Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 2535329, 11 pages
http://dx.doi.org/10.1155/2016/2535329

Research Article

Exploiting Explicit and Implicit Feedback for Personalized Ranking

Gai Li[1] **and Qiang Chen**[2]

[1] *School of Electronics and Information Engineering, Shunde Polytechnic, Guangdong, Shunde 528333, China*
[2] *Department of Computer Science, Guangdong University of Education, Guangdong, Guangzhou 510303, China*

Correspondence should be addressed to Gai Li; ligai999@126.com

Received 14 October 2015; Accepted 12 November 2015

Academic Editor: Anna Pandolfi

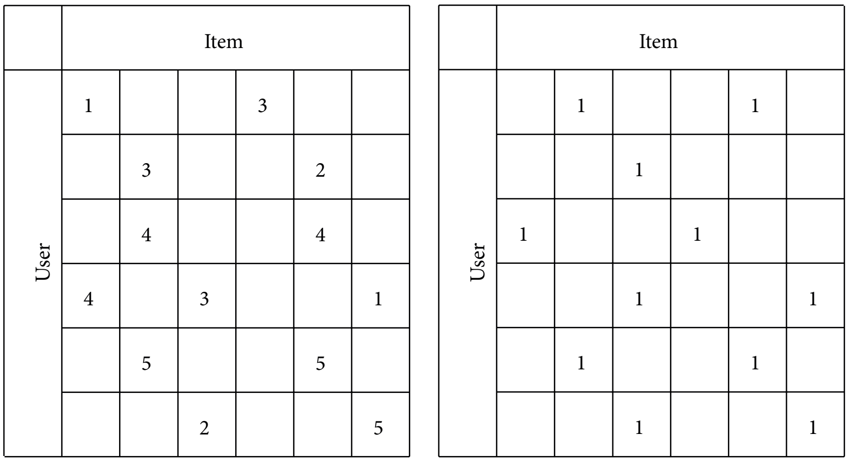
The problem of the previous researches on personalized ranking is that they focused on either explicit feedback data or implicit feedback data rather than making full use of the information in the dataset. Until now, nobody has studied personalized ranking algorithm by exploiting both explicit and implicit feedback. In order to overcome the defects of prior researches, a new personalized ranking algorithm (MERR_SVD++) based on the newest xCLiMF model and SVD++ algorithm was proposed, which exploited both explicit and implicit feedback simultaneously and optimized the well-known evaluation metric Expected Reciprocal Rank (ERR). Experimental results on practical datasets showed that our proposed algorithm outperformed existing personalized ranking algorithms over different evaluation metrics and that the running time of MERR_SVD++ showed a linear correlation with the number of rating. Because of its high precision and the good expansibility, MERR_SVD++ is suitable for processing big data and has wide application prospect in the field of internet information recommendation.

1. Introduction

As e-commerce is growing in popularity, an important challenge is helping customers sort through a large variety of offered products to easily find the ones they will enjoy the most. One of the tools that address this challenge is the recommender system, which is attracting a lot of attention recently [1–4]. Recommender systems are a subclass of information filtering systems that seek to predict the “rating” or “preference” that users would give to an item [1]. Preferences for items are learned from users’ past interactions with the system, such as purchase histories or the click logs. The purpose of the system is to recommend items that users might like from a large collection. Recommender systems have been applied to many areas on the Internet, such as the e-commerce system Amazon, the DVD rental system Netflix, and Google News. Recommender systems are usually classified into three categories based on how recommendations are made: content-based recommendations, collaborative filtering (CF), and hybrid approaches. In these approaches, collaborative filtering is the most widely used and the most successful.

Recently, collaborative filtering algorithm has been widely studied in both academic and industrial fields. The data processed by collaborative filtering algorithm are divided into two categories: explicit feedback data (e.g., ratings, votes) and implicit feedback data (e.g., clicks, purchases). Explicit feedback data are more widely used in the research fields of recommender system [1, 2, 4–7]. They are often in the form of numeric ratings from users to express their preferences regarding specific items. Implicit feedback data are easier to collect. The research on implicit feedback about CF is also called One-Class Collaborative Filtering (OCCF) [8–15], in which only positive implicit feedback or only positive examples can be observed. The explicit and implicit feedback data can be expressed in matrix form as shown in Figure 1. In the explicit feedback matrix, an element can be any real number, but often ratings are integers in the range (1~5), such as the ratings on Netflix, where a missing element represents a missing example. In the implicit feedback matrix, the positive-only user preferences data can be represented as a single-valued matrix.

Collaborative filtering algorithms also can be divided into two categories: collaborative filtering (CF) algorithms based



	Item					
User	1			3		
		3			2	
		4			4	
	4		3			1
		5			5	
			2			5

(a)

	Item					
User		1			1	
			1			
	1			1		
			1			1
		1			1	
			1			1

(b)

Figure 1: Examples of an explicit feedback matrix (a) and an implicit feedback matrix (b) for a recommender system.

on rating prediction [2, 4, 5, 8, 9, 12, 16, 17] and personalized ranking (PR) algorithms based on ranking prediction [3, 6, 7, 10, 11, 13–15, 18]. In collaborative filtering algorithms based on rating prediction, one predicts the actual rating for an item that a customer has not rated yet and then ranks the items according to the predicted ratings. On the other hand, for personalized ranking algorithms based on ranking prediction, one predicts a preference ordering over the yet unrated items without going through the intermeditate step of rating prediction. Actually, from the recommendation perspective, the order over the items is more important than their rating. Therefore, in this paper, we focus on personalized ranking algorithms based on ranking prediction.

The problem of the previous researches on personalized ranking is that they focused on either explicit feedback data or implicit feedback data rather than making full use of the information in the dataset. However, in most real world recommender systems both explicit and implicit user feedback are abundant and could potentially complement each other. It is desirable to be able to unify these two heterogeneous forms of user feedback in order to generate more accurate recommendations. The idea of complementing explicit feedback with implicit feedback was first proposed in [16], where the author considered explicit feedback as how the users rated the movies and implicit feedback as what movies were rated by the users. The two forms of feedback were combined via a factorized neighborhood model (called Singular Value Decomposition++, SVD++), an extension of traditional nearest item-based model in which the item-item similarity matrix was approximated via low rank factorization. In order to make full use of explicit and implicit feedback, Liu et al. [17] proposed Co-Rating model, which developed matrix factorization models that could be trained from explicit and implicit feedback simultaneously. Both SVD++ and Co-Rating are based on rating prediction. Until now, nobody has studied personalized ranking algorithm by exploiting both explicit and implicit feedback.

In order to overcome the defects of prior researches, this paper proposes a new personalized ranking algorithm (MERR_SVD++), which exploits both explicit and implicit feedback and optimizes Expected Reciprocal Rank (ERR) based on the newest Extended Collaborative Less-Is-More Filtering (xCLiMF) model [18] and SVD++ algorithm. Experimental results on practical datasets showed that our proposed algorithm outperformed existing personalized ranking algorithms over different evaluation metrics and that the running time of MERR_SVD++ showed a linear correlation with the number of rating. Because of its high precision and the good expansibility, MERR_SVD++ is suitable for processing big data and has wide application prospect in the field of internet information recommendation.

