Theme Article: Special Issue on Intelligent Recommendation with Advanced AI and Learning

Collaborative Filtering With Ranking-Based Priors on Unknown Ratings

Jin Chen
University of Electronic Science and Technology of China

Kai Zheng
University of Electronic Science and Technology of China

Defu Lian
University of Science and Technology of China

***Abstract*—Advanced collaborative filtering methods based on explicit feedback assume that unknown ratings are missing not at random. The state-of-the-art algorithm hypothesizes that unknown items are weakly rated and sets an explicit prior to unknown ratings. However, the prior assuming unknown ratings be close to zero may be questionable and it is challenging to set appropriate prior ratings for unknown items. In this article, to avert the use of prior ratings, we propose a ranking-based prior by hypothesizing that each user's unknown ratings are close to each other. This prior essentially acts as a regularizer to penalize the discrepancy of predicted ratings between any two unknown items. With the ranking-based prior, we design a generic collaborative filtering framework for explicit feedback and develop an efficient optimization algorithm for parameter learning. We finally evaluate the proposed algorithms on four real-world rating datasets. The results show that the proposed algorithms consistently outperform the state-of-the-art baselines and that the ranking-based prior leads to superior recommendation accuracy.**

Digital Object Identifier 10.1109/MIS.2020.3000012
Date of publication 4 June 2020; date of current version 29 October 2020.

Published by the IEEE Computer Society

With the rapid development of digital media, recommendation systems are ubiquitous in daily life, which brings convenience to personal life and brings considerable profit to enterprises. At the same time, information grows explosively which benefits exploration of user potential interests but leads to information overload. There are two common types of data in recommendation systems. The first type is implicit feedback which is positive-only and is a mixture of negative and unseen data with unknown data. Another type is explicit feedback which contains ratings ranging from 1 to 5, but is rare compared with implicit feedback. Both types of data contain abundant unknown ratings which attracts a lot of studies for better recommendations.

In earlier researches, unknown ratings are assumed *missing at random* (MAR), that is the probability of a rating to be missing does not depend on its value. Depending on the assumption, only observed data is used for learning on full data. However, the *missing not at random* (MNAR), investigated by Marlin, has been validated. Building on the MNAR assumption, joint learning of unknown data helps reduce the bias of estimation produced by learning of observed data and several collaborative filtering algorithms have been proposed for improving recommendation accuracy.

The state-of-the-art algorithm hypothesizes unknown items to be weakly rated, which corresponds with the MNAR assumption, as evidence provided by Marlin shows that low ratings are much more likely to be missing from the observed data than high ratings. Then, an explicit prior is set on unknown ratings and suggests that the prediction of unknown ratings should not deviate far from zero values. However, such a strong prior may be questionable since it assigns worst ratings to unknown items. It is challenging to set appropriate prior ratings to unknown items.

To avert the use of explicit prior ratings, we propose a ranking-based prior, which hypothesizes each user's unknown ratings are likely to be lower values and should not deviate from each other for a user. This prior essentially acts as a regularizer to penalize the discrepancy of predicted ratings between any two unknown items for each user.

There are two common tasks in recommendation systems: rating prediction and recommending unseen items for users. Rating prediction aims to predict the missing ratings in the user-item matrix and regression-based models are proposed to improve prediction accuracy, while the aim of the latter one is to provide an appropriate ranking list and ranking-based models are designed. Obviously, the aim of recommendations of unseen items is more aligned with the goal of recommendation, and this work focuses on the optimization of the ranking-based model. Since more ranking information is included in explicit feedback, where users directly report their preferences on items, it is natural to exploit our prior to ranking-based explicit framework.

With the ranking-based prior, we design a generic collaborative filtering framework for explicit feedback. Regularizing the ranking-base loss functions, the unknown items are likely rated by low values than observed items, and items with higher ratings are ranked higher than lower rated items related to their truth ratings. As optimizing the pair-wise loss function is time-consuming and a quantity of operations on the sparse rating matrix are needed, to disengage the dependence of updating latent factors among items, we then develop an efficient optimization algorithm based on the caching strategy for parameter learning. The time complexity is then dramatically reduced, being only linearly proportional to the number of ratings in each round. To our best knowledge, this is the first ranking-based interpretation of the missing data in a matrix factorization framework.

The contributions of this article are summarized as follows.

- This work is the first to propose a ranking-based prior on unknown ratings in a matrix factorization framework, to incorporate the MNAR assumption.
- We develop an efficient optimization algorithm for fast updating latent factors. The time complexity in each round is linearly proportional to the number of ratings.
- We evaluate the proposed algorithms on four real-world rating datasets. The results show that the proposed algorithm is significantly

better than the state-of-the-art baselines. Moreover, the ranking-based prior is also verified to bring improvements of ranking quantity.

RELATED WORK

Rating prediction and recommendation of items are two main tasks in recommendation systems. Rating prediction aims to improve the prediction accuracy in the rating matrix while item recommendation aims to provide a ranking list depending on the user preferences.

Rating prediction is usually studied with explicit feedback and depending on the assumption that similar users prefer similar items. Some neighborhood-based recommendation methods are proposed, such as User-KNN, Item-KNN. The matrix factorization method assumes that the rating matrix can be decomposed into low-rank matrices. There are several methods for optimizing, such as directly decomposition with SVD, alternating least square (ALS), coordinate descent, and stochastic gradient descent. Probabilistic models are also proposed for matrix factorization methods, where the model factor variables are assumed to be marginally independent while rating variables are assumed to be conditionally independent. Several Bayesian models using variational approximations for performing inference are proposed such as BPMF and HPF.

These approaches also differ in how to incorporate the missing data. The learning model based on MAR, which assumes that the rating of the missing data does not depend on the value of rated data and other values, has shown good performances in some cases. However, the assumption is sometimes violated in recommender systems and several works validated the hypothesis that the missing data are MNAR, which indicates that averaging only observed ratings can be severely biased. In Sindhwani's work, the missing ratings are optimized variables and an EM algorithm is proposed for estimation for missing data. Recently, two approaches are proposed to address the MNAR problem and improve the performances a lot: the error-imputation-based (EIB) approach and the inverse-propensity-scoring (PIS) approach. The EIB approach suffers from large bias while PIS approach suffers from high variance.

