A Robust Bayesian Probabilistic Matrix Factorization Model for Collaborative Filtering Recommender Systems Based on User Anomaly Rating Behavior Detection

Hongtao Yu[1,2], Lijun Sun[1,2] and Fuzhi Zhang[1,2,*]
[1] School of Information Science and Engineering, Yanshan University
Qinhuangdao, China
[2] The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Yanshan University
Qinhuangdao, China
[e-mail: xjzfz@ysu.edu.cn]
*Corresponding author: Fuzhi Zhang

Received September 17, 2018; revised January 5, 2019; accepted March 6, 2019; published September 30, 2019

Abstract

Collaborative filtering recommender systems are vulnerable to shilling attacks in which malicious users may inject biased profiles to promote or demote a particular item being recommended. To tackle this problem, many robust collaborative recommendation methods have been presented. Unfortunately, the robustness of most methods is improved at the expense of prediction accuracy. In this paper, we construct a robust Bayesian probabilistic matrix factorization model for collaborative filtering recommender systems by incorporating the detection of user anomaly rating behaviors. We first detect the anomaly rating behaviors of users by the modified K-means algorithm and target item identification method to generate an indicator matrix of attack users. Then we incorporate the indicator matrix of attack users to construct a robust Bayesian probabilistic matrix factorization model and based on which a robust collaborative recommendation algorithm is devised. The experimental results on the MovieLens and Netflix datasets show that our model can significantly improve the robustness and recommendation accuracy compared with three baseline methods.

Keywords: Recommender systems, shilling attacks, anomaly rating behavior detection, robust Bayesian probabilistic matrix factorization

This work was supported by the National Natural Science Foundation of China (Nos. 61772452, 61379116), the Natural Science Foundation of Hebei Province, China (No. F2015203046).

1. Introduction

Nowadays, recommender systems have been applied to solve the information overload problem in many areas such as online product recommendations in e-commerce websites [1], Web-page recommendations in intelligent Web systems [2], POI (Point of Interest) recommendations in location-based social networks [3], and cloud service recommendations in cloud computing market [4]. Collaborative filtering (CF) [5] is a commonly-used technique in recommender systems, which has been widely used in e-commerce sites such as Amazon and eBay. CF methods are categorized as memory- and model-based methods [6]. Memory-based methods include user- and item-based approaches, which make recommendations based on similarity between users or items. Model-based methods first train a model using the known ratings of users, then exploit the model to predict ratings for unrated items.

Due to the open nature of CF-based systems, however, malicious users may bias the output of such systems by injecting fake profiles. This behavior has been known as shilling attacks or profile injection attacks [7], [8]. The fake profiles are called attack profiles or shilling profiles. Depending on the purpose of attacks, shilling attacks can be categorized into either push attacks (i.e., attacks that are designed to increase the probability of an item being recommended) or nuke attacks (i.e., attacks that are designed to decrease the probability of an item being recommended) [9]. The well-studied shilling attacks include random attack, average attack, and AoP (Average over Popular items) attack, etc [7], [8], [10]. These attacks present a challenge to the credibility of CF-based systems. Therefore, how to guarantee the credibility of CF-based systems has become a problem that cannot be ignored.

To reduce the impact of shilling attacks, many methods for detecting such attacks have been presented [11-16]. These detection methods mainly adopt binary classification to spot and filter shilling profiles, which are easy to filter out genuine profiles. An alternative way is to improve the robustness of recommendation algorithms. Robustness is the ability of a recommender system to make stable recommendations when its rating database is contaminated with noise data or malicious ratings [17], which has been investigated in the context of shilling attacks. Although a variety of robust CF algorithms have been presented, most of them improve robustness at the cost of decreasing accuracy. This is because the existing matrix factorization based robust recommendation methods need to discard outliers in parameter estimation, which may lead to the rejection of some genuine users' ratings, thus resulting in loss of accuracy.

To address these problems, we present a robust Bayesian probabilistic matrix factorization (BPMF) model for CF systems based on user anomaly rating behavior detection. Particularly, we first use clustering technique and target item identification method to detect the anomaly rating users, then we combine the detection results with BPMF model to construct a robust CF model and devise a robust CF algorithm to make recommendations.