The rest of this paper is organized as follows: Section 2 introduces previous related work; Section 3 demonstrates the problem formalization and SVD++ algorithm; a new personalized ranking algorithm (MERR_SVD++) is proposed in Section 4; the experimental results and discussion are presented in Section 5, followed by the conclusion and future work in Section 6.

2. Related Work

2.1. Rating Prediction. In conventional CF tasks, the most frequently used evaluation metrics are the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE). Therefore, rating prediction (such as the Netflix Prize) has been the most popular method for solving the CF problem. Rating prediction methods are always regression based: they minimize the error of predicted ratings and true ratings. The simplest algorithm for rating prediction is K-Nearest-Neighbor (KNN) [19], which predicts the missing ratings from the neighborhood of users or items. KNN is a memory-based algorithm, and one needs to compute all the similarities between different users or items. More efficient algorithms are model based: they build a model from the visible ratings and compute all the missing ratings from the model. Widely used model-based rating prediction methods include PLSA [20], the Restricted Boltzmann Machine (RBM) [21], and a series of matrix factorization techniques [22–25].

2.2. Learning to Rank. LTR is the core technology for information retrieval. When a query is input into a search engine, LTR is responsible for ranking all the documents or Web pages according to their relevance to this query or other objectives. Many LTR algorithms have been proposed recently, and they can be classified into three categories: pointwise, listwise, and pairwise [3, 26].

In the pointwise approach, it is assumed that each query-document pair in the training data has a numerical or ordinal score. Then the LTR problem can be approximated by a regression problem: given a single query-document pair, its score is predicted.

As the name suggests, the listwise approach takes the entire set of documents associated with a query in the training data as the input to construct a model and predict their scores.

The pairwise approach does not focus on accurately predicting the degree of relevance of each document; instead, it mainly cares about the relative order of two documents. In this sense, it is closer to the concept of "ranking."

2.3. Personalized Ranking Algorithms for Collaborative Filtering. The algorithms about personalized ranking can also be divided into two categories: personalized ranking with implicit feedback (PRIF) [6, 7, 10, 11, 13–15] and personalized ranking with explicit feedback (PREF) [18, 27–29]. The foremost of PRIF is Bayesian Personalized Ranking (BPR) [11], which converts the OCCF problem into a ranking problem. Pan et al. [13] proposed Adaptive Bayesian Personalized Ranking (ABPR), which generalized BPR algorithm for homogeneous implicit feedback and learned the confidence adaptively. Pan et al. [15] have proposed Group Bayesian Personalized Ranking (GBPR), via introducing richer interactions among users. In GBPR, it introduces group preference, to relax the individual and independence assumptions of BPR. Jahrer and Toscher [6] proposed SVD and AFM, which used a ranking based objective function constructed by matrix decomposition model and a stochastic gradient descent optimizer. Takács and Tikk [7] proposed RankALS, which presented a computationally effective approach for the direct minimization of a ranking objective function without sampling. Gai [10] proposed a new PRIF algorithm (Pairwise Probabilistic Matrix Factorization (PPMF)) to further improve the performance of previous PRIF algorithms. Recently, Shi et al. [14] proposed Collaborative Less-is-More Filtering (CLiMF), in which the model parameters were learned by directly maximizing the well-known information retrieval metric: Mean Reciprocal Rank (MRR). However, CLiMF is not suitable for other evaluation metrics (e.g., MAP, AUC, and NDCG). References [6, 7, 10, 11, 13–15] can improve the performance of OCCF by solving the data sparsity and imbalance problems of PRIF to a certain extent. As for PREF, [27] was adapted from PLSA and [28] employed the KNN technique of CF, both of which utilized the pairwise ranking method. Reference [29] utilized the listwise method to build its ranker, which was a variation of Maximum Margin Factorization [22]. Shi et al. [18] proposed Extended Collaborative Less-Is-More Filtering (xCLiMF) model, which could be seen as a generalization of the CLiMF method. The key idea of the xCLiMF algorithm is that it builds a recommendation model by optimizing Expected Reciprocal Rank, an evaluation metric that generalizes Reciprocal Rank (RR) in order to incorporate user' explicit feedback. References [18, 27–29] can also improve the performance of PREF by solving the data sparsity and imbalance problems of PREF to a certain extent.

3. Problem Formalization and SVD++

3.1. Problem Formalization. In this paper, we use capital letters to denote a matrix (such as X). Given a matrix X, X_{ij} represents its element, X_i indicates the ith row of X, X_j symbolizes the jth column of X, and X^T stands for the transpose of X.

3.2. SVD++. Given that a matrix $X = (X_{ij})_{m\times n}$, the total number of users is m, and the total number of items is n, if X is an explicit feedback matrix, then $X_{ij} \in \{0, 1, 2, 3, 4, 5\}$ or X_{ij} is unknown. We want to approximate X with a low rank matrix $\widehat{X} = (\widehat{X}_{ij})_{m\times n}$, where $\widehat{X} = U^T V$, $U \in C^{d\times m}$, $V \in C^{d\times n}$, U and V denote the explicit feature matrix with ranks of d for users and items, respectively, $d \ll k$, and k denotes the rank of X, $k \le \min(m, n)$.

If X is an implicit feedback matrix, then $X_{ij} \in \{0, 1\}$ or X_{ij} is unknown. We want to approximate X with a low rank matrix $\widehat{X} = (\widehat{X}_{ij})_{m\times n}$, where $\widehat{X} = H^T W$, $H \in C^{d\times m}$, $W \in C^{d\times n}$, H and W denote the implicit feature matrix with ranks of d for users and items, respectively.

SVD++ is a collaborative filtering algorithm unifying explicit and implicit feedback based on rating prediction and matrix factorization [16]. In SVD++, matrix V denotes the explicit and implicit feature matrix of items simultaneously.

The feature matrix of users can be defined as

$$U_u + |N(u)|^{-1/2} \sum_{j\in N(u)} W_j, \quad (1)$$

where U_u is used to characterize the user's explicit feedback, $|N(u)|^{-1/2} \sum_{j\in N(u)} W_j$ is used to characterize the user's implicit feedback, $N(u)$ denotes the set of all items that user u gave implicit feedback, and W_j denotes the implicit feature vector of item j.

So the prediction formula of $\widehat{X}_{ui}$ in SVD++ can be defined as

$$\widehat{X}_{ui} = \left(U_u + |N(u)|^{-1/2} \sum_{j\in N(u)} W_j\right)^T V_i. \quad (2)$$

4. Exploiting Explicit and Implicit Feedback for Personalized Ranking

In this section, we will firstly introduce our MERR_SVD++ model, then give the learning algorithm of this model, and finally analyze its computational complexity.

4.1. Exploiting Explicit and Implicit Feedback for Personalized Ranking. In practical applications, the user scans the results list from top to bottom and stops when a result is found that fully satisfies the user's information need. The usefulness of an item at rank i is dependent on the usefulness of the items at rank less than i. Reciprocal Rank (RR) is an important evaluation metric in the research field of information retrieval [14], and it strongly emphasizes the relevance of results returned at the top of the list. The ERR measure is a generalized version of RR designed to be used with multiple relevance level data (e.g., ratings). ERR has similar properties to RR in that it strongly emphasizes the relevance of results returned at the top of the list.

