# Inferring Users' Preferences from Crowdsourced Pairwise Comparisons: A Matrix Completion Approach

**Jinfeng Yi**[†] **Rong Jin**[†] **Shaili Jain**[*] **Anil K. Jain**[†]
[†]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
{yijinfen, rongjin, jain}@cse.msu.edu
[*]Microsoft, Bellevue, WA 98004 USA
shj@microsoft.com

## Abstract

Inferring user preferences over a set of items is an important problem that has found numerous applications. This work focuses on the scenario where the explicit feature representation of items is unavailable, a setup that is similar to collaborative filtering. In order to learn a user's preferences from his/her response to only a small number of pairwise comparisons, we propose to leverage the pairwise comparisons made by many crowd users, a problem we refer to as **crowdranking**. The proposed crowdranking framework is based on the theory of matrix completion, and we present efficient algorithms for solving the related optimization problem. Our theoretical analysis shows that, on average, only $O(r \log m)$ pairwise queries are needed to accurately recover the ranking list of $m$ items for the target user, where $r$ is the rank of the unknown rating matrix, $r \ll m$. Our empirical study with two real-world benchmark datasets for collaborative filtering and one crowdranking dataset we collected via Amazon Mechanical Turk shows the promising performance of the proposed algorithm compared to the state-of-the-art approaches.

## Introduction

In this paper, we focus on the problem of inferring a user's preference over a set of items, given his response to a *small* number of pairwise comparisons. More specifically, given a collection of $m$ items and a target user $u_t$, our goal is to infer the ranking list of $m$ items for $u_t$ by his preferences over a small number of item pairs. Similar to collaborative filtering (Goldberg et al. 1992), we assume no explicit feature representation is provided for items, which distinguishes our work from the existing studies on learning to rank (Li 2011). We emphasize the challenge of our problem because according to (Jamieson and Nowak 2011), at least $O(m \log m)$ pairwise comparisons are needed to obtain a full ranking list for $m$ items.

We propose to address this challenge by leveraging the pairwise preference information obtained from a set of crowd users via crowdsourcing. More specifically, we assume that besides the target user $u_t$, we have a large number of crowd users and a set of pairwise comparisons made by each crowd user. Our goal is to dramatically reduce the number of pairwise queries needed to infer the ranking list for $u_t$ by effectively exploiting the pairwise preference information provided by crowd users. Similar to most studies on crowdsourcing, our work also divides a major task into many small pieces and asks individual crowd workers to solve them; we refer to our problem as *crowdsourced ranking*, or *crowdranking* for short. We note that the problem of inferring a user's preferences from a collection of pairwise comparisons made by many users has been studied before (Liu, Zhao, and Yang 2009; Rendle et al. 2009). Since pairwise comparisons generally reveal less information about users' interests compared to numerical ratings, these algorithms require a large number of pairwise comparisons, making them impractical for real-world applications. The key contribution of this work is to show, both theoretically and empirically, that by intelligently exploiting the pairwise comparisons made by crowd users, we can accurately infer the list of preferred items by only asking the target user a small number of pairwise comparison questions.

Crowdranking is closely related to collaborative filtering where the goal is to infer the numerical ratings of items for the target user $u_t$ based on the ratings of training (crowd) users. Since numerical ratings of items can always be converted to pairwise comparisons (but not true the other way around), in this regard, we can view crowdranking as a generalization of collaborative filtering. Generally speaking, crowdranking with pairwise comparisons is preferable over traditional collaborative filtering with numerical ratings. This is because pairwise comparisons can be derived from implicit feedback, such as the click-through information in online advertising and web search, when explicit feedback (e.g. numerical ratings) from users is either unavailable or difficult to obtain. Furthermore, according to earlier studies (Liu, Zhao, and Yang 2009; Negahban, Oh, and Shah 2012), often it is more reliable to obtain pairwise preference information of items from individual users than their numerical ratings, due to the subjectiveness when users provide numerical ratings.

In order to effectively explore the pairwise preference information from a collection of crowd users, we assume that despite the diversity in the ranking lists from different users, there exist a small number of underlying intrinsic ranking functions such that all the ranking lists are derived from a linear combination of these intrinsic ranking functions. We refer to this assumption as the *low rank* assumption, which

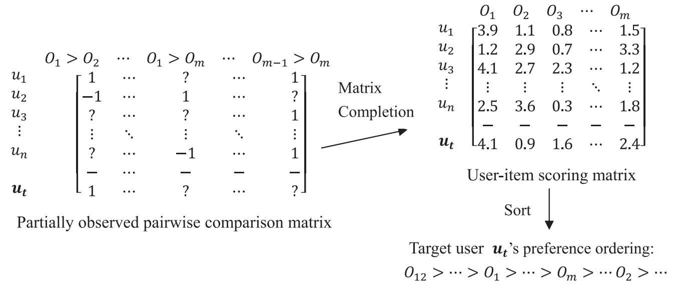

Partially observed pairwise comparison matrix
Matrix Completion
User-item scoring matrix
Sort
Target user $u_t$'s preference ordering:
$o_{12} > \cdots > o_1 > \cdots > o_m > \cdots o_2 > \cdots$

Figure 1: The proposed framework for crowdranking. The target user $u_t$ is augmented with the $n$ crowd users $u_1, \cdots, u_n$ to form a $(n+1) \times C_m^2$ partially observed pairwise comparison matrix, where $m$ is the number of objects to be ranked. The entry of 1 in the $p$-th row of the matrix indicates that user $u_p$ prefers item $o_i$ to $o_j$, $-1$ if $u_p$ prefers item $o_j$ to $o_i$, and ? if the pair is not rated by $u_p$. The $(n+1) \times m$ user-item scoring matrix is completed under the low rank assumption. The preference ordering of the target user $u_t$ is derived from the sorting of his scores over the $m$ items.

was first introduced in recommender systems by (Sarwar et al. 2000). We have developed a matrix completion framework for crowdranking based on this low rank assumption. Figure 1 depicts the proposed framework. We show that, when there are only $r$ intrinsic ranking functions among the crowd ($r$ is also referred to as rank of the rating matrix) , on average, only $O(r \log m)$ pairwise queries are needed to accurately recover the ranking list of $m$ items for the target user, provided the size of crowd is sufficiently large. We verify the effectiveness of the proposed algorithm for crowdranking using two benchmark datasets for collaborative filtering and one crowdsourced dataset that we collected through Amazon Mechanical Turk. Our results show that the proposed algorithm is effective for crowdranking and comparable to the state-of-the-art algorithms for collaborative filtering.

## Related work

In this section, we review the related work on collaborative filtering, crowdsourced ranking and learning to rank, given a set of pairwise comparisons.