Most of the works regularize the loss function of regression models depending on MNAR assumption. However, ranking-based models are also important in recommendation systems. Pair-wise methods are proposed for collaborative filtering, which minimizes the difference between the two ranked items and has a benefit of avoiding the bias of different users. A list-wise approach is recently proposed depending on a permutation model. However, the influence of missing data is ignored in ranking-based models. In this article, we corporate the unknown data for ranking-based collaborative filtering depending on the MNAR assumption to improve the ranking performance.

PROBLEM DEFINITION

For most explicit feedback, the rating r_{ui}, which usually ranges from 1 to 5 in rating matrix $\mathbf{R}$, represents the score on the ith item rated by the uth user. In this article, we assume that all the ratings are positive and that the items with higher ratings are more likely to be preferred.

There are N items that have interactions with M users in a dataset. The set $U = \{u_1, u_2, \ldots, u_m\}$ is the set of all users and the set $E_u = \{i \in E | r_{ui} > 0\}$ denotes the items the uth user rated where $E = \{e_1, e_2, \ldots, e_n\}$ is the set of all items. Similarly, the set $U_l = \{u \in U | r_{ul} > 0\}$ denotes the users who rated the lth item. Matrix factorization method is widely used in collaborative filtering recommendation system, which learns the latent matrix for the user and items and predicts the ratings. Matrices $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ denote the latent factor matrix for users and items. The predict rating $\hat{r}_{ui}$ is the inner product of $\mathbf{p}_u \in \mathbb{R}^K$ and $\mathbf{q}_i \in \mathbb{R}^K$.

In this article, we pay attention to the problem of ranking-based models. A simple objective function can be formulated as

$$\min_{\mathbf{P},\mathbf{Q}} L(r_{uij}, \hat{r}_{uij}) + \lambda R(\mathbf{P}, \mathbf{Q})$$

where $r_{uij} = r_{ui} - r_{uj}$ and $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$. L is the loss function for the related items and R is the function of regularizer. λ is the parameter to control the influence of the regularizer.

Table 1. Notations.

Notation	Representations
E_u	$E_u = \{i \in E | r_{ui} > 0\}$
U_i	$U_i = \{u \in U | r_{ui} > 0\}$
$\mathbf{p}_u$	the latent vector for the uth user
$\mathbf{q}_i$	the latent vector for the ith item
$\mathbf{Q}_u$	$\mathbf{Q}_{[E_u]}$, the submatrix of $\mathbf{Q}$
$\mathbf{P}_l$	$\mathbf{P}_{[U_l]}$, the submatrix of $\mathbf{P}$
$\tilde{\mathbf{q}}$	$\sum_i \mathbf{q}_i$, the sum of the item latent vectors
$\tilde{\mathbf{q}}_u$	$\sum_{i \in E_u} \mathbf{q}_i$
r_{ui}	the real rating for user u and item i
$\hat{r}_{ui}$	$\mathbf{p}_u^T \mathbf{q}_i$, the estimate rating
N_u	the number of the items the uth user rated
C_u	the number of the unobserved items
r_{uij}	$r_{ui} - r_{uj}$
$\hat{r}_{uij}$	$\hat{r}_{ui} - \hat{r}_{uj}$
$\tilde{\mathbf{Q}}$	each row of $\tilde{\mathbf{Q}}$ is $\tilde{\mathbf{q}}_u$
$\|R\|$	the number of the observed ratings ($r_{ui} > 0$)

For the sake of simplicity, we summarize the notations referred in Table 1.

RECOMMENDATION FOR EXPLICIT FEEDBACK

We first motivate the design of the ranking-based prior on unknown ratings and then propose a generic collaborative filtering framework for explicit feedback with the ranking-based prior. The framework is denoted as CF-RPU for short.

Ranking-Based Prior on Unknown Ratings

In most matrix factorization models, latent representation of users and items, after being learned from rated data, are used to predict the ratings for all unknown items. However, the unknown data usually follow the MNAR assumption. The estimation of ratings on full data only depending on observed data is biased, which indicates that the process of missing values is important to improve the ranking performance. Based on the MNAR assumption, which shows that high rating items are more rated items than low rating items, the state-of-the-art algorithm, introduced by Devooght, proposes a prior on unknown data which is formulated as

$$\sum_u \sum_{i \notin E_u} (\hat{r}_0 - \mathbf{p}_u^T \mathbf{q}_i)^2 = \sum_u \sum_{i \notin E_u} (0 - \mathbf{p}_u^T \mathbf{q}_i)^2$$

where $\hat{r}_0$ is the estimate prior on the unknown data, being suggested to be zero which drives the prediction of unknown items not far from zero. From this perspective, the ratings for unknown items should be close to a low value, which implies some similarities between unknown items. Furthermore, it is quite challenging to set an appropriate explicit prior on the unknown ratings.

In order to avert the use of uniform prior ratings on all unknown data, we propose a ranking-based prior, by hypothesizing that for each user the ratings on unknown data are close to each other. In other words, this ranking-based prior essentially acts as a regularizer to penalize the discrepancy of predicted ratings between any two unknown items for a user. Then, the ranking-based prior on unknown data is formulated as follows:

$$\begin{aligned}\mathcal{W} &= \sum_u \sum_{i,j \notin E_u} (\hat{r}_{ui} - \hat{r}_{uj})^2 \\ &= \sum_u \sum_{i,j \notin E_u} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2.\end{aligned}$$

Compared with the explicit prior, the ranking-based prior successfully averts the use of explicit prior ratings for unknown data. In fact, we may set an implicit prior on unknown ratings, which is eliminated by the difference of predicted ratings between two unknown items. Furthermore, the implicit prior for each user is different which avoids the uniform value of the prior for all unknown data.