Using the definition of ERR in [18, 30], we can describe ERR for a ranked item list of user u as follows:

$$\mathrm{ERR}_u = \sum_{i=1}^{n} \frac{P_{ui}}{R_{ui}}, \tag{3}$$

where R_{ui} denotes the rank position of item i for user u, when all items are ranked in descending order of the predicted relevance values. And P_{ui} denotes the probability that the user u stops at position i. As in [30], P_{ui} is defined as follows:

$$P_{ui} = r_{ui} \prod_{j=1}^{n} \left(1 - r_{uj} I\left(R_{uj} < R_{ui}\right)\right), \tag{4}$$

where $I(x)$ is an indicator function, equal to 1 if the condition is true, and otherwise 0. And r_{ui} denotes the probability that user u finds the item i relevant. Substituting (4) into (3), we obtain the calculation formula of ERR_u:

$$\mathrm{ERR}_u = \sum_{i=1}^{n} \frac{r_{ui}}{R_{ui}} \prod_{j=1}^{n} \left(1 - r_{uj} I\left(R_{uj} < R_{ui}\right)\right). \tag{5}$$

We use a mapping function similar to the one used in [18], to convert ratings (or levels of relevance in general) to probabilities of relevance, as follows:

$$r_{ui} = \begin{cases} \dfrac{2^{X_{ui}} - 1}{2^{X_{\max}}}, & I_{ui} > 0 \\ 0 & I_{ui} = 0, \end{cases} \tag{6}$$

where I_{ui} is an indicator function. Note that $I_{ui} > 0$ ($I_{ui} = 0$) indicates that user u's preference to item i is known (unknown). X_{ui} denotes the rating given by user u to item i, and $X_{\max}$ is the highest rating.

In this paper, we define that $R(u)$ denotes the set of all items that user u gave explicit feedback, so $N(u) = R(u)$ in the dataset that the users only gave explicit feedback, and the implicit feedback dataset is created by setting all the rating data in explicit feedback dataset as 1. A toy example can be seen in Figure 2. If the dataset contains both explicit feedback data and implicit feedback data, $N(u) = R(u) + M(u)$, $M(u)$ denoting the set of all items that user u only gave implicit feedback. A toy example can be seen in Figure 3. The influence of implicit feedback on the performance of MERR_SVD++ can be found in Section 5.4.2. If we use the traditional SVD++ algorithm to unify explicit and implicit feedback data, the prediction formula of $\widehat{X}_{ui}$ in SVD++ can be defined as

$$\widehat{X}_{ui} = \left(U_u + |N(u)|^{-1/2} \sum_{j \in N(u)} W_j\right)^T V_i. \tag{7}$$

So far, through the introduction of SVD++, we can exploit both explicit and implicit feedback simultaneously by optimizing evaluation metric ERR. So we call our model MERR_SVD++.

Note that the value of rank R_{ui} depends on the value of $\widehat{X}_{ui}$. For example, if the predicted relevance value $\widehat{X}_{ui}$ of item i is the second highest among all the items for user u, then we will have $R_{ui} = 2$.

Similar to other ranking measures such as RR, ERR is also nonsmooth with respect to the latent factors of users and items, that is, U, V, and W. It is thus impossible to optimize ERR directly using conventional optimization techniques. We thus employ smoothing techniques that were also used in CLiMF [14] and xCLiMF [18], to attain a smoothed version of ERR. In particular we approximate the rank-based terms $1/R_{ui}$ and $I(R_{uj} < R_{ui})$ in (5) by smooth functions with respect to the model parameters U, V, and W. The approximate formula is as follows:

$$\begin{gathered} I\left(R_{uj} < R_{ui}\right) \approx g\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right), \\ \frac{1}{R_{ui}} = g\left(\widehat{X}_{ui}\right), \end{gathered} \tag{8}$$

where $g(x)$ is a logistic function, that is, $g(x) = 1/(1 + e^{-x})$.

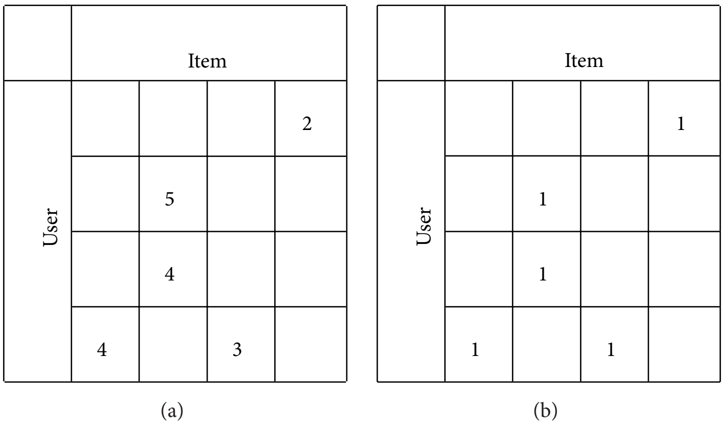
Substituting (8) into (5), we obtain a smoothed approximation of ERR_u:

$$\mathrm{ERR}_u = \sum_{i=1}^{n} r_{ui} g\left(\widehat{X}_{ui}\right) \prod_{j=1}^{n} \left(1 - r_{uj} g\left(\widehat{X}_{u(j-i)}\right)\right). \tag{9}$$

Note that for notation convenience, we make use of the substitution $\widehat{X}_{u(j-i)} = \widehat{X}_{uj} - \widehat{X}_{ui}$.

Given the monotonicity of the logarithm function, the model parameters that maximize (9) are equivalent to the parameters that maximize $\ln((1/|N(u)|)\mathrm{ERR}_u)$. Specifically, we have

$$\begin{aligned} U_u, V, W &= \underset{U_u, V, W}{\arg\max} \left\{\mathrm{ERR}_u\right\} \\ &= \underset{U_u, V, W}{\arg\max} \left\{\ln\left((1/N(u))\, \mathrm{ERR}_u\right)\right\} \\ &= \underset{U_u, V, W}{\arg\max} \left\{\ln\left(\sum_{j=1}^{n} \frac{r_{ui} g\left(\widehat{X}_{ui}\right)}{|N(u)|} \prod_{j=1}^{n} \left(1 - r_{uj} g\left(\widehat{X}_{u(j-i)}\right)\right)\right)\right\}. \end{aligned} \tag{10}$$



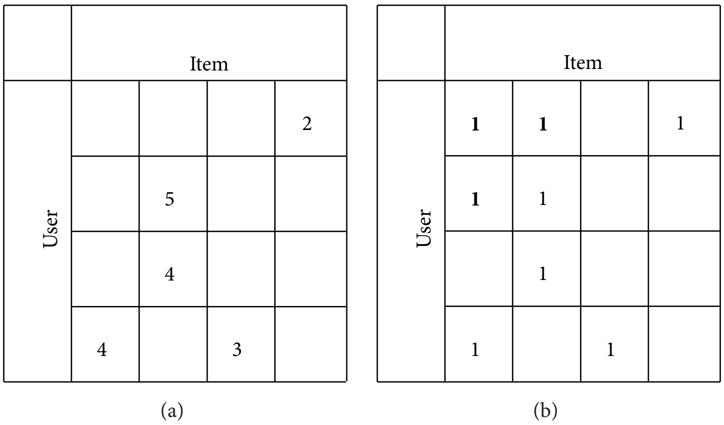
(a)

	Item			
User				2
		5		
		4		
	4		3	

(b)

	Item			
User				1
		1		
		1		
	1		1	

Figure 2: A toy example of the dataset that the users only gave explicit feedback. (a) denotes the explicit feedback dataset. (b) denotes the implicit feedback dataset.