*Collaborative filtering* can be divided into two categories: memory-based approaches, and model-based approaches (Su and Khoshgoftaar 2009). Typical examples of memory-based collaborative filtering approaches are item-based and user-based methods. They assume that similar users would give similar ratings, or similar items would also be rated similarly. Several different similarity functions have been exploited to measure the similarity between two users/items based on their ratings, including Cosine similarity (Sarwar et al. 2001), Pearson Correlation (Herlocker, Konstan, and Riedl 2002) and conditional probability-based similarity (Deshpande and Karypis 2004). In contrast to the memory-based approaches, model-based approaches learn a generative model from the observed ratings that can be used to predict the unobserved ratings (Heckerman et al. 2000; Canny 2002; Xue et al. 2005). Many state-of-the-art algorithms for collaborative filtering are based on matrix factorization (Salakhutdinov and Mnih 2008; Rennie and Srebro 2005). The key idea behind them is to find a low rank user-rating matrix that best approximates the observed ratings. Finally, several algorithms were developed to infer lists of items preferred by individual users from a collection of pairwise comparison made by the users (Liu, Zhao, and Yang 2009; Rendle et al. 2009). The key difference between this study and the existing algorithms is that we focus on the scenario when the number of available pairwise comparisons is small (i.e., less than 20 in our experiments).

*Learning to rank* trains a statistical model for ranking tasks (Li 2011). Popular approaches for learning to rank with pairwise comparisons include Active Ranking (Jamieson and Nowak 2011), RankNet (Burges et al. 2005), IR SVM (Cao et al. 2006), and LambdaRank (Burges, Ragno, and Le 2006). Since learning to rank requires a vector representation of the items to be ranked, they can not be directly applied to crowdranking.

*Crowdsourced ranking* focuses on learning a single ranking function from the ranked items. In (Chen et al. 2013), the authors proposed a Bayesian framework to actively select pairwise comparison queries, and effectively combine the pairwise comparisons acquired by crowdsourcing to form a single ranking list. In (Negahban, Oh, and Shah 2012), the authors developed an iterative rank aggregation algorithm to effectively infer ratings for items from pairwise comparisons using Bradley-Terry-Luce model. Since all these studies assume that there exists a common ranking list shared by all the crowd users, they can not be applied to our problem where a different ranking list is learned for an individual user. The authors in (Tamuz et al. 2011) use crowdsourced data to learn a similarity matrix for a given set of objects. While the proposed method uses pairwise queries (i.e. is $a$ better than $b$?) for recommendation, their learning method is based on triplet queries (i.e. is $a$ more similar to $b$ than to $c$?).

## Crowdranking by Matrix Completion

We first present the problem of crowdranking from the viewpoint of matrix completion. Let $u_1, \ldots, u_n$ be the crowd of $n$ workers or users, and $o_1, \ldots, o_m$ be a set of $m$ items to be ranked. We introduce the notation $o_j \succ_{u_i} o_k$ if item $o_j$ is preferred by user $u_i$ over item $o_k$. We encode each pairwise comparison $o_j \succ_{u_i} o_k$ by a triple $(u_i, o_j, o_k)$, and denote by $\Omega = \{(u_i, o_j, o_k) : o_j \succ_{u_i} o_k\}$ the set of triplets that encode all the pairwise comparisons obtained from the crowd of $n$ users. In addition, we have a target user $u_t$ who provides pairwise preferences for a small subset of item pairs, denoted by $\Omega_t = \{(i,j) : o_i \succ_{u_t} o_j\}$. Our goal is to infer the full ranking of $m$ items for the target user $u_t$ by effectively exploiting the pairwise preference information obtained from the crowd.

We will examine two setups for crowdranking. In the first setup, we do not distinguish between the target user and the crowd users. Instead of only inferring the ranking list for the target user, our goal is to infer the ranking lists for *all* the

$(n+1)$ users, including the $n$ crowd users and the target user, $u_t$. We refer to this setting as *transductive crowdranking*. In the second setup, we only aim to infer the ranking list for the target user $u_t$, which we refer to as *inductive crowdranking*. Although a transductive crowdranking method can always be applied to the inductive setting, it is computationally expensive as it needs to first infer the ranking lists for all the users. We address the computational issue by developing an approximate algorithm for inductive crowdranking.

## Significance of Pairwise Comparison in Learning User Preference

In this study, we emphasize the reason to infer user's preference list from pairwise comparisons. Alternatively, we can also infer the user's preference list based on the numerical ratings provided by crowd users, an approach that is followed by most collaborative filtering methods. The fundamental limitation in using numerical ratings arises from the fact that two users sharing the same preference list may give different ratings to the same set of items. This will lead to a small similarity between the two users computed based on ratings and as a result, the incorrect conclusion that they share different preference of items. In other words, due to the subjectiveness in users' ratings, the user-item-rating matrix may not accurately reflect the underlying correlation in preference among different users. Pairwise comparison addresses this limitation by asking the users to provide their preferences between pairs of items which is more reliable in capturing the correlation among users' preferences.

To illustrate the limitation of numerical ratings, we construct a toy example consisting of 50 users and 60 items. We divide the 60 items into three groups, each with twenty items. We assume all the 50 users share the same preference list of items, i.e. items in the first group are more preferred than items in the second group, and items in the second group are more preferred than items in the third group. Let $\mathbf{f} = (f_1, \ldots, f_{60})^\top$ be the canonical rating vector that gives rating 5 to items in the first group, 3 to items in the second group, and 1 to items in the third group. To capture the subjectiveness in user's ratings, we introduce five random vectors $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,60})^\top \in \{-1, +1\}^{60}$, $i = 1, 2, \ldots, 5$, where each entry $v_{i,j}$ has equal chance to be $+1$ and $-1$, with each random vector modeling a different type of rating subjectiveness. The final rating vector for user $u_i$ is given by $\mathbf{f} + \mathbf{v}_k$, where $k$ is an integer randomly sampled from 1 to 5. Since the gap between groups of items in rating vector $\mathbf{f}$ is at least 2, the perturbation vector $\mathbf{v}_i$ does not change the ordering between groups of items. We construct a matrix $F = (\mathbf{f}_1, \ldots, \mathbf{f}_{50})$ that includes the ratings of all 50 users. Figure 2 (a) shows a heat map for the matrix $F$.