However, direct computation of the ranking-based prior on unknown data is time consuming and we rewrite it as

$$\mathcal{W} = \mathcal{W}_0 - 2\mathcal{W}_1 + \mathcal{W}_2$$

where

$$\begin{aligned}\mathcal{W}_0 &= \sum_u \sum_{i,j} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \\ \mathcal{W}_1 &= \sum_u \sum_{i \in E_u} \sum_j (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2 \\ \mathcal{W}_2 &= \sum_u \sum_{i,j \in E_u} (\mathbf{p}_u^T \mathbf{q}_i - \mathbf{p}_u^T \mathbf{q}_j)^2.\end{aligned}$$

In this way, we expand the computation into two parts: computation on full set and observed data which is quickly accessible in the sparse rating matrix. Below, we derive each part with respect to the user and latent vectors. We obtain the final derivation of the user latent vector $\mathbf{p}_u$ as

$$\frac{\partial \mathcal{W}}{\partial \mathbf{p}_u} = (N - N_u)(\mathbf{Q}^T\mathbf{Q} - \mathbf{Q}_u^T\mathbf{Q}_u)\mathbf{p}_u - (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u)^T(\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u)\mathbf{p}_u \quad (1)$$

where $\tilde{\mathbf{q}} = \sum_j \mathbf{q}_j$ is the sum over all the item latent vectors and $\tilde{\mathbf{q}}_u = \sum_{i \in E_u} \mathbf{q}_i$ is the sum of the item latent vectors corresponding to the uth user. $\mathbf{Q}_u \in \mathbb{R}^{N_u \times K}$ is the submatrix of $\mathbf{Q}$ and each row is the latent vector of items rated by the uth user. N is the number of the items and N_u is the number of the items the uth user rated. Similarly, we derive $\mathcal{W}$ with respect to the item latent vector $\mathbf{q}_l$ as

$$\begin{aligned}\frac{\partial \mathcal{W}}{\partial \mathbf{q}_l} &= \sum_u C_u \mathbf{p}_u^T \mathbf{p}_u \mathbf{q}_l - \sum_u \mathbf{p}_u^T \mathbf{p}_u (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u) \\ &\quad - \sum_{u \in U_l} C_u \mathbf{p}_u^T \mathbf{p}_u \mathbf{q}_l + \sum_{u \in U_l} \mathbf{p}_u^T \mathbf{p}_u (\tilde{\mathbf{q}} - \tilde{\mathbf{q}}_u) \\ &= (\check{\mathbf{H}} - \check{\mathbf{H}}_l)\mathbf{q}_l - (\mathbf{H} - \mathbf{H}_l)\tilde{\mathbf{q}} + \mathbf{P}^T\tilde{r} - \mathbf{P}_l^T\tilde{r}_l.\end{aligned}$$

where $C_u = N - N_u$ is the number of unknown items according to a user u. $\check{\mathbf{H}} = \sum_u C_u \mathbf{p}_u^T \mathbf{p}_u = \mathbf{P}^T \mathrm{diag}([C_1, .., C_m])\mathbf{P}$ and $\check{\mathbf{H}}_l = \sum_{u \in U_l} C_u \mathbf{p}_u^T \mathbf{p}_u$. Similarly, $\mathbf{H} = \mathbf{P}^T\mathbf{P}$ and $\mathbf{H}_l = \mathbf{P}_l^T\mathbf{P}_l$. $\tilde{r} = \sum_u \mathbf{p}_u^T \tilde{\mathbf{q}}_u$ and $\tilde{r}_l = \sum_{u \in U_l} \mathbf{p}_u^T \tilde{\mathbf{q}}_u$. With the quadratic form, we can obtain the Hessian matrix which contributes to efficient update depending on Newton's method. Moreover, there is no external parameter or estimate in the regularizer.

Generic Framework for Explicit Feedback

There are two main tasks in recommender systems: rating prediction and ranking the candidate items. It is more likely that ranking-based matrix factorization models align better with the ultimate goal of recommendation. The advanced and efficient ranking-based matrix factorization tailored for the ranking objective is defined as

$$y = \sum_{u \in U} \sum_{i \in E_u} \sum_{j \in E} [(r_{ui} - r_{uj}) - (\hat{r}_{ui} - \hat{r}_{uj})]^2.$$

This objective function actually considers two ranking cases: 1) The comparisons between rated items that lead to a higher ranking position for highly rated items. 2) The comparisons between rated items and unknown items that ensure the rated items be ranked higher than the unknown items.

We integrate the ranking-based prior into the objective function, where more unknown items are estimated by a similar and small prior according to a user and derive a generic collaborative filtering framework for explicit feedback

$$\mathcal{L} = \sum_{u \in U} \sum_{i \in E_u} \sum_{j \in E_u} L(r_{uij}, \hat{r}_{uij}) + \beta \sum_{u \in U} \sum_{i \in E_u} \sum_{j \notin E_u} L(r_{ui}, \hat{r}_{uij}) + \lambda \mathcal{W} \quad (2)$$

$L(y, x)$ is the loss function and more generally, it can be square loss $(y - x)^2$, absolute loss $|y - x|$, logit loss $\mathrm{logit}(1 + e^{yx})$, and so on. β and λ are the parameters for tuning the influence of the corresponding part.

By imposing the ranking-based prior on unknown ratings, the objective function is more tailored for the ranking-based recommendation.

Noticing on the objective function (2), as there are much more unknown data, the numerical range of each part varies greatly. As a consequence, the ranking-based prior on unknown data has a dominant impact, which is also analyzed in Devooght's work by introducing a parameter ρ to adjust the weight of items. To reduce the sensitivity of the parameter, we propose a normalization strategy to deal with the imbalance problem

$$\begin{aligned}\mathcal{L}' &= \sum_{u \in U} \sum_{i,j \in E_u} p(N_u N_u) L(r_{uij}, \hat{r}_{uij}) \\ &\quad + \beta \sum_{u \in U} \sum_{i \in E_u} \sum_{j \notin E_u} p\big(N_u C_u\big) L(r_{ui}, \hat{r}_{uij}) \\ &\quad + \lambda \sum_{u} \sum_{i,j \notin E_u} p\big(C_u C_u\big)(\hat{r}_{ui} - \hat{r}_{uj})^2\end{aligned}$$

where N_u is the number of rated items for the uth user and C_u is the number of the unknown items. Obviously, $C_u = N - N_u$. $p(\cdot)$ is the function of normalization. $r_{uij} = r_{ui} - r_{uj}$ and $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$. Specifically, the function can be $p(x) = \frac{1}{x}$, $p(x) = \frac{1}{\log(x)}$, $p(x) = \frac{1}{\sqrt{x}}$, and other functions. In this way, we control the scale of each part and keep them close to each other which is beneficial to parameter tuning. Even when the

data sparsity increases, the small number of rated items still plays an important role.

APPLICATIONS AND OPTIMIZATION

In this section, we exploit specific methods into our proposed framework for recommendations. To be more specific, we choose the square loss and the normalization strategy based on the square root. We then design an effective optimization algorithm for parameter learning.