(a)

	Item			
User				2
		5		
		4		
	4		3	

(b)

	Item			
User	**1**	**1**		1
	1	1		
		1		
	1		1	

Figure 3: A toy example of the dataset that contains both explicit feedback data and implicit feedback data. (a) denotes the explicit feedback dataset. (b) denotes the implicit feedback dataset, and the numbers in bold denote the dataset that users only gave implicit feedback.

Based on Jensen's inequality and the concavity of the logarithm function in a similar manner to [14, 18], we derive the lower bound of $\ln((1/|N(u)|)\mathrm{ERR}_u)$ as follows:

$$\begin{aligned}
&\ln\left(\frac{1}{|N(u)|}\mathrm{ERR}_u\right) \\
&\quad = \ln\left[\sum_{j=1}^{n}\frac{r_{ui}g(\widehat{X}_{ui})}{|N(u)|}\prod_{j=1}^{n}\left(1-r_{uj}g\left(\widehat{X}_{u(j-i)}\right)\right)\right] \\
&\quad \geq \frac{1}{|N(u)|} \\
&\quad \cdot \sum_{j=1}^{n} r_{ui}\ln\left[g\left(\widehat{X}_{ui}\right)\prod_{j=1}^{n}\left(1-r_{uj}g\left(\widehat{X}_{u(j-i)}\right)\right)\right] \\
&\quad = \frac{1}{|N(u)|} \\
&\quad \cdot \sum_{j=1}^{n} r_{ui}\left[\ln g\left(\widehat{X}_{ui}\right)+\sum_{j=1}^{n}\ln\left(1-r_{uj}g\left(\widehat{X}_{u(j-i)}\right)\right)\right].
\end{aligned} \tag{11}$$

We can neglect the constant $1/|N(u)|$ in the lower bound and obtain a new objective function:

$$\begin{aligned}
&L\left(U_u, V, W\right) \\
&\quad = \sum_{j=1}^{n} r_{ui}\left[\ln g\left(\widehat{X}_{ui}\right)+\sum_{j=1}^{n}\ln\left(1-r_{uj}g\left(\widehat{X}_{u(j-i)}\right)\right)\right].
\end{aligned} \tag{12}$$

Taking into account all m users and using the Frobenius norm of the latent factors for regularization, we obtain the objective function of MERR_SVD++:

$$\begin{aligned}
L(U,V,W) &= \sum_{u}^{m}\sum_{i=1}^{n} r_{ui}\left[\ln g\left(\widehat{X}_{ui}\right)\right. \\
&\quad \left. + \sum_{j=1}^{n}\ln\left(1-r_{uj}g\left(\widehat{X}_{u(j-i)}\right)\right)\right] - \frac{\lambda}{2}\left(\|U\|^2+\|V\|^2\right. \\
&\quad \left. + \|W\|^2\right),
\end{aligned} \tag{13}$$

in which λ denotes the regularization coefficient and $\|U\|$ denotes the Frobenius norm of U. Note that the lower bound $L(U, V, W)$ is much less complex than the original objective function in (9), and standard optimization methods, for example, gradient ascend, can be used to learn the optimal model parameters U, V, and W.

4.2. Optimization. We can now maximize the objective function (13) with respect to the latent factors $\arg\max_{U,V,W} L(U, V, W)$. Note that $L(U, V, W)$ represents an approximation of the mean value of ERR across all the users. We can thus remove the constant coefficient $1/m$, since it has no influence on the optimization of $L(U, V, W)$. Since the objective function is smooth we can use gradient ascent for the optimization. The gradients can be derived in a similar manner to xCLiMF [18], as shown in the following:

$$\begin{aligned}\frac{\partial L}{\partial U_u} = \sum_{i=1}^{n} r_{ui} \Bigg[& g\left(-\widehat{X}_{ui}\right) V_i \\ & + \sum_{j=1}^{n} \frac{r_{uj} g'\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right)\left(V_i - V_j\right)}{1 - r_{uj} g\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right)} \Bigg] - \lambda U_u,\end{aligned} \tag{14}$$

$$\begin{aligned}\frac{\partial L}{\partial V_i} = r_{ui} \Bigg[& g\left(-\widehat{X}_{ui}\right) V_i + \sum_{j=1}^{n} r_{uj} g'\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right) \\ & \cdot \Bigg(\frac{1}{1 - r_{uj} g\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right)} \\ & - \frac{1}{1 - r_{ui} g\left(\widehat{X}_{ui} - \widehat{X}_{uj}\right)} \Bigg) \Bigg] \Bigg(U_u + |N(u)|^{-1/2} \\ & \cdot \sum_{k \in N(u)} W_k \Bigg) - \lambda V_i,\end{aligned} \tag{15}$$

$$\begin{aligned}\frac{\partial L}{\partial W_i} = r_{ui} \Bigg[& g\left(-\widehat{X}_{ui}\right) V_i \\ & + \sum_{j=1}^{n} \frac{r_{uj} g'\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right)\left(V_i - V_j\right)}{1 - r_{uj} g\left(\widehat{X}_{uj} - \widehat{X}_{ui}\right)} \Bigg] - \lambda W_i.\end{aligned} \tag{16}$$

The learning algorithm for the MERR_SVD++ model is outlined in Algorithm 1.

The published research papers in [8–11, 13, 18] show that the use of the normal distribution $N(0, 0.01)$ for the initialization of feature matrix is very effective, so we still use this approach for the initialization of U, V, and W in our proposed algorithm.