Since in this example, all the users share the same preference list of items, we would expect the ratings of different users to be linearly dependent and therefore matrix $F$ to be of rank 1. Figure 2 (b) shows the distribution of singular values for matrix $F$, indicating that the rank of $F$ is 5, significantly larger than 1. On the other hand, we can construct a pairwise comparisons matrix $Z \in \mathbb{R}^{50 \times \frac{60 \times 59}{2}}$ for the 50 users, where each column corresponds to a user pairwise comparison between two items. Figure 2 (b) shows the distribution of singular values for $Z$, indicating that $Z$ is a rank-1 matrix. This simple example illustrates that user-pairwise-comparison matrix is more reliable than user-item-rating matrix in capturing the dependence among the preference lists of different users. Thus, by focusing on the pairwise comparisons of preference, we are able to remove the artifacts caused by the subjectiveness in users' ratings in modeling user's preference.

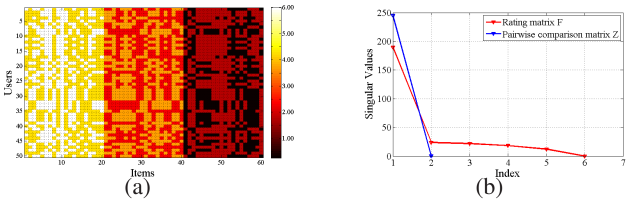


Figure 2: A toy example that illustrates the limitation of inferring user's preference based on numerical ratings of 60 items. (a) Heat map representation for the ratings of 50 users. The brighter the color, the higher the rating. It shows that all the 50 users share the same preference list of items, i.e. items in the first group are more preferred than items in the second group, and items in the second group are more preferred than items in the third group. (b) Plots of singular values for the rating matrix and pairwise comparison matrix. It shows that although all the users share the same preference list of items, the rank of the rating matrix $F$ is significantly larger than 1. As a comparison, the pairwise comparison matrix $Z$, which uses pairwise comparisons instead of numerical ratings, is a rank-1 matrix.

## Transductive Crowdranking

Since we do not distinguish between the crowd users and the target user in transductive crowdranking, we will simply focus on predicting the ranking lists for the crowd users. Let $f_1(\cdot), \ldots, f_n(\cdot)$ be the *unknown* ranking functions adopted by the $n$ crowd users. Let $\mathbf{f}_i = (f_i(o_1), \ldots, f_i(o_m))^\top \in \mathbb{R}^m$ be the vector of rating scores given by the user $u_i$ to $m$ items, and let $F = (\mathbf{f}_1, \ldots, \mathbf{f}_n)^\top \in \mathbb{R}^{n \times m}$ be the rating matrix that includes the rating scores by all the users. Note that each crowd user only rates a small subset of the $m$ items, the matrix $F$ is a partially observed matrix with a majority of its entries to be unobserved. In order to infer the ranking lists for all the crowd users, it is equivalent to inferring the rating matrix $F$. Below, we will first present a matrix completion based framework for recovering the rating matrix $F$, and then present an algorithm that efficiently solves the related optimization problem.

**A Matrix Completion Framework for Transductive Crowdranking** We first discuss how the low rank assumption affects the choice of the rating matrix $F$. Under the low rank assumption, there exist $r$ intrinsic ranking functions $\widehat{f}_1(\cdot), \ldots, \widehat{f}_r(\cdot)$, where $r \ll n$, such that every $f_i(\cdot)$ is a linear combination of $\{\widehat{f}_i(\cdot)\}_{i=1}^r$. More specifically, let $\widehat{\mathbf{f}}_i = (\widehat{f}_i(o_1), \ldots, \widehat{f}_i(o_m))^\top \in \mathbb{R}^m$ be the rating scores given by the intrinsic ranking function $\widehat{f}_i$. Then, there exists a vector of coefficients $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,r})^\top \in \mathbb{R}^r$

for each user $u_i$ such that $f_i(\cdot) = \sum_{j=1}^{r} a_{i,j}\widehat{f}_j$. As a result, the low rank assumption implies that there exists a matrix $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{R}^{r\times n}$ such that $F = A^\top(\widehat{\mathbf{f}}_1, \ldots, \widehat{\mathbf{f}}_r)$, which is equivalent to assuming that the rank of $F$ is at most $r$.

To measure the consistency between the rating matrix $F$ and the observed pairwise comparisons from crowd users, we introduce a convex loss function $\ell(z)$ that is monotonically decreasing in $z$. Given a pairwise comparison $o_j \succ_{u_i} o_k$, the inconsistency of $F$ is measured by $\ell(F_{i,j} - F_{i,k})$: the larger the loss function, the larger the inconsistency between $F$ and the pairwise comparisons. Example loss functions include hinge loss $\ell(z) = \max(0, 1-z)$ and logistic loss $\ell(z) = \log(1 + e^{-z})$. Using the loss function, the inconsistency between $F$ and all the observed pairwise comparisons encoded by the triplets in $\Omega$ is then given by

$$\sum_{(i,j,k)\in\Omega} \ell(F_{i,j} - F_{i,k})$$

Our goal is to search for a low rank matrix $F \in \mathbb{R}^{n\times m}$ that is consistent with most of the observed pairwise comparisons in $\Omega$, leading to the following optimization problem

$$\min_{F\in\mathbb{R}^{n\times m}} \quad \lambda\, rank(F) + \sum_{(i,j,k)\in\Omega} \ell(F_{i,j} - F_{i,k}) \quad (1)$$

where $\lambda > 0$ is introduced to balance the tradeoff between minimizing the rank of $F$ and reducing the inconsistency with respect to the observed pairwise comparisons in $\Omega$. Since $rank(F)$ is a non-convex function, we approximate $rank(F)$ by the trace norm $|F|_{tr}$, the convex envelope of $rank(F)$ (Candès and Recht 2009), and convert (1) into a convex optimization problem

$$\min_{F\in\mathbb{R}^{n\times m}} \quad \mathcal{L}(F) = \lambda|F|_{tr} + \sum_{(i,j,k)\in\Omega} \ell(F_{i,j} - F_{i,k}) (2)$$

Let $\widehat{F}$ be the solution obtained from (2). The following theorem, known through our personal communication with Robert Nowak[1] provides a theoretical guarantee for the solution $\widehat{F}$.

**Theorem 1.** *Assume $\ell(z)$ in (2) is a hinge loss function. Suppose the true rating matrix $F^*$ is at most rank $r$ and $\Omega$ is the set of pairwise comparisons chosen uniformly at random. Then with a probability $1-\delta$, the normalized Kendall tau distance between rankings estimated from $\widehat{F}$, the optimal solution for (2), and the true rating matrix $F^*$is upper bounded by*

$$\frac{r(m+n)(\log(m) + \log(1/\delta))}{|\Omega|},$$

where the normalized Kendall tau distance is within the range $[0, 1]$.