The contributions of this paper are summarized below:

1) We use the clustering algorithm and target item identification method to detect user anomaly rating behaviors, and based on which an indicator matrix of attack users is generated.
2) We present a robust BPMF model by incorporating the indicator matrix of attack users and based on it a robust CF algorithm is devised.

3)We carry out experiments on different datasets and compare our model with other approaches.

The rest of paper is organized as follows. Section 2 briefly introduces the research on robust recommendation algorithms. Section 3 describes the proposed model in detail. Experimental results are presented in Section 4. The conclusion and future work are given in Section 5.

2. Related Work

Research on robust recommendation methods has been conducted over the past decade and has achieved considerable results. Mehta et al. [18] proposed a M-estimator based matrix factorization algorithm (MMF). MMF can find outliers by monitoring whether the residual is in a certain range. Nevertheless, the reported results show that MMF is only effective for small-scale attacks. Cheng and Hurley [19] proposed a least trimmed squares estimator based matrix factorization algorithm (LTSMF), which performed better than MMF. However, the accuracy of LTSMF algorithm is limited because some genuine users' ratings with the largest residuals are also discarded. Mehta and Nejdl [20] proposed a robust recommendation algorithm, i.e., VarSelect SVD. VarSelect SVD has been proved to be robust against shilling attacks, but it needs a prior knowledge of attack size. In [21], a robust recommendation method was proposed. This method first uses the relevance vector machine classifier to measure suspicious users, then mines the implicit trust between users according to their ratings, and incorporates the results of measurement to build a multidimensional trust model. By combining the trust model, neighbor model, and matrix factorization, a robust recommendation algorithm is finally developed. In [22], a robust CF method was proposed. It uses R1-norm to build a robust non-negative MF model, and based on it a robust CF algorithm is developed to make recommendations.

Probabilistic matrix factorization (PMF) [23] is a special MF, which is applicable to large and sparse datasets. Liu et al. [24] presented a new PMF model, which improved prediction accuracy by combining user relations with rating matrix. Nevertheless, the prediction of this model is easily affected by malicious ratings. In [25], a BPMF-based recommendation model was proposed. This model introduces the prior distribution over the hyper-parameters on the basis of PMF to avoid over-fitting. It proves that BPMF is better than PMF in accuracy. In [26], a robust recommendation model is proposed, which improved the prediction accuracy and robustness by using the long tail distribution or excluding attack users. Li et al. [27] presented a metadata-enhanced variational Bayesian MF model for robust recommendation. It fuses the BPMF model with metadata, which can weaken the effect of malicious users on the model's posterior, and thus guarantees the robustness of algorithm.

In this work, we aim to build a robust CF model with strong attack-resistant capability and high recommendation accuracy. Unlike the methods in [18,19,22] that use the robust estimators or R1-norm to limit the impact of ratings with the largest residuals on the recommendation models, which are easy to discard genuine users' ratings with maxinum residuals, our model only filters ratings on the target item based on the detection results of anomaly rating users. Different from the approach in [20], our model does not require a prior knowledge of attack size. Unlike the approach in [21], our model uses an unsupervised clustering algorithm and target item identification method to detect anomaly rating users, which does not need to train the classification model.

3. The Proposed Model

In this section, we first propose an approach for detecting anomaly rating users based on the modified K-means algorithm and target item identification method, then we combine the detection results with Bayesian probabilistic matrix factorization model to build a robust CF model and devise the corresponding robust CF algorithm which is called RBPMF-CF.

3.1 Detecting Anomaly Rating Behavior of Users

In the context of shilling attacks, the attack users usually give the highest rating for the item that they want to promote. This means the attack users generally have greater residual (i.e., the difference between a user's real and predicted ratings) than that of genuine users. To illustrate the characteristic of ratings for the attack users, we randomly choose 144 genuine users from the Movielens 100K dataset, and inject attack profiles generated by average attack, random attack, and AoP attack, respectively. These attacks are all push attacks, the filler size and attack size are set to 3%, respectively. Based on these profiles, the mean residuals of genuine and attack users are calculated, respectively. **Fig. 1** depicts the mean residuals of 228 users which include 144 genuine users, 28 AoP attack users, 28 average attack users, and 28 random attack users.