4.3. Computational Complexity. Here, we first analyze the complexity of the learning process for one iteration. By exploiting the data sparseness in X, the computational complexity of the gradient in (14) is $O(d\widehat{n}^2 m + dm)$. Note that $\widehat{n}$ denotes the average number of relevant items across all the users. The computational complexity of the gradient in (16) is also $O(d\widehat{n}^2 m + dm)$. The computational complexity of the gradient in (15) is $O(d\widehat{n}^2 m + d\widehat{n}m)$. Hence, the complexity of the learning algorithm in one iteration is in the order of $O(d\widehat{n}^2 m)$. In the case that $\widehat{n}$ is a small number, that is, $\widehat{n}^2 \ll m$, the complexity is linear to the number of users in the data collection. Note that we have $s = \widehat{n}m$, in which s denotes the number of nonzeros in the user-item matrix. The complexity of the learning algorithm is then $O(d\widehat{n}s)$. Since we usually have $\widehat{n} \ll s$, the complexity is $O(ds)$ even in the case that $\widehat{n}$ is large, that is, being linear to the number of nonzeros (i.e., relevant observations in the data). In sum, our analysis shows that MERR_SVD++ is suitable for large scale use cases. Note that we also empirically verify the complexity of the learning algorithm in Section 5.4.3.

```
Input: Training set $X$, learning rate $\alpha$, regularization $\lambda$,
The number of the feature $d$, and the maximal number
Of iterations itermax
Output: feature matrix $U$, $V$, $W$
  for $u = 1, 2, \ldots, m$ do
    %Index relevant items for user $u$;
    $N(u) = \{j \mid X_{ij} > 0, 0 \le j \le n\}$;
  end
  Initialize $U^{(0)}$, $V^{(0)}$ and $W^{(0)}$ with random values,
  and $t = 0$;
  repeat
    for $u = 1, 2, \ldots, m$ do
      %Update $U_u$;
      $U_u^{(t+1)} = U_u^{(t)} + \alpha \frac{\partial L}{\partial U_u^{(t)}}$;
      for $i \in N(u)$ do
      %Update $V_i$, $W_i$;
        $V_i^{(t+1)} = V_i^{(t)} + \alpha \frac{\partial L}{\partial V_i^{(t)}}$;
        $W_i^{(t+1)} = W_i^{(t)} + \alpha \frac{\partial L}{\partial W_i^{(t)}}$;
      End
    End
  $t = t + 1$;
  until $t \ge$ itermax
  $U = U^{(t)}$, $V = V^{(t)}$, $W = W^{(t)}$;
```

Algorithm 1: MERR_SVD++.

5. Experiment

5.1. Datasets. We use two datasets for the experiments. The first is the MovieLens 1 million dataset (ML1m) [18], which contains ca. 1M ratings (1–5 scale) from ca. 6K users and 3.7K movies. The sparseness of the ML1m dataset is 95.53%. The second dataset is the Netflix dataset [2, 16, 17], which contains 100,000,000 ratings (1–5 scale) from 480,189 users on 17,770 movies. Due to the huge size of the Netflix data, we extract a subset of 10,000 users and 10,000 movies, in which each user has rated more than 100 different movies.

5.2. Evaluation Metrics. Just as has been justified by [17], NDCG is a very suitable evaluation metric for personalized ranking algorithms which combine explicit and implicit feedback. And our proposed MERR_SVD++ algorithm exploits both explicit and implicit feedback simultaneously and optimizes the well-known personalized ranking evaluation metric Expected Reciprocal Rank (ERR). So we use the NDCG and ERR as evaluation metrics for the predictability of models in this paper.

NDCG is another most widely used measure for ranking problems. To define NDCG_u for a user u, one first needs to define DCG_u:

$$\mathrm{DCG}_u = \sum_{i=1}^{m} \frac{2^{\mathrm{pref}(i)} - 1}{\log(i+1)}, \tag{17}$$

where $\mathrm{pref}(i)$ is a binary indicator returning 1 if the ith item is preferred and 0 otherwise. DCG_u is then normalized by the ideal ranked list into the interval $[0, 1]$:

$$\mathrm{NDCG}_u = \frac{\mathrm{DCG}_u}{\mathrm{IDCG}_u}, \tag{18}$$

where IDCG_u denotes that the ranked list is sorted exactly according to the user's tastes: positive items are placed at the head of the ranked list. The NDCG of all the users is the mean score of each user.

ERR is a generalized version of Reciprocal Rank (RR) designed to be used with multiple relevance level data (e.g., ratings). It has similar properties to RR in that it strongly emphasizes the relevance of results returned at the top of the list. Using the definition of ERR in [18], we can define ERR for a ranked item list of user u as follows:

$$\mathrm{ERR}_u = \sum_{i=1}^{n} \frac{r_{ui}}{R_{ui}} \prod_{j=1}^{n} \left(1 - r_{uj} I\left(R_{uj} < R_{ui}\right)\right). \tag{19}$$

Similar to NDCG, the ERR of all the users is the mean score of each user.

Since in recommender systems the user's satisfaction is dominated by only a few items on the top of the recommendation list, our evaluation in the following experiments focuses on the performance of top-5 recommended items, that is, NDCG@5 and ERR@5.

5.3. Experiment Setup. For each dataset, we randomly selected 5 rated items (movies) and 1,000 unrated items (movies) for each user to form a test set. We then randomly selected a varying number of rated items from the rest to form a training set. For example, just as in [14, 18], under the condition of "Given 5," we randomly selected 5 rated items (disjoint to the items in the test set) for each user in order to generate a training set. We investigated a variety of "Given" conditions for the training sets, that is, 5, 10, and 15 for the ML1m dataset and 10, 20, and 30 for the extracted Netflix dataset. Generated recommendation lists for each user are compared to the ground truth in the test set in order to measure the performance.

All the models were implemented in MATLAB R2009a. For MERR_SVD++, the value of the regularization parameter λ was selected from range $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ and optimal parameter value was used. And the learning rate α was selected from set A, $A = \{\alpha \mid \alpha = 0.0001 \times 2^{c},\ \alpha \leq 0.5,\ c > 0$ and c is a constant$\}$, and the optimal parameter value was also used. In order to compare their performances fairly, for all matrix factorization models we set the number of features to be 10. The optimal values of all parameters for all the baseline models used are determined individually. More detailed setting methods of the parameters for all the baselines can be found in the corresponding references. For all the algorithms used in our experiments, we repeated the experiment 5 times for each of the different conditions of each dataset, and the performances reported were averaged across 5 runs.

5.4. Experiment Results. In this section we present a series of experiments to evaluate MERR_SVD++. We designed the experiments in order to address the following research questions:

(1) Does the proposed MERR_SVD++ outperform state-of-the-art personalized ranking approaches for top-N recommendation?

(2) Does the performance of MERR_SVD++ improve when we only increase the number of implicit feedback data for each user?

(3) Is MERR_SVD++ scalable for large scale use cases?

5.4.1. Performance Comparison. We compare the performance of MERR_SVD++ with that of five baseline algorithms. The approaches we compare with are listed below:

(i) Co-Rating [17]: a state-of-the-art CF model that can be trained from explicit and implicit feedback simultaneously.

