	0.6385	-

Table 2. NDCG@10 on different datasets, for different numbers of observed ratings per user.

	AltSVM			RobiRank
Precision@	$C = 1000$	$C = 2000$	$C = 5000$	
1	0.2165	0.2973	**0.3635**	0.3009
2	0.1965	0.2657	**0.3297**	0.2695
5	0.1572	0.2097	**0.2697**	0.2300
10	0.1265	0.1709	**0.2223**	0.1922
100	0.0526	0.0678	**0.0819**	0.0781

Table 3. Precision@K on the binarized MovieLens1m dataset.

# cores	1	2	4	8	16
Time(seconds)	963.1	691.8	365.1	188.3	111.0
Speedup	1x	1.4x	2.6x	5.1x	8.7x

Table 4. Scalability of AltSVM on the binarized MovieLens1m dataset.

We have also experimented with collaborative ranking on binary ratings. We compare our algorithm against RobiRank (Yun et al., 2014), which is a recently proposed algorithm for collaborative ranking with binary ratings. We ran an experiment on a *binarized* version of the Movielens1m dataset. In this case, the movies rated by a user is assumed to be relevant to the user, and the other items are not. Since it is inefficient to take all possible comparisons which are in average a half million per user, we subsampled C comparisons for each user. Both algorithms are set to estimate rank-100 matrices. Table 3 shows that our algorithm provides better performance than RobiRank.

5.3. Computational speed and Scalability

We now show the computational speed and scalability of our practical algorithm, AltSVM. The experiments were run on a single 16-core machine in the Stampede Cluster at University of Texas.

Figures 2a and 2b show NDCG@10 over time of our algorithms with 1, 4, and 16 threads, compared to CofiRank. Figure 2c shows Precision@10 over time of our algorithm with $C = 5000$. We note that our algorithm converges faster, while the sample size $|\Omega|$ for our algorithm is larger than the number of training ratings that are used in the competing algorithms. Table 4 shows the scalability of AltSVM. We measured the time to achieve 10^{-5} tolerance on the binarized MovieLens1m dataset. As can be seen in the table, we could achieve significant speedup.

6. Conclusion

We considered the collaborative ranking problem where one fits a low-rank matrix to the pairwise comparisons by multiple users. We showed that the convex relaxation of the empirical risk minimization provides good generalization guarantees. For the large-scale practical settings, we also proposed a non-convex algorithm, which alternately solves two SVM problems. Our algorithm was shown to outperform the existing ones and parallelizes well.

References

Ailon, Nir. Active learning ranking from pairwise preferences with almost optimal query complexity. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 810–818, 2011.

Balakrishnan, Suhrid and Chopra, Sumit. Collaborative ranking. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2012.

Bradley, Ralph Allan and Terry, Milton E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pp. 324–345, 1952.

Davenport, Mark A, Plan, Yaniv, Berg, Ewout van den, and Wootters, Mary. 1-bit matrix completion. *Information and Inference*, 3(3):189–223, 2014.

Hajek, Bruce, Oh, Sewoong, and Xu, Jiaming. Minimax-optimal inference from partial rankings. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Herbrich, Ralf, Graepel, Thore, and Obermayer, Klaus. *Large Margin Rank Boundaries for Ordinal Regression*, chapter 7, pp. 115–132. MIT Press, January 2000.

Hsieh, Cho-Jui, Chang, Kai-Wei, Lin, Chih-Jen, Keerthi, S. Sathiya, and Sundararajan, S. A dual coordinate descent method for large-scale linear SVM. In *International Conference on Machine Learning (ICML)*, 2008.

Hsieh, Cho-Jui, Yu, Hsiang-Fu, and Dhillon, Inderjit S. PASSCoDe: Parallel asynchronous stochastic dual coordinate descent. In *International Conference on Machine Learning (ICML)*, 2015.

Hu, Yifan, Koren, Yehuda, and Volinsky, Chris. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM)*, pp. 263–272. IEEE, 2008.

Jamieson, K. G. and Nowak, R. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2011a.

Jamieson, Kevin G. and Nowak, Robert D. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2011b.

Joachims, Thorsten. Optimizing search engines using clickthrough data. In *SIGKDD*, 2002.

Lee, Joonseok, Bengio, Samy, Kim, Seungyeon, Lebanon, Guy, and Singer, Yoram. Local collaborative ranking. In *International World Wide Web Conference (WWW)*, 2014.

Liu, Nathan N, Zhao, Min, and Yang, Qiang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 759–766. ACM, 2009.

Liu, Tie-Yan. *Learning to Rank for Information Retrieval*. Now Publishers Inc., 2009.

Lu, Yu and Negahban, Sahand. Individualized rank aggregation using nuclear norm regularization. *ArXiv e-prints: 1410.0860*, Oct 2014.

Luce, Duncan R. *Individual Choice Behavior*. Wiley, 1959.

Negahban, Sahand, Oh, Sewoong, and Shah, Devavrat. Iterative ranking from pair-wise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Niu, Feng, Recht, Benjamin, Ré, Christopher, and Wright, Stephen. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Rendle, Steffen, Freudenthaler, Christoph, Gantner, Zeno, and Schmidt-Thieme, Lars. 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, Shai and Zhang, Tong. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research (JMLR)*, pp. 567–599, 2013.

Shi, Yue, Karatzoglou, Alexandros, Baltrunas, Linas, Larson, Martha, Oliver, Nuria, and Hanjalic, Alan. Climf: collaborative less-is-more filtering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 3077–3081. AAAI Press, 2013.

Srebro, Nathan, Rennie, Jason, and Jaakkola, Tommi. Maximum margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

Volkovs, Maksims N. and Zemel, Richard S. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Wauthier, Fabian L., Jordan, Michael I., and Jojic, Nebojsa. Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning (ICML)*, 2013.

Weimer, Markus, Karatzoglou, Alexandros, Le, Quoc V., and Smola, Alex. Cofirank: maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Weston, Jason, Want, Chong, Weiss, Ron, and Berenzeig, Adam. Latent collaborative retrieval. In *International Conference on Machine Learning (ICML)*, 2012.

Xu, Jiaming, Wu, Rui, Zhu, Kai, Hajek, Bruce, Srikant, R, and Ying, Lei. Jointly clustering rows and columns of binary matrices: Algorithms and trade-offs. In *ACM Sigmetrics*, 2013.

Yi, Jinfeng, Jin, Rong, Jain, Shaili, and Jain, Anil. Inferring users preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

Yun, Hyokun, Raman, Parameswaran, and Vishwanathan, S. V. N. Ranking via robust binary classification and parallel parameter estimation in large-scale data. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.