**Remark** Given two ranking lists $\tau_1$ and $\tau_2$, the normalized Kendall tau distance between $\tau_1$ and $\tau_2$ is given by

$$K(\tau_1, \tau_2) = \frac{2}{m(m-1)} \sum_{j=1}^{m} \sum_{i>j}^{m} \bar{K}_{i,j}(\tau_1, \tau_2)$$

where $\bar{K}_{i,j}(\tau_1, \tau_2) = 0$ if $i$ and $j$ are in the same order in $\tau_1$ and $\tau_2$ and 1, otherwise. As a result, the normalized Kendall tau distance can also be interpreted as the probability of any two randomly sampled items to be in different order in the ranking lists $\tau_1$ and $\tau_2$. To be more precise, let $\widehat{\tau}_i$ and $\tau_i^*$ be the ranking lists for user $u_i$ derived from rating matrices $\widehat{F}$ and $F^*$, respectively. If we set the number of observed pairwise comparisons in $\Omega$ to be $|\Omega| = 30r(m+n)\log m = \Omega(r(m+n)\log m)$, then, with a probability $1 - m^{-2}$, for any user $u_i$ and for 90% of randomly sampled item pairs $o_j$ and $o_k$, they will be in the same order in the ranking lists $\widehat{\tau}_i$ and $\tau_i^*$. This result implies, in order to obtain accurate ranking lists for all the crowd users, on average, we only need to query each user with $O(r\log m)$ pairwise comparisons, significantly smaller than $O(m\log m)$ pairwise comparisons that are required when the training set of pairwise comparisons from crowd users is not available.

**An Efficient Algorithm for Transductive Crowdranking** Following Theorem 1, we set the loss function in (2) to be a hinge loss, i.e. $\ell(z) = \max(0, 1-z)$. One popular approach for solving the optimization problem in (2) is gradient descent. At each iteration $t$, given the current solution $F_t$ for (2), we first compute a subgradient of the objective function $\mathcal{L}(F)$ at $F = F_t$, denoted by $\partial\mathcal{L}(F_t)$, and then update the solution by

$$F_{t+1} \quad = \quad F_t - \eta_t\partial\mathcal{L}(F_t) \quad (3)$$

where $\eta_t > 0$ is the step size at the $t$-th iteration. Let $F_t = U_t\Sigma_tV_t^\top$ be the singular value decomposition of $F_t$. Since $U_tV_t^\top$ is a subgradient of $|F|_{tr}$ at $F = F_t$, we have

$$\partial\mathcal{L}(F_T) = \left(\sum_{(i,j,k)\in\Omega} \ell'(F_{i,j} - F_{i,k})\mathbf{e}_i^n(\mathbf{e}_j^m - \mathbf{e}_k^m)^\top\right) - U_tV_t^\top$$

where $\mathbf{e}_i^n$ is a vector of $n$ dimensions with all the elements being zero except that its $i$th entry is 1.

The main computational challenge in implementing the gradient descent approach for optimizing (2) arises from the high cost in computing the singular value decomposition of $F_t$, particularly when both $n$ and $m$ are large. We address this computational challenge by exploiting the stochastic subgradient descent algorithm that was recently proposed for matrix completion (Avron et al. 2012). The key idea is to approximate the gradient $\partial\mathcal{L}(F_t)$ by a low rank matrix, and update the solution $F_t$ using the low rank approximation of the gradient. To this end, we introduce a low rank probing matrix $Y \in \mathbb{R}^{m\times k}$, where $k \ll m$. We assume $\mathrm{E}[YY^\top] = I_m$, i.e. $YY^\top$ forms an unbiased estimate of the identity matrix $I_m$. Using the probing matrix $Y$, we approximate the gradient $\partial\mathcal{L}(F_t)$ by

$$\widetilde{\partial}\mathcal{L}(F_t) = \partial\mathcal{L}(F_t)YY^\top$$

[1] http://nowak.ece.wisc.edu/

There are two important properties of the approximated gradient $\widetilde{\partial}\mathcal{L}(F_t)$ that can be used to facilitate our computation. First, $\widetilde{\partial}\mathcal{L}(F_t)$ is an unbiased estimate of $\partial\mathcal{L}(F_t)$, i.e. $\mathrm{E}[\widetilde{\partial}\mathcal{L}(F_t)] = \nabla\mathcal{L}(F_t)$. Second, $\widetilde{\partial}\mathcal{L}(F_t)$ is a low rank matrix, i.e. $rank(\widetilde{\partial}\mathcal{L}(F_t)) \leq k$. The unbiased property of $\widetilde{\partial}\mathcal{L}(F_t)$ ensures that the solution generated by the proposed algorithm will converge to the true optimal solution when the approximate gradient $\widetilde{\partial}\mathcal{L}(F_t)$ is used for updating. The low rank property of $\widetilde{\partial}\mathcal{L}(F_t)$ allows for more efficient singular value decomposition.

Given the approximate gradient $\widetilde{\partial}\mathcal{L}(F_t)$, we update the solution by

$$F_{t+1}= \Pi_r(F'_{t+1}), \tag{4}$$

$$F'_{t+1}=F_t - \eta\widetilde{\partial}\mathcal{L}(F_t) = [U_t\Sigma_t, -\eta\nabla\mathcal{L}(F_t)Y]\begin{bmatrix}V_t^\top\\ Y^\top\end{bmatrix} \tag{5}$$

where operator $\Pi_r(Z) = U_r(Z)\Sigma(Z)V_r(Z)^\top$ computes the best rank-$r$ approximation of $Z$. If the rating matrix $F_t$ is of rank $r$, the rank of matrix $F'_{t+1}$ is upper bounded by $r + k$, and as a result, the singular vector decomposition of $F'_{t+1}$ can be computed more efficiently by explicitly exploring the low rank structure of $F'_{t+1}$ (Avron et al. 2012). In particular, the cost of computing the first $r$ eigenvalues and eigenvectors of $F'_{t+1}$ is $O(r(r + k)\max(n, m))$. Compared to directly computing the first $r$ singular values and singular vectors of $F'_{t+1}$ whose cost is $O(rnm)$, this is significantly more efficient. In our implementation, we follow (Avron et al. 2012) who randomly sample $k$ vectors from the following set

$$\mathbf{e}_1^m, \mathbf{e}_2^m, \ldots, \mathbf{e}_m^m,$$

and form the probing matrix $Y = \sqrt{\frac{m}{k}}\left(\mathbf{e}_{i_1}^m, \ldots, \mathbf{e}_{i_k}^m\right)$, where $\mathbf{e}_{i_j}^m, j = 1, \ldots, k$ are randomly sampled vectors. It is easy to verify that the probing matrix generated in this way will guarantee to be an unbiased estimator of the identity matrix.