As shown in **Fig. 1**, the mean residuals of attackers are greater than those of genuine users, which means the ratings given by the attackers are generally greater than those of most genuine users. It can be seen from **Fig. 1**, some mean residuals of AoP attackers are close to those of most genuine users. This is because AoP attack is the obfuscated form of average attack. Unlike average attack profiles, AoP attack profiles use a certain percentage of popular items as filler items, which makes them look like genuine ones.

Due to the high similarity between attackers, we utilize a modified K-means algorithm to cluster anomaly rating users and further spot the attackers by the target item identification method.

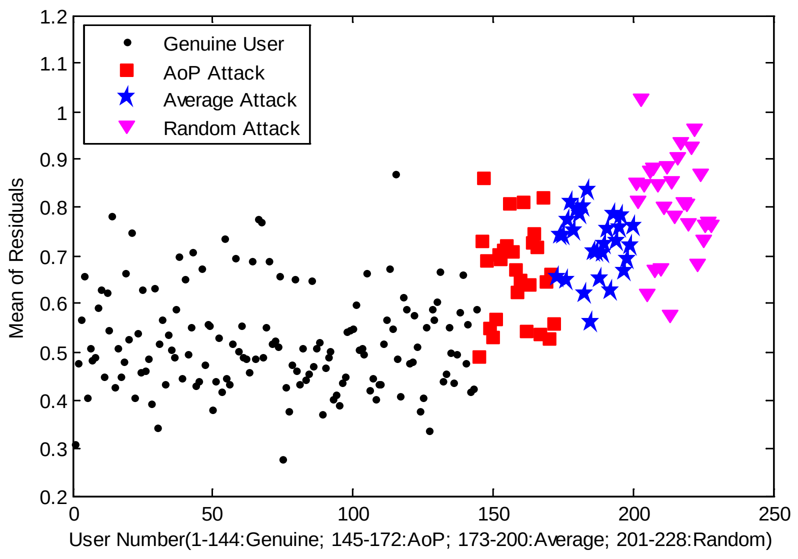


Fig. 1. The mean residuals of user ratings

3.1.1 Clustering Anomaly Rating Users

K-means is a conventional clustering algorithm, which splits a dataset samples into K clusters [28], [29]. This algorithm usually uses Euclidean distance as the metric to measure the similarity between two samples. The samples with high similarity are grouped into the same cluster, and the samples between clusters have low similarity. However, the K-means algorithm with Euclidean distance-based similarity measurement cannot well separate the attack users from genuine ones due to the high similarity between them. Hence, we present a similarity measurement metric to modify K-means algorithm in order to better group the attack users.

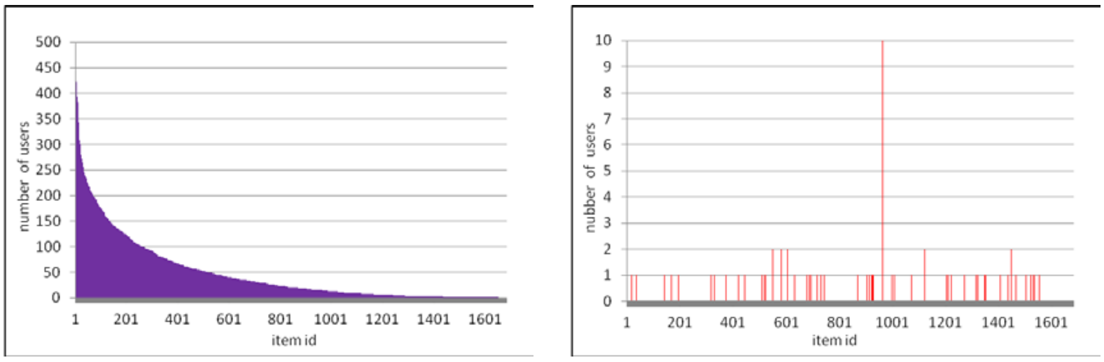
Definition 1 *popularity degree of item* (*PDI*). The popularity degree of item *i*, *PDI*(*i*) is denoted by

$$PDI(i) = \sum_{u \in \mathcal{K}_u} \Gamma(r_{ui}) \tag{1}$$

$$\Gamma(r_{ui}) = \begin{cases} 1, & r_{ui} \neq \varnothing \\ 0, & r_{ui} = \varnothing \end{cases} \tag{2}$$

where r_{ui} denotes the rating of user u for item i, $\mathcal{K}_u$ denotes the set of users, $r_{ui} \neq \varnothing$ represents that user u rates on item i, $r_{ui} = \varnothing$ represents that user u does not rate on item i.