Loss Function

The square loss function is widely used in recommendation systems due to its simplicity. Furthermore, the differentiability of square loss functions has an edge over fast optimization. More importantly, with the square loss function, we can derive the analytical solution for updating latent factors.

Normalization

To reduce the sensitivity of the coefficients, we propose the normalization strategy to address the imbalanced problems and we below analyze the different functions for normalization. It is obvious that when the number gets large, the value of linear normalization function $(p(x) = \frac{1}{x})$ gets close to zero, which indicates that the influence of the unknown data will be dramatically weakened. Furthermore, it is also unreasonable for those users who rated more items. However, the strategies with the squared root function $(p(x) = \frac{1}{\sqrt{x}})$ and the logit function $(p(x) = \frac{1}{\log x})$ avoid the zero value which help maintain the influence of the component. We select the sqrt form $p(x) = \frac{1}{\sqrt{x}}$ function for normalization. Combining with square loss function and sqrt normalization, the objective function is defined as follows:

$$\begin{aligned}\mathcal{L}^{'} = &\sum_{u\in U}\sum_{i,j\in E_u} z\,(r_{uij} - \hat{r}_{uij})^2 \\ &+ \sum_{u\in U}\sum_{i\in E_u}\sum_{j\notin E_u} z_\beta (r_{ui} - \hat{r}_{uij})^2 \\ &+ \sum_{u}\sum_{i,j\notin E_u} z_\lambda (\hat{r}_{ui} - \hat{r}_{uj})^2\end{aligned} \tag{3}$$

where $z = \frac{1}{\sqrt{N_u N_u}}$, $z_\beta = \frac{\beta}{\sqrt{N_u C_u}}$, and $z_\lambda = \frac{\lambda}{\sqrt{C_u C_u}}$. Obviously, the numerous comparisons between plenty of items increase the complexity of learning. To disengage the dependence of updating latent factors among items, we propose an effective optimization algorithm for parameter learning.

Optimization With ALS

The ALS method is widely used in recommendation systems due to its fast convergence and easy parallelization. In this article, the ALS method is used for optimizing the square loss function.

We avoid the direct computation for unobserved items by splitting it into the computation for rated items and all items for quick updating on a huge sparse matrix.

Take the derivation of the user latent vector as an example, we derive the loss function with respect to the user latent vector as

$$\begin{aligned}\frac{\partial y}{\partial \mathbf{p}_u} = &z_\beta N_u \mathbf{Q}^T\mathbf{Q}\mathbf{p}_u + \gamma_0 \mathbf{Q}_u^T\mathbf{Q}_u\mathbf{p}_u - \mathbf{Q}_u^T\mathbf{r}_z \\ &+ \gamma_1\tilde{\mathbf{q}} + \gamma_2\tilde{\mathbf{q}}_u - (z - 2z_\beta)\mathbf{p}_u^T\tilde{\mathbf{q}}_u\tilde{\mathbf{q}}_u^T \\ &- z_\beta\mathbf{p}_u^T\tilde{\mathbf{q}}\tilde{\mathbf{q}}_u - z_\beta\mathbf{p}_u^T\tilde{\mathbf{q}}_u\tilde{\mathbf{q}}\end{aligned}$$

where $\mathbf{r}_z \in \mathbb{R}^{N_u}$ is a vector and each element is $zN_u r_{ui} - z_\beta C_u r_{ui}$ of the rated items, $\gamma_0 = (z - z_\beta)N_u + z_\beta C_u$, $\gamma_1 = \sum_{i\in E_u} z_\beta r_{ui}$, $\gamma_2 = \sum_{i\in E_u}(z - z_\beta)r_{ui}$, $\mathbf{Q}_u$ is the submatrix of $\mathbf{Q}$ which contains the latent vectors of the rated items by the user u. $\tilde{\mathbf{q}} \in \mathbb{R}^{1\times K}$ is the sum over $\mathbf{q}_j$ and $\tilde{\mathbf{q}}_u = \sum_{i\in E_u}\mathbf{q}_i$.

The gradient with respect to $\mathbf{p}_u$ can be rewritten as the multiplication of a Hessian matrix $\mathbf{A}_p \in \mathbb{R}^{K\times K}$ and a vector $\mathbf{b}_p$. We can set the gradient to zero, $\frac{\partial \mathcal{L}^{'}}{\partial \mathbf{p}_u} = \frac{\partial \hat{y}}{\partial \mathbf{p}_u} + z_\lambda \frac{\partial \mathcal{W}}{\partial \mathbf{p}_u} = 0$ and update the latent vector based on $\mathbf{A}_p^{-1}\mathbf{b}_p$. Furthermore, the values of $\mathbf{Q}^T\mathbf{Q}$ and $\tilde{\mathbf{q}}$ are independent to specific users, so that they can be precomputed.

Following the similar approach, the gradient of the objective function with respect to latent vector $\mathbf{q}_l$ of an item l is

$$\begin{aligned}\frac{\partial y}{\partial \mathbf{q}_l} = &(\dot{\mathbf{H}} + \breve{\mathbf{H}}_l)\mathbf{q}_l - \mathbf{t}_l\tilde{\mathbf{q}} - \breve{\mathbf{P}}^T\tilde{r} + \breve{\mathbf{P}}_l^T\tilde{r}_l \\ &- \mathbf{P}_l^T\mathbf{d}_l + \mathbf{P}^T\mathbf{f} + \mathbf{P}_l^T\mathbf{k}_l\end{aligned}$$

where $\dot{\mathbf{H}} = \sum_u z_1\mathbf{p}_u^T\mathbf{p}_u$ and $z_1 = z_\beta N_u$, $\breve{\mathbf{H}}_l = \sum_{u\in U_l} z_0\mathbf{p}_u^T\mathbf{p}_u$, and $z_0 = 2zN_u + z_\beta(N - 2N_u)$. $\mathbf{t}_l = \sum_{u\in U_l} z_\beta\mathbf{p}_u^T\mathbf{p}_u$. Each row of $\breve{\mathbf{P}}$ is $z_\beta\mathbf{p}_u$. Each row of $\breve{\mathbf{P}}$ is $(2z_\beta - 2z)\mathbf{p}_u$. Each element in vector $\mathbf{d}$ is $(2zN_u + z_\beta)r_{ul}$ related to item l. Each element in

vector $\mathbf{f}$ is $\sum_{i \in E_u} z_\beta r_{ui}$. Similarly, each element in vector $\mathbf{k}$ is $\sum_{i \in E_u} \big(2zN_u + z_\beta(N - N_u)\big) r_{ui}$.