## Inductive Crowdranking

The main limitation of transductive crowdranking arises from its high computational cost, i.e. in order to predict the ranking list of items for the target user $u_t$, we have to make the predictions for all the $n$ crowd users as well. To improve the computational efficiency, we divide the algorithm for inductive crowdranking into two phases: *learning phase* and *prediction phase*. In the learning phase, we estimate the rating matrix $\widehat{F}$ for the $n$ crowd users based on the observed pairwise comparisons in $\Omega$; in the prediction phase, we estimate the rating vector $\widetilde{\mathbf{f}} \in \mathbb{R}^m$ for the target user $u_t$ based on his pairwise comparisons in $\Omega_t$ and the estimated rating matrix $\widehat{F}$. Since the learning phase overlaps with the transductive crowdranking, we will focus our discussion on the prediction phase.

By fixing the rating matrix $\widehat{F}$ for the crowd users, the rating matrix for both the crowd users and the target user is given by $[\widehat{F}^\top, \widetilde{\mathbf{f}}]^\top$. By replacing $F$ in (2) with $[\widehat{F}^\top, \widetilde{\mathbf{f}}]^\top$, we have the following optimization problem for $\widetilde{\mathbf{f}}$

$$\widetilde{\mathbf{f}} = \arg\min_{\mathbf{f}\in\mathbb{R}^m} \lambda \left|\left| \begin{bmatrix}\widehat{F}\\ \mathbf{f}^\top\end{bmatrix} \right|\right|_{tr} + \sum_{(i,j)\in\Omega_t} \ell(f_i - f_j). \tag{6}$$

Directly solving the optimization problem in (6) is computationally expensive because it requires evaluating the trace norm for a large matrix $[\widehat{F}^\top, \mathbf{f}]^\top$. We address this challenge by exploiting the following lemma that gives an alternative expression for trace norm of a matrix.

**Lemma 1.** *(Rennie and Srebro 2005) For any matrix $X \in \mathbb{R}^{n\times m}$, we have*

$$|X|_{tr} = \min_{UV^\top=X} \frac{1}{2}|U|_F^2 + \frac{1}{2}|V|_F^2$$

*where $|X|_F$ measures the Frobenius norm of matrix $X$.*

Using Lemma 1, we obtain the following upper bound for $|[\widehat{F}^\top, \widetilde{\mathbf{f}}]^\top|_{tr}$.

**Lemma 2.** *Let $\mathbf{v}_1, \ldots, \mathbf{v}_r$ be the right singular vectors of $\widehat{F}$ and let $\sigma_1^2, \ldots, \sigma_r^2$ be the corresponding singular values. Then, for any vector $\mathbf{f}$ lying in the subspace spanned by the row vectors in $\widehat{F}$, we have*

$$\left|\left| \begin{bmatrix}\widehat{F}\\ \mathbf{f}^\top\end{bmatrix} \right|\right|_{tr} \leq |\widehat{F}|_{tr} + \frac{1}{2}\sum_{i=1}^r \frac{|\mathbf{f}^\top\mathbf{v}_i|^2}{\sigma_i^2}$$

The proof of Lemma 2 can be found in the appendix. By replacing the trace norm in (6) with the upper bound given in Lemma 2, we have the following approximate optimization problem for learning the rating vector $\widetilde{\mathbf{f}}$ for the target user

$$\widetilde{\mathbf{f}} = \arg\min_{\mathbf{f}\in\mathbb{R}^m} \frac{\lambda}{2}\sum_{i=1}^r \frac{|\mathbf{f}^\top\mathbf{v}_i|^2}{\sigma_i^2} + \sum_{(i,j)\in\Omega_t} \ell(f_i - f_j) \tag{7}$$

**Remark** Compared to (6), the optimization problem in (7) is significantly simplified because it does not involve evaluating the trace norm of a large matrix. In addition, the first term in the objective function in (7), i.e. $\frac{\lambda}{2}\sum_{i=1}^r \frac{|\mathbf{f}^\top\mathbf{v}_i|^2}{\sigma_i^2}$, essentially plays the role of regularizer: it restricts the solution $\widetilde{\mathbf{f}}$ to the subspace spanned by the singular vectors of $\widehat{F}$; by weighting $\mathbf{f}^\top\mathbf{v}_i$ by $\sigma_i^{-2}$, it essentially favors the solution $\widetilde{\mathbf{f}}$ that aligns with the eigenvectors of the largest $r$ eigenvalues.

We briefly discuss how to efficiently solve the optimization problem in (7). In order to take advantage of the existing algorithms for solving the optimization problem in (7), we introduce a variable $\mathbf{a} \in \mathbb{R}^r$, and write $\mathbf{f}$ as

$$\mathbf{f} = \sum_{i=1}^r a_i\sigma_i\mathbf{v}_i$$

We also introduce a vector representation $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,r})^\top$ for each item $o_i$ as $x_{i,j} = \sigma_j v_{j,i}$. It is easy to verify that $f_i = \mathbf{x}_i^\top\mathbf{a}$, based on which eq. (7) is changed to the following optimization problem

$$\mathbf{a} = \arg\min_{\mathbf{a}\in\mathbb{R}^r} \frac{\lambda}{2}|\mathbf{a}|^2 + \sum_{(i,j)\in\Omega_t} \ell([\mathbf{x}_i - \mathbf{x}_j]^\top\mathbf{a}) \tag{8}$$

Since (8) is a standard optimization problem found in learning to rank, it can be efficiently solved by using the existing tools such as LIBLINEAR[2].

## Experiments

In this section, we verify the effectiveness of the proposed algorithms for crowdranking by conducting two sets of experiments. In the first set of experiments, we apply the proposed algorithms to collaborative filtering problems where user ratings for individual items are provided. Instead of directly modeling user ratings like most collaborative filtering methods, we first convert user ratings into a set of pairwise comparisons, and then apply the proposed algorithms to infer user's preference of items based on the pairwise comparisons. We compare the performance of the proposed algorithms to the state-of-the-art algorithms for collaborative filtering. In the second experiment, we collect crowdsourced data on pairwise comparisons from many users where the items of interest are ladies hand-bags. We evaluate the performance of the proposed algorithms by comparing the ranking lists inferred by the proposed algorithms to the ranking lists provided by the target users. Throughout this section, we will refer to the proposed algorithm for transductive crowdranking and inductive crowdranking as **T-CR** and **I-CR**, respectively, for brevity.