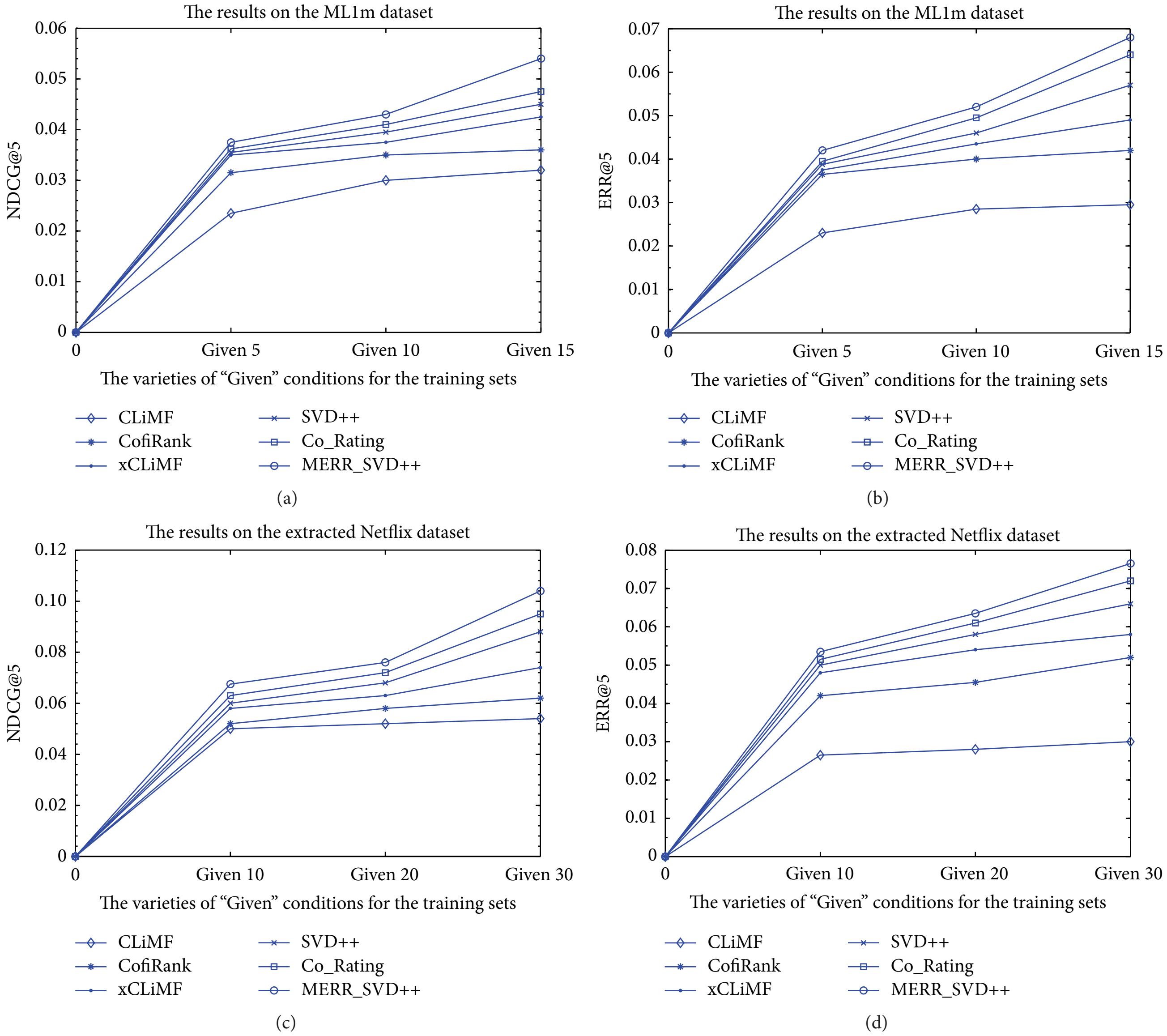
(ii) SVD++ [16]: the first proposed CF model that combines explicit and implicit feedback.

(iii) xCLiMF [18]: a state-of-the-art PR approach which aims at directly optimizing ERR, for top-N recommendation in domains with explicit feedback data (e.g., ratings).

(iv) CofiRank [29]: a PR approach that optimizes the NDCG measure [28] for domains with explicit feedback data (e.g., ratings). The implementation is based on the publicly available software package from the authors.

(v) CLiMF [14]: a state-of-the-art PR approach that optimizes the Mean Reciprocal Rank (MRR) measure [31] for domains with implicit feedback data (e.g., click, follow). We use the explicit feedback datasets by binarizing the rating values with a threshold. On the ML1m dataset and the extracted Netflix dataset, we take ratings 4 and 5 (the highest two relevance levels), respectively, as the relevance threshold for top-N recommendation.

The results of the experiments on the ML1m and the extracted Netflix datasets are shown in Figure 4. Rows denote



(a)

(b)

(c)

(d)

Figure 4: The performance comparison of MERR_SVD++ and baselines.

the varieties of "Given" conditions for the training sets based on the ML1m and the extracted Netflix datasets and columns denote the quality of NDCG and ERR. Figure 4 shows that MERR_SVD++ outperforms the baseline approaches in terms of both ERR and NDCG in all of the cases. The results show that the improvement of ERR aligns consistently with the improvement of NDCG, indicating that optimizing ERR would not degrade the utility of recommendations that are captured by the NDCG measure. It can be seen that the relative performance of MERR_SVD++ improves as the number of observed ratings from the users increases. This result indicates that MERR_SVD++ can learn better top-N recommendation models if more observations of the graded relevance data from users can be used. The results also reveal that it is difficult to model user preferences encoded in multiple levels of relevance with limited observations, in particular, when the number of observations is lower than the number of relevance levels.

Compared to Co-Rating, which is based on rating prediction, MERR_SVD++ is based on ranking prediction and succeeds in enhancing the top-ranked performance by optimizing ERR. As reported in [17], the performance of SVD++ is slightly weaker than that of Co-Rating, which is because SVD++ model only attempts to approximate the observed ratings and does not model preferences expressed in implicit feedback. The results also show that Co-Rating and SVD++ significantly outperform xCLiMF and CofiRank, which confirms our belief that implicit feedback could indeed complement explicit feedback. It can be seen in Figure 4 that xCLiMF significantly outperforms CLiMF in terms of

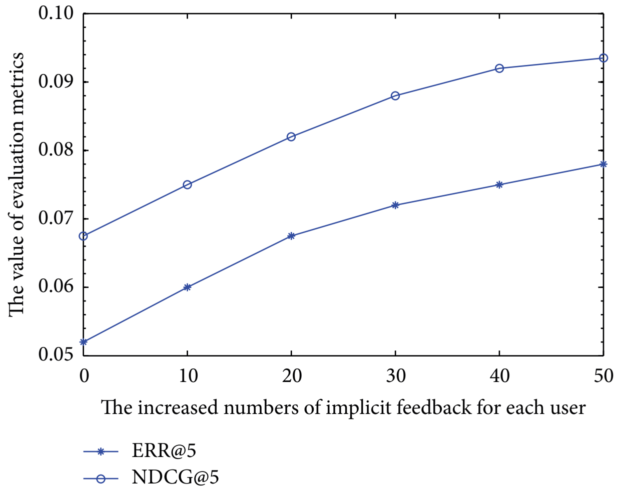


Figure 5: The influence of implicit feedback on the performance of MERR_SVD++.

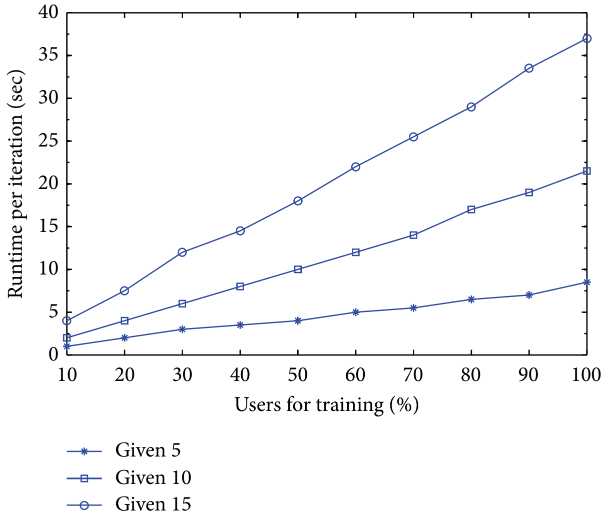


Figure 6: Scalability analysis of MERR_SVD++ in terms of the number of users in the training set.

both ERR and NDCG, across all the settings of relevance thresholds and datasets. The results indicate that the information loss from binarizing multilevel relevance data would inevitably make recommendation models based on binary relevance data, such as CLiMF, suboptimal for the use cases with explicit feedback data.