Algorithm 1. CF-RPU Algorithm

Input: The rating matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$
Output: Latent factors $\mathbf{P} \in \mathbb{R}^{M \times K}$, and $\mathbf{Q} \in \mathbb{R}^{N \times K}$
Randomly initialize $\mathbf{P}$, $\mathbf{Q}$;
repeat
  Compute $\mathbf{Q}^T\mathbf{Q}, \tilde{\mathbf{q}}$;
  for $u = 1 : M$ **do**
    Calculate $\mathbf{Q}_u^T\mathbf{Q}_u$, $\tilde{\mathbf{q}}_u$;
    Calculate A_u, b_u;
    Update $\mathbf{p}_u = \mathbf{A}_u^{-1}\mathbf{b}_u$;
  for $u = 1 : M$ **do**
    $\tilde{\mathbf{Q}}[u:] \leftarrow \mathbf{Q}_u^T 1$;
    $\tilde{r}[u] \leftarrow \mathbf{p}_u \tilde{\mathbf{Q}}[u:]^T$;
    Compute related $\mathbf{H}, \mathbf{t}$;
  for $l = 1 : N$ **do**
    Compute related $\mathbf{H}_l, \mathbf{t}_l$;
    Calculate $\mathbf{A}_l$, $\mathbf{b}_l$;
    Update $\mathbf{q}_l = \mathbf{A}_l^{-1}\mathbf{b}_l$;
    $\tilde{\mathbf{q}} \leftarrow \tilde{\mathbf{q}} - \mathbf{q}_l^{old} + \mathbf{q}_l$;
    $\tilde{\mathbf{Q}}'[U_l:] \leftarrow \tilde{\mathbf{Q}}[U_l:] - \mathbf{q}_l^{old} + \mathbf{q}_l$;
    $\tilde{r}[U_l] \leftarrow \mathbf{P}[U_l:](\mathbf{q}_l - \mathbf{q}_l^{old})$;
until converage

The item latent vector $\mathbf{q}_l$ can be similarly updated by setting the gradient as 0. There is something different from updating the user latent vector where some important statistics are changing during the learning loops. Fortunately, we can adopt the caching and updating strategy by portioning the important statistics into several parts. Only $\tilde{r}$ and $\tilde{\mathbf{q}}$ are changing during the update of each item latent vector. Regarding updating $\tilde{\mathbf{q}}$ after $\mathbf{q}_l$ being learned, the new $\tilde{\mathbf{q}}$ can be updated as $\tilde{\mathbf{q}} - \mathbf{q}_{l\text{old}} + \mathbf{q}_{l\text{new}}$ rather than summing over all item latent vectors again. The computation of $\tilde{r}$ can be done by $\text{diag}(\mathbf{P}^T\tilde{\mathbf{Q}})$ and each row $\tilde{\mathbf{Q}}$ is $\tilde{\mathbf{q}}_u$. After learning $\mathbf{q}_l$, the related $\tilde{\mathbf{q}}_u$ are updated. After picking related columns in $\mathbf{P}$ and changed $\tilde{\mathbf{Q}}$, $\tilde{r}$ can be updated. In this way, the update of item latent factors can be quickly done which avoids recalculating some variables from scratch. Algorithm (1) shows the overall procedure of the proposed algorithm. After optimization, we can obtain the matrix $\mathbf{P}$ and $\mathbf{Q}$ to predict the ratings.

Table 2. Statistics of the datasets.

Datasets	Ratings	Users	Items	Sparsity
MovieLens	9 983 739	69 838	8939	83.669%
Yelp	2 103 895	77 277	45 638	99.940%
Amazon	4 701 968	158 650	128 939	99.978%
Netflix	100 396 329	463 770	17 764	98.781%

Time Complexity According to (3), the original time complexity of the objective function is $O(M(M+N)^2\ K^2 + (M+N)K^3)$ due to the numerous comparisons between plenty of unknown items. For explicit feedback, the number of the items a user rated is much smaller than the number of all items $(N_u << N)$ due to the sparsity of the data. The dimension K of the latent matrix is also much smaller than the number of the users and items. Our method reduces the time complexity to $O(|\mathbf{R}|K^2 + (M+N)K^3)$, where $|\mathbf{R}|$ is the number of the observed ratings in matrix $\mathbf{R}$. Note that $|\mathbf{R}|$ is much smaller than $M(M+N)^2$. The inversion of the matrix is assumed to take $O(K^3)$ time, even though more efficient algorithms exist, but probably are less relevant for the typically small values of dimensions.

EXPERIMENTS

In this part, we perform several experiments to investigate the two following key questions.

- Whether the proposed ranking-based prior on unknown ratings help improve ranking?
- Whether the proposed generic collaborative filtering framework for explicit feedback shows good performance?

Experiments Setup

Datasets In this work, we exploit our methods on four ground-truth datasets. The ratings of all these three datasets range from 1 to 5. The statistics of the datasets are shown in Table 2.

- *MovieLens*: The MovieLens dataset is a famous dataset for movie ratings. We choose the form of MovieLens10 M dataset, including 10 000 054 ratings from 71 567 users for 10 681 items. We filter out users and items with less than 20 ratings.
- *Amazon*: The Amazon dataset is sparser than MovieLens dataset and it contains the ratings for Amazon books, including 8 898 041 ratings.

We filter out users and items with less than 10 ratings.

- *Yelp*: The Yelp dataset is a famous dataset with businesses and reviews. We filter out the users and items with less than 10 interactions.
- *Netflix*: The volume of the dataset is the biggest in these four datasets. We filter out the users with less than 10 ratings and items with less than 20 ratings.

Data Splitting For each user, we randomly sample his 80% ratings as the training set and the rest 20% as the testing test. We fit a model to the training set and evaluate it in the test set. After tuning the parameters, we do five times cross-validation experiments and report the average performance.

Evaluation Metric As we focus on the task of ranking, we use two standard metrics of ranking evaluation: Normalized discounted cumulative gain (NDCG) and recall.