### Experiment (I): Application to Collaborative Filtering

Two real-world collaborative filtering data sets are used in our experiments:

- *MovieLens 1M Data Set*[3] is comprised of $3,952$ movies rated by $6,040$ users. Each movie is rated on a scale from $1$ to $5$, with $5$ indicating the best movie and $1$ indicating the worst movie. There are a total of one million ratings in this dataset.
- *Jester Data Set*[4] is a subset of a larger Jester database (Goldberg et al. 2001). It is comprised of $100$ jokes rated by $10,000$ users. Each joke is rated from $-10$ to $10$, with $10$ for the best joke and $-10$ for the worst joke. There is a total of $56,350$ ratings in this dataset.

Following the standard evaluation protocol of collaborative filtering, for both the datasets, we randomly sample $70\%$ of users as the crowd users (training set), and use the remaining $30\%$ as the target users (test set). Since the proposed algorithm is designed for pairwise comparisons, we convert the rating information into a set of pairwise comparisons. More specifically, we create a triplet $o_j \succ_{u_i} o_k$ if item $o_j$ is rated higher by user $u_i$ than item $o_k$. We note that some information is lost through this conversion. For instance, the derived pairwise comparisons do not reflect the difference in ratings between two items, which could be potentially valuable in inferring user's preference list. In addition, no pairwise comparison data is generated if two items are given the same rating. Despite this information loss in converting ratings to pairwise comparisons, our empirical study shows that the proposed algorithms yield comparable performance as the state-of-the art algorithms for collaborative filtering that explicitly model the rating information.

To simulate the process of querying a target user $u_t$, we randomly select $5$, $10$, $15$, and $20$ pairwise comparisons made by $u_t$, and infer its ranking list of items based on the limited pairwise comparisons and the pairwise comparisons from the crowd users. We note that we focus on the realistic scenario when only a small number of pairwise comparison questions can be asked to infer the user's preference, a problem that is often referred to as "cold start" in collaborative filtering (Schein et al. 2002).

We compare the proposed algorithms (T-CR and I-CR) to the following state-of-the-art baseline algorithms:

- **M$^3$F**: maximum-margin matrix factorization algorithm (Rennie and Srebro 2005),
- **PMF**: probabilistic matrix factorization algorithm (Salakhutdinov and Mnih 2008), and
- **BPR**: Bayesian personalized ranking (Rendle et al. 2009).

Among them, M$^3$F and PMF are state-of-the-art collaborative filtering algorithms which directly apply users' ratings to learn their preferences. Instead of using pairwise comparisons, both of them are applied directly to users' ratings to learn their preferences. BPR, similar to the proposed algorithm, learns the users' preferred items from a collection of pairwise comparison made by many crowd users. For a fair comparison, for each target user, we feed the baseline collaborative filtering algorithms with ratings for $5$, $10$, $15$ and $20$ randomly selected items, and feed the BPR with the preferred items in the same numbers of randomly selected pairwise comparisons.

Since the proposed algorithms (T-CR and I-CR) are only able to predict ranking lists for users, we can not evaluate the performance of the proposed algorithms using the standard evaluation metrics for collaborative filtering (e.g. Mean Absolute Error (MAE)) that require comparing the true numerical ratings with the inferred ones. We thus adopt the normalized discounted cumulative gain (NDCG for short) (Järvelin and Kekäläinen 2002) as our evaluation metric. It evaluates the prediction performance based on the first $p$ items ranked by the prediction model, where $p$ is set to be $5$ and $10$ in our study. All the experiments are performed on a PC with Intel Xeon $2.40$ GHz processor and $64.0$ GB of main memory. Each experiment is repeated five times, and the performance averaged over the five trials is reported in Table 1.

We first observe that for both the testbeds, the proposed algorithms (T-CR and I-CR) yield very similar performance, which validates that the approximation made by the I-CR algorithm is accurate. Compared to the baseline collaborative filtering algorithms, we observe that overall, the proposed algorithms for crowdranking yield slightly better performance measured by NDCG. We want to point out that in the evaluation of recommender systems, a small numerical improvement is usually expected, as it is clearly demonstrated by the Netflix Prize (Bennett and Lanning 2007). The results are still impressive since we can use more easily generated pairwise comparisons to achieve slightly bet-

[2] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

[3] http://www.grouplens.org/node/73

[4] http://eigentaste.berkeley.edu/dataset/

Table 1: Performance comparison of the proposed CrowdRanking algorithms (T-CR and I-CR) and the baseline collaborative filtering algorithms (M$^3$F, PMF, BPR) on MovieLens 1M and Jester datasets.

| Datasets | #pairs /ratings | | T-CR | I-CR | M$^3$F | PMF | BPR |
|---|---|---|---|---|---|---|---|
| MovieLens | 5 | NDCG@5 | 0.72 | **0.73** | 0.71 | 0.70 | 0.68 |
| | | NDCG@10 | **0.73** | **0.73** | 0.71 | 0.70 | 0.68 |
| | 10 | NDCG@5 | 0.73 | **0.75** | 0.73 | 0.72 | 0.69 |
| | | NDCG@10 | 0.73 | **0.75** | 0.73 | 0.71 | 0.69 |
| | 15 | NDCG@5 | **0.76** | **0.76** | 0.75 | 0.75 | 0.72 |
| | | NDCG@10 | **0.76** | 0.75 | 0.75 | 0.75 | 0.73 |
| | 20 | NDCG@5 | 0.76 | **0.77** | **0.77** | 0.75 | 0.74 |
| | | NDCG@10 | 0.76 | **0.77** | **0.77** | 0.76 | 0.75 |
| Jester | 5 | NDCG@5 | **0.37** | **0.37** | 0.36 | 0.34 | 0.33 |
| | | NDCG@10 | **0.40** | **0.40** | 0.39 | 0.38 | 0.37 |
| | 10 | NDCG@5 | **0.40** | **0.40** | 0.39 | 0.36 | 0.35 |
| | | NDCG@10 | 0.42 | **0.43** | 0.42 | 0.40 | 0.38 |
| | 15 | NDCG@5 | 0.40 | **0.41** | **0.41** | 0.39 | 0.38 |
| | | NDCG@10 | **0.44** | **0.44** | 0.43 | 0.42 | 0.41 |
| | 20 | NDCG@5 | 0.42 | 0.43 | **0.44** | 0.42 | 0.41 |
| | | NDCG@10 | 0.46 | **0.47** | 0.46 | 0.44 | 0.44 |

ter performance than using numerical ratings. Besides, the improvement is statistically significant when the number of observed ratings/pairwise-comparisons for each target user is restricted to 10 or less. This may appear to be counter intuitive as a significant amount of information is believed to be lost through the process of converting user's ratings into pairwise comparisons. However, in our opinion, the success of the proposed algorithms can be attributed to the fact that pairwise comparisons are more reliable in reflecting a user's preference of items than his numerical ratings. Thus, despite the perceived information loss, modeling pairwise comparisons can be more effective for recommender systems particularly when the number of ratings given by the target user is small. This makes the proposed algorithms especially useful for the "cold start" problem in recommender systems (Schein et al. 2002), when making recommendations for new users with limited information.