Hence, we give a positive answer to our first research question.

5.4.2. The Influence of Implicit Feedback on the Performance of MERR_SVD++. The influence of implicit feedback on the performance of MERR_SVD++ can be found in Figure 5. Here, $N(u) = R(u) + M(u)$. $M(u)$ denotes the set of all items that user u only gave implicit feedback. Rows denote the increased numbers of implicit feedback for each user and columns denote the quality of ERR and NDCG. In our experiment, the increased numbers of implicit feedback for each user are the same. We use the extracted Netflix dataset under the condition of "Given 10." Figure 5 shows that the quality of ERR and NDCG of MERR_SVD++ synchronously and linearly improves with the increase of implicit feedback for each user, which confirms our belief that implicit feedback could indeed complement explicit feedback.

With this experimental result, we give a positive answer to our second research question.

5.4.3. Scalability. The last experiment investigated the scalability of MERR_SVD++, by measuring the training time that was required for the training set at different scales. Firstly, as analyzed in Section 4.3, the computational complexity of MERR_SVD++ is linear in the number of users in the training set when the average number of items rated per user is fixed. To demonstrate the scalability, we used different numbers of users in the training set under each condition: we randomly selected from 10% to 100% users in the training set and their rated items as the training data for learning the latent factors. The results on the ML1m dataset are shown in Figure 6. We can observe that the computational time under each condition increases almost linearly to the increase of the number of users. Secondly, as also discussed in Section 4.3, the computational complexity of MERR_SVD++ could be further approximated to be linear to the amount of known data (i.e., nonzero entries in the training user-item matrix). To demonstrate this, we examined the runtime of the learning algorithm against different scales of the training sets under different "Given" conditions. The result is shown in Figure 6, from which we can observe that the average runtime of the learning algorithm per iteration increases almost linearly as the number of nonzeros in the training set increases.

The observations from this experiment allow us to answer our last research question positively.

6. Conclusion and Future Work

The problem of the previous researches on personalized ranking is that they focused on either explicit feedback data or implicit feedback data rather than making full use of the information in the dataset. Until now, nobody has studied personalized ranking algorithm by exploiting both explicit and implicit feedback. In order to overcome the defects of prior researches, in this paper we have presented a new personalized ranking algorithm (MERR_SVD++) by exploiting both explicit and implicit feedback simultaneously. MERR_SVD++ optimizes the well-known evaluation metric Expected Reciprocal Rank (ERR) and is based on the newest xCLiMF model and SVD++ algorithm. Experimental results on practical datasets showed that our proposed algorithm outperformed existing personalized ranking algorithms over different evaluation metrics and that the running time of MERR_SVD++ showed a linear correlation with the number of rating. Because of its high precision and the good expansibility, MERR_SVD++ is suitable for processing big data and can greatly improve the recommendation speed and validity by solving the latency problem of personalized recommendation and has wide application prospect in the field of internet information recommendation. And because MERR_SVD++ exploits both explicit and implicit feedback simultaneously, MERR_SVD++ can solve the data sparsity and imbalance problems of personalized ranking algorithms to a certain extent.

For future work, we plan to extend our algorithm to richer ones, so that our algorithm can solve the grey sheep problem and cold start problem of personalized recommendation. Also we would like to explore more useful information from the explicit feedback and implicit feedback simultaneously.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is sponsored in part by the National Natural Science Foundation of China (nos. 61370186, 61403264, 61402122, 61003140, 61033010, and 61272414), Science and Technology Planning Project of Guangdong Province (nos. 2014A010103040 and 2014B010116001), Science and Technology Planning Project of Guangzhou (nos. 2014J4100032 and 201510010203), the Ministry of Education and China Mobile Research Fund (no. MCM20121051), the second batch open subject of mechanical and electrical professional group engineering technology development center in Foshan city (no. 2015-KJZX139), and the 2015 Research Backbone Teachers Training Program of Shunde Polytechnic (no. 2015-KJZX014).

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] S. Ahn, A. Korattikara, and N. Liu, "Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC," in *Proceedings of the 21th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 401–410, ACM Press, Sydney, Australia, August 2015.

[3] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM '12)*, pp. 143–152, IEEE, Seattle, Wash, USA, February 2012.

[4] Q. Yao and J. Kwok, "Accelerated inexact soft-impute for fast large-scale matrix completion," in *Proceedings of the 24rd International Joint Conference on Artificial Intelligence*, pp. 4002–4008, ACM Press, Buenos Aires, Argentina, July 2015.

[5] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pp. 193–202, ACM, New York, NY, USA, August 2014.

[6] M. Jahrer and A. Toscher, "Collaborative filtering ensemble for ranking," in *Proceedings of the 17nd International Conference on Knowledge Discovery and Data Mining*, pp. 153–167, ACM, San Diego, Calif, USA, August 2011.

[7] G. Takács and D. Tikk, "Alternating least squares for personalized ranking," in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 83–90, ACM, Dublin, Ireland, 2012.

[8] G. Li and L. Li, "Dual collaborative topic modeling from implicit feedbacks," in *Proceedings of the IEEE International Conference on Security, Pattern Analysis, and Cybernetics*, pp. 395–404, Wuhan, China, October 2014.

[9] H. Wang, X. Shi, and D. Yeung, "Relational stacked denoising autoencoder for tag recommendation," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 3052–3058, ACM Press, Austin, Tex, USA, January 2015.

[10] L. Gai, "Pairwise probabilistic matrix factorization for implicit feedback collaborative filtering," in *Proceedings of the IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC '14)*, pp. 181–190, Wuhan, China, October 2014.

[11] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '09)*, pp. 452–461, Morgan Kaufmann, Montreal, Canada, 2009.

[12] L. Guo, J. Ma, H. R. Jiang, Z. M. Chen, and C. M. Xing, "Social trust aware item recommendation for implicit feedback," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 1039–1053, 2015.

[13] W. K. Pan, H. Zhong, C. F. Xu, and Z. Ming, "Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks," *Knowledge-Based Systems*, vol. 73, pp. 173–180, 2015.

[14] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "CLiMF: collaborative less-is-more filtering," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, pp. 3077–3081, ACM, Beijing, China, August 2013.

[15] W. K. Pan and L. Chen, "GBPR: group preference based bayesian personalized ranking for one-class collaborative filtering," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, pp. 2691–2697, Beijing, China, 2013.

[16] Y. Koren, "Factor in the neighbors: scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1–24, 2010.

[17] N. N. Liu, E. W. Xiang, M. Zhao, and Q. Yang, "Unifying explicit and implicit feedback for collaborative filtering," in *Proceedings of the 19th International Conference on Information and Knowledge Management (CIKM '10)*, pp. 1445–1448, ACM, Toronto, Canada, October 2010.