Fig. 2 illustrates the difference of PDI between genuine and attack users. In Fig. 2, genuine users tend to rate the popular items, and the attack users rate all items with equal probability.



(a) The PDI of genuine users (b) The PDI of attack users

Fig. 2. The difference of PDI between genuine and attack users. Note that abscissa represents the sequence numbers of items after sorting on PDI in descending order

Definition 2 *average rating popularity degree of user* (*APDU*). The average rating popularity degree of user *u*, *APDU(u)*, is defined as below:

$$APDU(u) = \frac{\sum_{i \in I_u} PDI(i)}{|I_u|} \tag{3}$$

where I_u denotes the set of items rated by user u, $|I_u|$ is the number of items in set I_u.

Fig. 3 illustrates the average rating popularity degree of genuine and attack users. In Fig. 3, the APDU of attackers is less than that of the majority of genuine users. Therefore, it can be used as the basis of cluster center selection and similarity measurement.

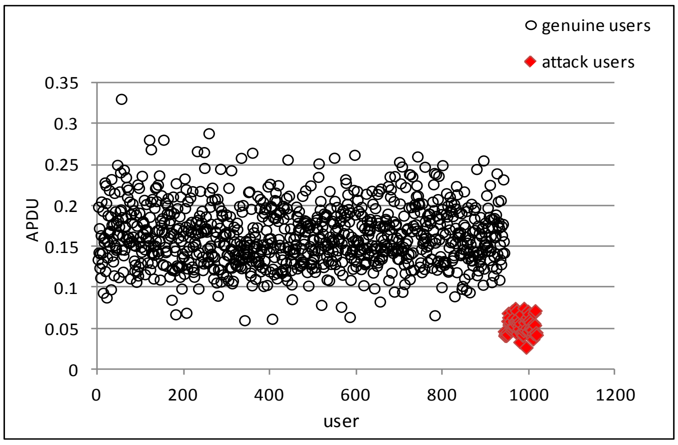


Fig. 3. APDU of users

Definition 3 *the distance between users* (*DIST*). The distance between users *u* and *v* is defined as follows:

$$DIST(u,v) = \left| APDU(u) - APDU(v) \right| \tag{4}$$

where *APDU(u)* and *APDU(v)* are average rating popularity degrees of users *u* and *v*, respectively. The greater the *DIST(u,v)*, the larger the difference between users *u* and *v*.

DIST(u,v) has two properties. One is symmetry, i.e., *DIST(u,v)=DIST(v,u)*. The other is non-negativity, i.e., *DIST(u, v)*≥0. Therefore, it can be used for measuring the difference between users.

To better group attack users, we utilize *DIST(u,v)* as similarity measurement metric of K-means algorithm and adjust the selection strategy of its cluster centers. The main steps for clustering anomaly rating users are as follows:

Step 1. Initialize the cluster centers k_1 and k_2 by calculating $\underset{u \in \mathcal{K}_u}{mean}\, APDU(u)$ and $\underset{u \in \mathcal{K}_u}{min}\, APDU(u)$ in user rating database.

Step 2. For any user $u \in \mathcal{K}_u$, compute the distance between *u* and cluster centers by Eq. (4), and assign *u* to the nearest cluster.

Step 3. Update the cluster centers k_1 and k_2 by calculating the mean and minimum value of *APDU* in each cluster.

Step 4. Repeat steps 2 and 3 until k_1 and k_2 are no longer change, and obtain the cluster of anomaly rating users.

According to the above steps, we design an algorithm to cluster anomaly rating users.