We also evaluate the computational efficiency of the proposed crowdranking algorithms as well as baseline collaborative filtering algorithms. Table 2 shows that although the proposed transductive Crowdranking algorithm (T-CR) is not as efficient as the baseline algorithms, the proposed inductive Crowdranking algorithm (I-CR) is significantly more efficient than all the baseline approaches. In particular, I-CR is able to predict ranking lists of $1,800$ users' for $4,000$ movies in $5.1$ seconds and also predict ranking lists of $3,000$ users' for $100$ jokes in less than $1$ second.

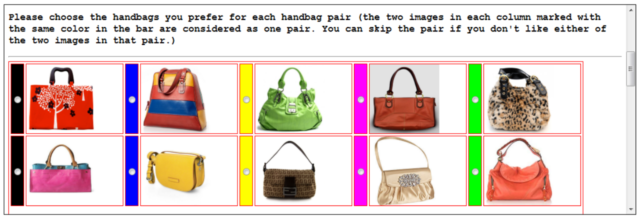
Table 2: CPU time (s) for learning the preferences for all the target users. All the algorithms were run on an Intel Xeon $2.40$ GHz processor with $64.0$ GB of main memory.

| Datasets | #pairs /ratings | T-CR | I-CR | M$^3$F | PMF | BPR |
|---|---|---|---|---|---|---|
| MovieLens | 5 | 3552 | **5.1** | 1180 | 101 | 10 |
| | 10 | 4395 | **5.1** | 1268 | 113 | 11 |
| | 15 | 4446 | **5.1** | 1303 | 115 | 11 |
| | 20 | 4549 | **5.1** | 1588 | 117 | 12 |
| Jester | 5 | 422 | **0.6** | 262 | 25 | 5.2 |
| | 10 | 460 | **0.6** | 231 | 28 | 5.4 |
| | 15 | 466 | **0.6** | 271 | 28 | 5.5 |
| | 20 | 471 | **0.7** | 297 | 29 | 5.7 |

## Experiment (II): A Crowdsourced Study for Crowdranking

In this experiment, we evaluate the proposed algorithms for crowdranking by conducting a crowdsourced study in the domain of ladies handbags. We downloaded a total of 500 images of ladies handbags from the Internet. Figure 3 shows some of the images from our handbag collection. Our goal is to infer a user's preferences of ladies handbags based on a limited number of pairwise comparisons she provides.

Figure 3: Example images from our collection of 500 ladies handbags



Please choose the handbags you prefer for each handbag pair (the two images in each column marked with the same color in the bar are considered as one pair. You can skip the pair if you don't like either of the two images in that pair.)

Figure 4: The user interface we designed in the Amazon Mechanical Turk to collect pairwise comparison data from crowd users

In order to collect the pairwise comparisons for crowd users, we employed 200 workers through the Amazon Mechanical Turk. For each worker, we randomly generated 100 pairs of handbags, and asked the worker to indicate which handbag she prefers in each pair. The total number of Human Intelligence Tasks (HIT) is 200 and we require that each worker can only work on one HIT, for the reason that we want to collect enough diverse ranking lists. Each HIT consisted of 100 image pairs and we paid the workers $\$1.60$ per HIT. Figure 4 shows the user interface that we developed in the Amazon Mechanical Turk to collect pairwise comparison data from 200 workers. As indicated in several studies (Zhou et al. 2012; Yi et al. 2012a), spammers could pose a serious problem to the integrity of crowdsourced data. In order to filter out the spammers or malicious workers, we introduced eight duplicate image pairs in each HIT, and determined whether a worker is a spammer based on the consistency of the pairwise comparison responses for the eight duplicate pairs. We note that the term "spammer" is slightly abused here by including any worker that is in-consistent in providing her answers. We observed that only 76 of the 200 workers provided the same preference decision for more than 75% of the eight duplicate pairs. Although the percentage of spammers here may appear to be high, it is consistent with the findings in (Yi et al. 2012b; Zhou et al. 2012).

To collect the preference information for target users, 10 female students in our lab were requested to serve as the target users. The reason we chose female students as target users is that they are more likely to have stronger preferences on ladies handbags than males. We collected the pairwise comparisons of the 10 target users by asking each of them to make pairwise comparisons for 100 pairs of handbags that are randomly sampled from our collection. We also collected the preference information of the same 10 target users by asking each of them to indicate the top 25 handbags that they like the most. This allows us to compare the ranking list inferred by the proposed algorithms with the top 25 preferred handbags indicated by each target user. We did not choose the crowd workers as the target user because this task requires users going through the entire list of handbags, which is time consuming and not suitable for crowd workers.

In this study, for each target user $u_t$, we first randomly sample 10, 20 and 30 pairwise comparisons provided by $u_t$. Then we predict her ranking list of handbags based on the limited pairwise comparisons provided by her and the pairwise comparisons from the crowd users. To evaluate the quality of the inferred ranking list, for each target user $u_t$, we measure the commonality between the first 25 handbags appearing in the ranking list and the 25 handbags preferred by $u_t$. The experimental results show that, given 10, 20 and 30 randomly sampled pairwise comparisons provided by $u_t$, the average number of common handbags in the two ranking lists (inferred and preferred) for the proposed transductive crowdranking algorithm are 5.3, 7.1, and 10.1, respectively. As a comparison, the baseline BPR method can only achieve 4.9, 6.7, and 9.8 common handbags with 10, 20 and 30 randomly sampled pairwise comparisons. It is not surprising to observe that the performance of the proposed crowdranking algorithm improves, as measured by the number of common handbags in the inferred and preferred lists, with increasing number of pairwise comparisons in the query. Compared to the BPR algorithm, the proposed algorithm is significantly more effective when the number of pairwise comparisons is small (i.e. 10 or 20), and, as expected, the difference between two algorithms dimmish when the number of pairwise comparisons provided by the target users reaches 30. Again, this is not surprising because the proposed algorithm is specifically designed for a small number of pairwise comparisons. Figure 5 shows the first 25 handbags selected by our 10 female target users, one row per user. A handbag image is highlighted by a red box if it also appears in the first 25 handbags inferred by the proposed algorithm.