NDCG is a popular metric in ranking problems. It considers both the rating prediction and the position of the ratings. The order of the items recommended is meaningful in this problem. We cutoff the ranking list of 200 items for all users and compute the average NDCG as our final metric.

Recall is another popular metric in recommendation problem. It shows the performance of the correct prediction over the observed samples in the testing set and it ignores the order of the items. We also return the top-200 items for all users and compute the average recall.

Baselines We compare our method with the following methods.

- *RankALS*: RankALS utilizes a ranking-based objective function and optimizes function with an ALS solver. We set the parameter $s_j = 1$ which means the items are equally important. We report the results after 30 iterations.
- *SLwithPrior*: The method sets a prior on unknown items to help limit the bias of the learning on rated items. It is the state-of-the-art algorithm we focus on. We choose the square loss as the loss function. We tune the coefficient of the explicit prior under value 1e-4, 1e-3,1e-2,1e-1, and run within 30 iterations.
- *SQL-Rank*: SQL-Rank is a list-wise method for collaborative filtering and casts the ranking problem as maximum likelihood under a permutation model. We follow the model for explicit datasets. We set the learning rate as 0.1, the learning decay as 0.95, and report the results after 200 iterations. We tune the coefficient of the regularizer with 1e-3, 1e-2, and 1e-1.
- *MF-DR-JL*: The method proposed a double robust estimator by combining the EIB and IPS estimators to correct the error deviation for observed items and inversely weight the corrections with the propensity to consider the MNAR effect. We follow the matrix-factorization based model with the propensity being 1. We sample at least 50 unknown items in each round and tune the learning rate within 300 iterations. We tune the learning rate with 1e-3,1e-4, and the coefficient of the L2-regulation of 1e-5, 1e-4, and 1e-3.
- *POP*: Item popularity based, nonpersonalized baseline method.

CF-RPU sqrt is our proposed collaborative filtering framework for explicit feedback with square loss function and sqrt normalization. CF-RPU is our proposed method without sqrt normalization. For all the methods, we return the best parameter under NDCG@200.

Results

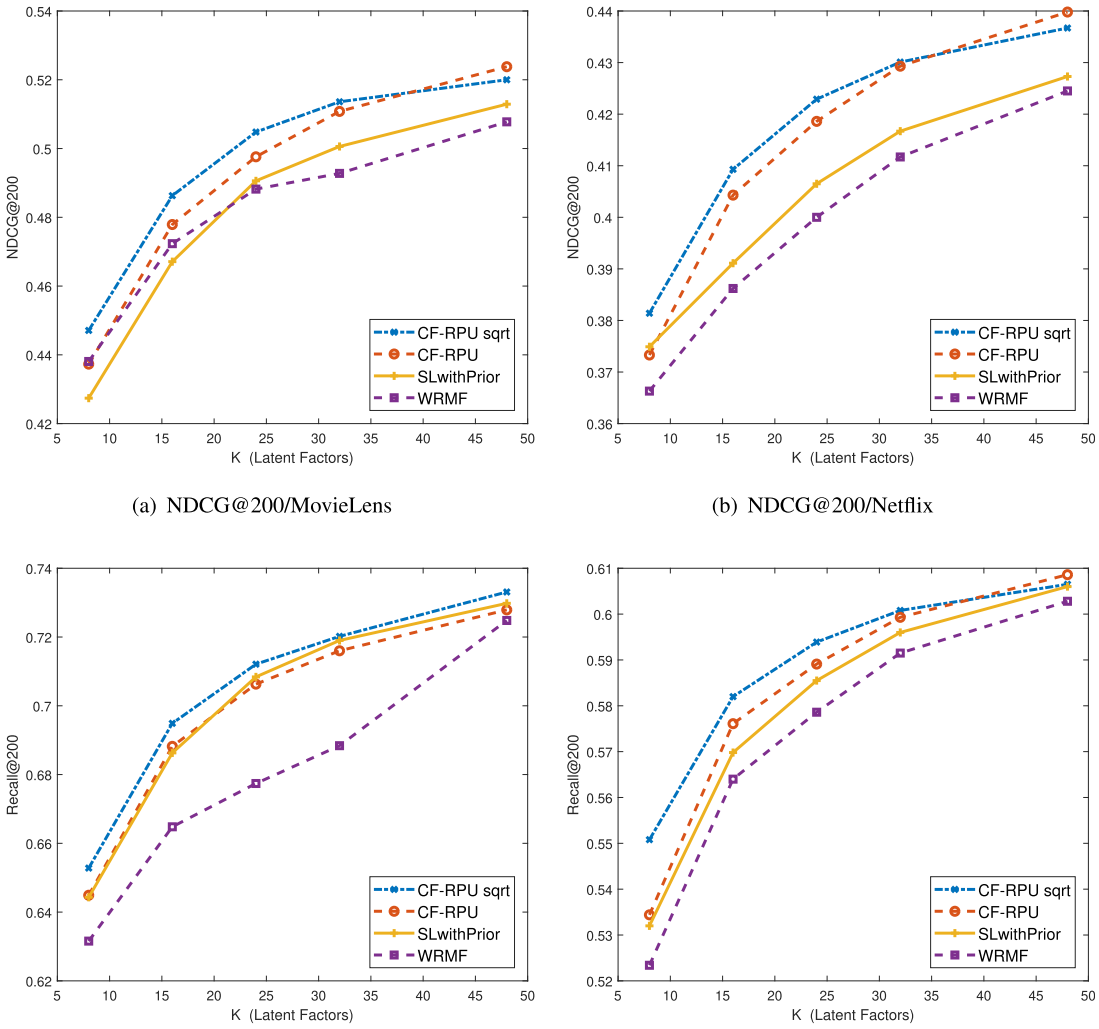
Overall Comparison Table 3 displays performances with respect to NDCG@5, NDCG@10, NDCG@20, and RECALL@20 on all four datasets. Experiments are run for five repetitions under the latent dimension $K = 32$. We have the following observations.

- The better performances of CF-RPU, CF-RPU sqrt, and SLwithPrior indicate that the utility of unknown data leads to significant improvements. CF-RPU has a relative 53% improvements on NDCG@5 on MovieLens dataset compared with RankALS which outperforms other algorithms without the help of unknown data. This is consistent with the *MNAR* assumption.