[18] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic, "XCLiMF: Optimizing expected reciprocal rank for data with multiple levels of relevance," in *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*, pp. 431–434, ACM, Hongkong, October 2013.

[19] J. Jiang, J. Lu, G. Zhang, and G. Long, "Scaling-up item-based collaborative filtering recommendation algorithm based on Hadoop," in *Proceedings of the 7th IEEE World Congress on Services*, pp. 490–497, IEEE, Washington, DC, USA, July 2011.

[20] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[21] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proceedings of the*

24rd International Conference on Machine Learning (ICML '07), pp. 791–798, ACM Press, Corvallis, Ore, USA, June 2007.

[22] N. Srebro, J. Rennie, and T. Jaakkola, "Maximum-margin matrix factorization," in *Proceedings of the 18th Annual Conference on Neural Information Processing Systems*, pp. 11–217, British Columbia, Canada, 2004.

[23] J. T. Liu, C. H. Wu, and W. Y. Liu, "Bayesian probabilistic matrix factorization with social relations and item contents for recommendation," *Decision Support Systems*, vol. 55, no. 3, pp. 838–850, 2013.

[24] D. Feldman and T. Tassa, "More constraints, smaller coresets: constrained matrix approximation of sparse big data," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*, pp. 249–258, ACM Press, Sydney, Australia, August 2015.

[25] S. Mirisaee, E. Gaussier, and A. Termier, "Improved local search for binary matrix factorization," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 1198–1204, ACM Press, Austin, Tex, USA, January 2015.

[26] T. Y. Liu, *Learning to Rank for Information Retrieval*, Springer, New York, NY, USA, 2011.

[27] N. N. Liu, M. Zhao, and Q. Yang, "Probabilistic latent preference analysis for collaborative filtering," in *Proceedings of the 18th International Conference on Information and Knowledge Management (CIKM '09)*, pp. 759–766, ACM Press, Hong Kong, November 2009.

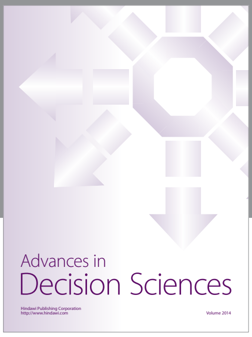
[28] N. N. Liu and Q. Yang, "EigenRank: a ranking-oriented approach to collaborative filtering," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR '08)*, pp. 83–90, ACM, Singapore, July 2008.

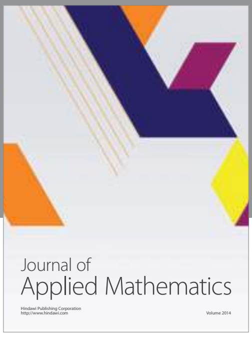
[29] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "CofiRank–maximum margin matrix factorization for collaborative ranking," in *Proceedings of the 21th Conference on Advances in Neural Information Processing Systems*, pp. 79–86, Curran Associates, Vancouver, Canada, 2007.

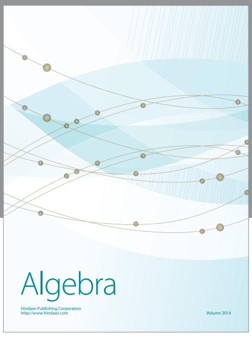
[30] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM International Conference on Information and Knowledge Management (CIKM '09)*, pp. 621–630, ACM, New York, NY, USA, November 2009.

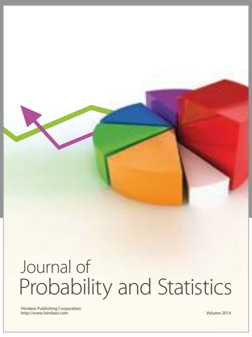
[31] E. M. Voorhees, "The trec-8 question answering track report," in *Proceedings of the 8th Text Retrieval Conference (TREC-8 '99)*, Gaithersburg, Md, USA, November 1999.

Advances in
Operations Research

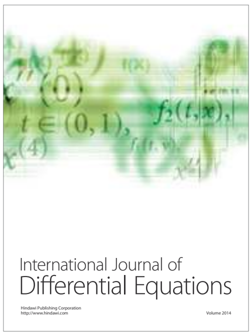

Advances in
Decision Sciences

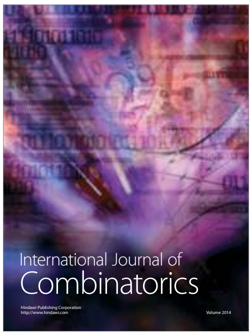

Journal of
Applied Mathematics

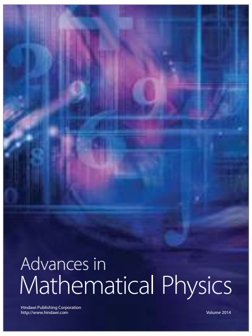

Algebra

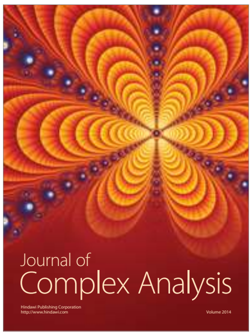

Journal of
Probability and Statistics

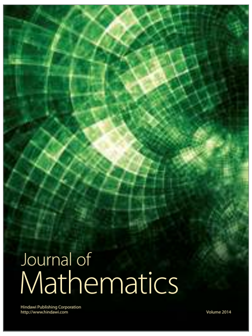
The Scientific
World Journal

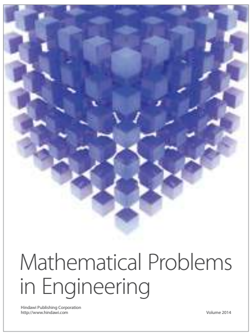

International Journal of
Differential Equations

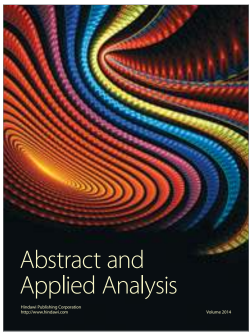

International Journal of
Combinatorics

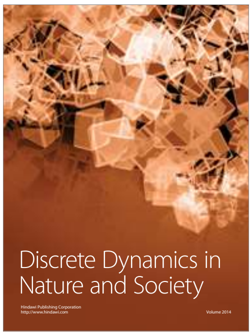

Advances in
Mathematical Physics

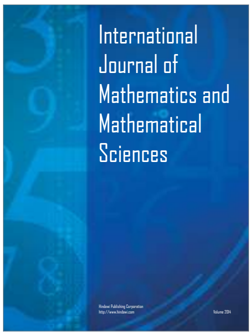

Journal of
Complex Analysis


Journal of
Mathematics


Mathematical Problems
in Engineering

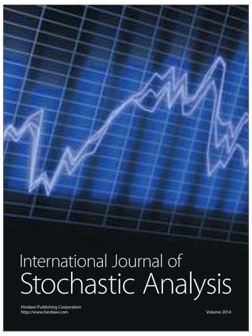

Abstract and
Applied Analysis


Discrete Dynamics in
Nature and Society


International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces


International Journal of
Stochastic Analysis

Journal of
Optimization