## Conclusions

In this paper, we have introduced a crowdranking problem where the goal is to infer the ranking list of $m$ items for a target user $u_t$ by effectively exploiting the pairwise comparisons made by $n$ crowd users. We present a crowdranking framework based on the theory of matrix completion. We examine two scenarios for crowd ranking: (i) transductive crowdranking, where the goal is to infer the ranking lists for both the crowd users and the target users, and (ii) inductive crowdranking, where we only need to infer the ranking list for a given target user. We present efficient algorithms, based on stochastic subgradient descent, for solving the related optimization problems. Our empirical studies with two benchmark datasets for collaborative filtering and one real-world dataset collected by us for crowdranking show that the proposed crowdranking algorithm can reliably predict target users' preferences based on their responses to only a small number of pairwise comparisons of the $m$ items.

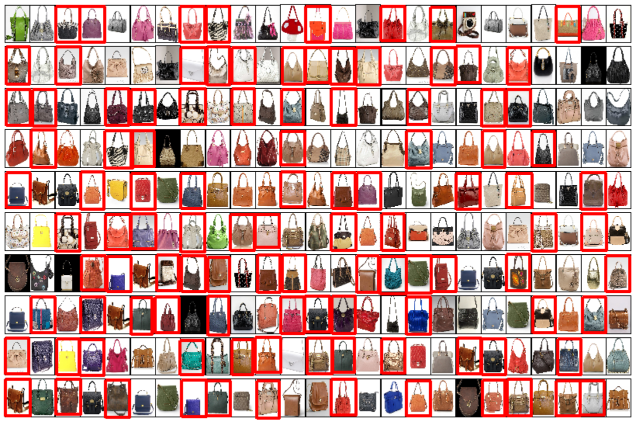


Figure 5: Each row contains the top 25 handbag images selected by one target user. The images highlighted by red boxes are those that also appear in the first 25 handbag predicted by the proposed algorithm. There are a total of 10 target users, one per row.

.

## Acknowledgements

This research was supported by ONR grant no. N00014-11-1-0100, N00014-12-1-0522 and N00014-12-10431.

## Appendix

In this appendix, we prove the Lemma 2. Let $\widehat{F} = U\Sigma V^\top$ be the singular value decomposition of $\widehat{F}$. Define $D = \text{diag}(\sigma_1, \ldots, \sigma_r)$. According to Lemma 1, for $A \in \mathbb{R}^{n\times k}$ and $B \in \mathbb{R}^{m\times k}$, where $k \geq r$ can be any integer, such that

$$AB^\top = \left\| \left[ \begin{array}{c} \widehat{F} \\ \mathbf{f}^\top \end{array} \right] \right\|_{tr}, \qquad (9)$$

we have

$$\left\| \left[ \begin{array}{c} \widehat{F} \\ \mathbf{f}^\top \end{array} \right] \right\|_{tr} \geq \frac{1}{2}|A|_F^2 + \frac{1}{2}|B|_F^2$$

If we choose $B = VD$, due to the condition in (9), $A$ is given by

$$A = \left[ \begin{array}{c} UD \\ \mathbf{f}^\top VD^{-1}] \end{array} \right]$$

We complete the proof by using the fact that

$$\frac{1}{2}|VD|_F^2+\frac{1}{2}|UD|_F^2+\frac{1}{2}|\mathbf{f}^\top VD^{-1}|^2 = |\widehat{F}|_{tr}+\frac{1}{2}\sum_{i=1}^{r}\frac{|\mathbf{f}^\top \mathbf{v}_i|^2}{\sigma_i^2}.$$

## References

Avron, H.; Kale, S.; Kasiviswanathan, S. P.; and Sindhwani, V. 2012. Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *ICML*.

Bennett, J., and Lanning, S. 2007. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD*.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, 89–96.

Burges, C. J. C.; Ragno, R.; and Le, Q. V. 2006. Learning to rank with nonsmooth cost functions. In *NIPS*, 193–200.

Candès, E. J., and Recht, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics* 9(6):717–772.

Canny, J. F. 2002. Collaborative filtering with privacy via factor analysis. In *SIGIR*, 238–245.

Cao, Y.; Xu, J.; Liu, T.-Y.; Li, H.; Huang, Y.; and Hon, H.-W. 2006. Adapting ranking svm to document retrieval. In *SIGIR*, 186–193.

Chen, X.; Bennett, P.; Collins-Thompson, K.; and Horvitz, E. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM*.

Deshpande, M., and Karypis, G. 2004. Item-based top-*n* recommendation algorithms. *ACM Trans. Inf. Syst.* 22(1):143–177.

Goldberg, D.; Nichols, D.; Oki, B.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35(12):61–70.

Goldberg, K. Y.; Roeder, T.; Gupta, D.; and Perkins, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.* 4(2):133–151.

Heckerman, D.; Chickering, D. M.; Meek, C.; Rounthwaite, R.; and Kadie, C. M. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR* 1:49–75.

Herlocker, J. L.; Konstan, J. A.; and Riedl, J. 2002. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.* 5(4):287–310.

Jamieson, K. G., and Nowak, R. D. 2011. Active ranking using pairwise comparisons. In *NIPS*, 2240–2248.

Järvelin, K., and Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.* 20(4):422–446.

Li, H. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers.

Liu, N. N.; Zhao, M.; and Yang, Q. 2009. Probabilistic latent preference analysis for collaborative filtering. In *CIKM*, 759–766.

Negahban, S.; Oh, S.; and Shah, D. 2012. Iterative ranking from pair-wise comparisons. In *NIPS*. 2483–2491.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 452–461.

Rennie, J. D. M., and Srebro, N. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 713–719.

Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, 880–887.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. T. 2000. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*, 285–295.

Schein, A. I.; Popescul, A.; Ungar, L. H.; and Pennock, D. M. 2002. Methods and metrics for cold-start recommendations. In *SIGIR*, 253–260.

Su, X., and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence* 2009:1–20.

Tamuz, O.; Liu, C.; Belongie, S.; Shamir, O.; and Kalai, A. 2011. Adaptively learning the crowd kernel. In *ICML*, 673–680.

Xue, G.; Lin, C.; Yang, Q.; Xi, W.; Zeng, H.; Yu, Y.; and Chen, Z. 2005. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*, 114–121.

Yi, J.; Jin, R.; Jain, A.; Jain, S.; and Yang, T. 2012a. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *NIPS*, 1781–1789.

Yi, J.; Jin, R.; Jain, A. K.; and Jain, S. 2012b. Crowdclustering with sparse pairwise labels: A matrix completion approach. In *AAAI Workshop on Human Computation*.

Zhou, D.; Platt, J. C.; Basu, S.; and Mao, Y. 2012. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, 2204–2212.