Table 3. Comparison with baselines. The dimension of the latent factor K is 32.

Dataset	Alg	NDCG@5	NDCG@10	NDCG@20	RECALL@20
MovieLens	CF-RPU sqrt	**0.4374 ± 0.0004**	**0.4128 ± 0.0002**	**0.4091 ± 0.0001**	**0.3206 ± 0.0004**
	CF-RPU	0.4319 ± 0.0005	0.4068 ± 0.0004	0.4025 ± 0.0003	0.3142 ± 0.0003
	SLwithPrior	0.4062 ± 0.0005	0.3844 ± 0.0004	0.3846 ± 0.0003	0.3076 ± 0.0001
	RankALS	0.2794 ± 0.0007	0.2671 ± 0.0008	0.2705 ± 0.0007	0.2048 ± 0.0006
	SQL-Rank	0.1940 ± 0.0018	0.1742 ± 0.0014	0.1623 ± 0.0012	0.1042 ± 0.0007
	MF-DR-JL	0.1433 ± 0.0008	0.1312 ± 0.0006	0.1281 ± 0.0003	0.0887 ± 0.0002
	POP	0.0856 ± 0.0005	0.0904 ± 0.0006	0.1046 ± 0.0003	0.1011 ± 0.0001
Yelp	CF-RPU sqrt	**0.0378 ± 0.0002**	**0.0442 ± 0.0003**	**0.0547 ± 0.0002**	**0.0829 ± 0.0004**
	CF-RPU	0.0365 ± 0.0005	0.0431 ± 0.0005	0.0538 ± 0.0005	0.0822 ± 0.0005
	SLwithPrior	0.0328 ± 0.0003	0.0395 ± 0.0004	0.0502 ± 0.0004	0.0785 ± 0.0005
	RankALS	0.0139 ± 0.0003	0.0171 ± 0.0003	0.0226 ± 0.0003	0.0365 ± 0.0004
	SQL-Rank	0.0037 ± 0.0002	0.0045 ± 0.0001	0.0059 ± 0.0001	0.0093 ± 0.0001
	MF-DR-JL	0.0008 ± 0.0004	0.0010 ± 0.0004	0.0015 ± 0.0004	0.0027 ± 0.0006
	POP	0.0065 ± 0.0002	0.0082 ± 0.0003	0.0108 ± 0.0002	0.0185 ± 0.0002
Amazon	CF-RPU sqrt	**0.0328 ± 0.0002**	**0.0391 ± 0.0002**	**0.0481 ± 0.0003**	**0.0738 ± 0.0004**
	CF-RPU	0.0317 ± 0.0002	0.0381 ± 0.0002	0.0473 ± 0.0002	0.0732 ± 0.0003
	SLwithPrior	0.0317 ± 0.0001	0.0373 ± 0.0002	0.0456 ± 0.0001	0.0687 ± 0.0002
	RankALS	0.0153 ± 0.0002	0.0193 ± 0.0002	0.0253 ± 0.0002	0.0418 ± 0.0003
	SQL-Rank	0.0032 ± 0.0001	0.0038 ± 0.0001	0.0048 ± 0.0001	0.0075 ± 0.0001
	MF-DR-JL	0.0001 ± 0.0000	0.0001 ± 0.0000	0.0002 ± 0.0000	0.0003 ± 0.0001
	POP	0.0044 ± 0.0001	0.0055 ± 0.0001	0.0070 ± 0.0009	0.0118 ± 0.0002
Netflix	CF-RPU sqrt	0.3929 ± 0.0004	0.3702 ± 0.0004	0.3572 ± 0.0003	**0.2074 ± 0.0002**
	CF-RPU	0.3908 ± 0.0003	0.3666 ± 0.0001	0.3526 ± 0.0009	0.2043 ± 0.0002
	SLwithPrior	**0.4118 ± 0.0003**	**0.3837 ± 0.0002**	**0.3659 ± 0.0002**	0.2068 ± 0.0002
	RankALS	0.2175 ± 0.0002	0.2090 ± 0.0002	0.2054 ± 0.0002	0.1045 ± 0.0001
	SQL-Rank	0.1945 ± 0.0007	0.1786 ± 0.0006	0.1646 ± 0.0004	0.0736 ± 0.0002
	MF-DR-JL	0.1407 ± 0.0001	0.1323 ± 0.0001	0.1273 ± 0.0001	0.0617 ± 0.0001
	POP	0.0702 ± 0.0001	0.0741 ± 0.0001	0.0814 ± 0.0008	0.0690 ± 0.0001

- Our approaches CF-RPU and CF-RPU sqrt outperform SLwithPrior algorithm on three datasets, indicating the proposed ranking-based prior on unknown ratings is more appropriate than the explicit prior in SLwithPrior. There are relative 4.2%, 5.6%, 7.4%, and 0.2% improvements in terms of RECALL@20 on the four datasets. SLwithPrior shows better performances on Netflix dataset. It is likely that the experiments on larger datasets, Netflix dataset, need to be run in more iterations for our methods.
- The proposed ranking-based framework for explicit feedback achieves superior performances compared with baseline algorithms, indicating the superiority of ranking-based



(a) NDCG@200/MovieLens (b) NDCG@200/Netflix

(c) RECALL@200/MovieLens (d) RECALL@200/Netflix

Figure 1. NDCG@200 and Recall@200 on MovieLens and Netflix dataset under the $K = 8, 16, 24, 32, 48$. (a) NDCG@200/MovieLens. (b) NDCG@200/Netflix. (c) RECALL@200/MovieLens. (d) RECALL@200/Netflix.

solutions. Compared with regression method MF-DR-JL and list-wise method SQL-Rank, our methods and RankALS, which apply ranking-based objective functions, improve a lot on four datasets in terms of NDCG and RECALL. From this phenomenon, it can be concluded that ranking-based solutions are suitable for item recommendation.

- When performing our tasks, CF-RPU sqrt outperforms CF-RPU, indicating that the normalization function benefits to solving imbalanced problems. Especially on MovieLens and Netflix datasets, the normalization strategy leads to 0.7% and 0.3% relative improvements. However, the Amazon dataset is much sparser than the three datasets and the performance with normalization strategy shows a little worse than that without normalization. A reason is that the sqrt normalization is not so good to weaken the influence of large quantities of unknown ratings.

Performances Versus Varying Dimensions

Algorithms are evaluated under different dimensions in MovieLens and Netflix datasets and we report the results in Figure 1. The dimensions vary in $\{8, 16, 24, 32, 48\}$.

In general, our proposed CF-RPU and CF-RPU sqrt algorithms perform better, indicating that our approaches are competing in spite of the increase of the dimension of the latent matrix.

Table 4. Comparison between with/without regularizer (K = 48).

DataSets	Regularizer	NDCG@200	Improvement	RECALL@200	Improvement
MovieLens	√	0.5223	24.71%	0.7274	6.61%
	×	0.4188		0.6823	
Yelp	√	0.1229	86.91%	0.3411	56.47%
	×	0.0695		0.2180	
Amazon	√	0.1063	46.21%	0.2888	32.17%
	×	0.0727		0.2185	
Netflix	√	0.4393	39.24%	0.6083	20.00%
	×	0.3155		0.5069	

Enlarging the dimensions, our approaches show consistent improvements over the SLwith-Prior and WRMF methods on MovieLens and Netflix datasets in terms of NDCG@200 and RECALL@200. Furthermore, there are more improvements on Netflix dataset which has a larger size, indicating that our methods have advantages over large datasets. Moreover, CF-RPU leads to more improvements on NDCG than RECALL, indicating that our method tends to improve the ranking quality.

Impact of the Ranking-Based Prior We conduct experiments to explore the influence of the ranking-based prior by setting $\lambda = 0$, which removes the compact of the unknown data. Empirical results are shown in Table 4.

The results reveal that the ranking-based prior on unknown ratings greatly improves the performance. Especially, there are significant improvements on NDCG@200 and RECALL@200 for Yelp (86.91% and 56.47%) and Amazon (46.21% and 32.17%) datasets which reflects that the superiority of the ranking-based prior for sparse datasets. It can be inferred that the unknown ratings have plenty of information and learning from the unknown ratings is helpful to limit the bias learned from rated items.

CONCLUSIONS

In this article, we propose a ranking-based prior on unknown data for ranking-based recommendation models. We avert the use of prior for unknown data and penalize the discrepancy of predicted ratings between any two unknown items. Then, we exploit it into a generic collaborative filtering framework for explicit feedback. Furthermore, we introduce strategies to deal with imbalance problems. For the specific application of framework, we design an efficient optimization algorithm with ALS method and then largely reduce time complexity. According to our experiments on the four ground-truth datasets, we can conclude that our proposed CF-RPU method has a better performance on ranking.

ACKNOWLEDGMENTS

The work was supported by grants from the National Natural Science Foundation of China (Grant 61976198, 61972069, 61832017, 61836007).

REFERENCES

1. B. M. Marlin and R. S. Zemel, "Collaborative prediction and ranking with non-random missing data," in *Proc. 3rd Ann. ACM Conf. Recommender Syst.*, 2009, pp. 5–12.
2. S. Wang *et al.*, "Intention nets: Psychology-inspired user choice behavior modeling for next-basket prediction," in *Proc. Assoc. Adv. Artif. Intell.*, 2020, pp. 1–8.
3. G. Takács and D. Tikk, "Alternating least squares for personalized ranking," in *Proc. 6th Ann. ACM Conf. Recommender Syst.*, 2012, pp. 83–90.
4. B. Marlin, R. S. Zemel, S. Roweis, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," in *Proc. 23rd Conf. Uncertainty Artif. Intell.*, 2007, pp. 267–275.

5. J. Chen, D. Lian, and K. Zheng, "Improving one-class collaborative filtering via ranking-based implicit regularizer," in *Proc. Assoc. Adv. Artif. Intell. Conf. Artif. Intell.*, 2019, vol. 33, pp. 37–44.
6. R. Devooght, N. Kourtellis, and A. Mantrach, "Dynamic matrix factorization with priors on unknown values," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 189–198.
7. S. Rendle, C. Freudenthaler, Z. Gantner, and L. S. Thiemeet, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intel. AUAI Press*, 2009, pp. 452–461.
8. S. Wang *et al.*, "Sequential recommender systems: challenges, progress and prospects," in *Proc. 28th Int. Joint Conf. Artif. Intell. AAAI Press*, 2019, pp. 6332–6338.
9. H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining.*, 2010, pp. 713–722.
10. D. Lian *et al.*, "Discrete content-aware matrix factorization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 325–334.
11. R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. Int. Conf. Mach. Learn. ACM*, 2008, pp. 880–887.
12. P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical Poisson factorization," in *Proc. Conf. Artif. Intell. UAI*, 2015, pp. 326–335.
13. X. Wang *et al.*, "Doubly robust joint learning for recommendation on data missing not at random," in *Proc. Int. Conf. Mach. Learn. ACM*, 2019, pp. 6638–6647.
14. L. Wu, C-J. Hsieh, and J. Sharpnack, "SQL-rank: A listwise approach to collaborative ranking," in *Proc. Int. Conf. Mach. Learn.. ACM*, 2018, pp. 5315–5324.
15. V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "One-class matrix completion with low-density factorizations," in *Proc. 10th IEEE Int. Conf. Data Mining*, 2010, pp. 1055–1060.

Jin Chen is currently working toward the M.S. degree with the University of Electronic Science and Technology of China (UESTC), Chengdu, China. She received the B.E. degree from the UESTC. Her main research interest includes recommendation system. Contact her at chenjin@std.uestc.edu.cn.

Defu Lian is currently the Research Professor with the School of Computer Science and Technology and the School of Data Science, University of Science and Technology of China, Langfang, China. His main research interests include data mining and recommender systems. He received the B.E. and Ph.D. degrees in computer science from the University of Science and Technology of China, in 2009 and 2014, respectively. He has authored or coauthored more than 60 papers in journals and conferences such as KDD, WWW, TKDE, and TOIS. He is the corresponding author of this article. Contact him at liandefu@ustc.edu.cn.

Kai Zheng is currently a Full Professor with the University of Electronic Science and Technology of China, Chengdu, China. His research interests include spatio-temporal database, trajectory data mining, spatial crowdsourcing, and recommendation systems. He received the Ph.D. degree in computer science from The University of Queensland, Brisbane, QLD, Australia. He has authored or coauthored more than 140 peer-reviewed papers in top-tier conferences and journals in the area of big data management. Contact him at zhengkai@uestc.edu.cn.