Algorithm 1 clustering anomaly rating users
Input: user rating matrix *R*
Output: suspicious users cluster C_s

```
1: $S_1 \leftarrow \varnothing$, $S_2 \leftarrow \varnothing$, $sum_1 \leftarrow 0$, $sum_2 \leftarrow 0$
2: for each item $i \in I$ do
3:   $PDI(i) = \sum_{u \in \mathcal{K}_u} \Gamma(r_{ui})$
4: end for
5: for each user $u \in \mathcal{K}_u$ do
6:   $APDU(u) = \sum_{i \in I_u} PDI(i) / \left| I_u \right|$
```

```
 7: end for
 8: $f_1 = mean(APDU)$, $f_2 = \min(APDU)$ /*$f_1$, $f_2$ denote the center of clusters $S_1$, $S_2$, respectively*/
 9: Initialize the center of clusters $S_1$, $S_2$ with $f_1$, $f_2$
10: repeat
11:   $k_1 \leftarrow f_1$, $k_2 \leftarrow f_2$
12:   for each user $u \in \kappa_u$ do
13:     compute the distance between $u$ and cluster center $k_1$ by $DIST_1 = |APDU(u) - k_1|$
14:     compute the distance between $u$ and cluster center $k_2$ by $DIST_2 = |APDU(u) - k_2|$
15:     if $DIST_1 < DIST_2$ then
16:       $S_1 \leftarrow S_1 \cup \{u\}$
17:     else
18:       $S_2 \leftarrow S_2 \cup \{u\}$
19:     end if
20:   end for
21:   for each user $u \in S_1$ do
22:     $sum_1 \leftarrow sum_1 + DIST_1$, $APDU_1(u) = \sum_{i \in I_u} PDI(i) / |I_u|$
23:   end for
24:   for each user $u \in S_2$ do
25:     $sum_2 \leftarrow sum_2 + DIST_2$, $APDU_2(u) = \sum_{i \in I_u} PDI(i) / |I_u|$
26:   end for
27:   if $\frac{sum_1}{|s_1|} \geq \frac{sum_2}{|s_2|}$ then
28:     $f_1 = mean(APDU_1)$, $f_2 = \min(APDU_2)$
29:   else
30:     $f_1 = \min(APDU_1)$, $f_2 = mean(APDU_2)$
31:   end if
32: until ($k_1 = f_1$ and $k_2 = f_2$)
33: if $\frac{sum_1}{|s_1|} \geq \frac{sum_2}{|s_2|}$ then
34:     $C_s \leftarrow S_2$
35: else
36:     $C_s \leftarrow S_1$
37: end if
38: return $C_s$
```

In Algorithm 1, Lines 1-32 are the modified K-means algorithm with new similarity measurement metric and new selection strategy of cluster centers. Lines 33-38 judge the cluster of anomaly rating users and return it.

The time complexity for algorithm 1 is analyzed below. The time complexity for Line 1, Lines 2-4, Lines 5-7, Lines 8-9, Lines 10-32, and Lines 33-38 is $O(1)$, $O(|I| \times |\kappa_u|)$,

$O(|I_{max}|\times|\kappa_u|)$, $O(|\kappa_u|)$, $repeat\_times\times[O(1)+O(|\kappa_u|)+O(|S_1|\times|I_{max}|)+O(|S_2|\times|I_{max}|)+O(|S_1|)+O(|S_2|)]$, and $O(1)$, respectively. Since $|S_1|<|\kappa_u|$, $|S_2|<|\kappa_u|$, $|I_{max}|<|I|$, and *repeat_times* is far less than $|\kappa_u|$, the time complexity for Lines 10-32 is at most $O(|\kappa_u|\times|I|)$. Thus, the time complexity of Algorithm 1 is $O(1)+O(|I|\times|\kappa_u|)+O(|I_{max}|\times|\kappa_u|)+O(|\kappa_u|)+O(|\kappa_u|\times|I|)+O(1)\approx O(|\kappa_u|\times|I|)$.

The space complexity analysis for Algorithm 1 is as follows. The space complexity for storing the input data (i.e., the user rating matrix *R*) and output data (i.e., the suspicious users cluster C_s) is at most $O(|\kappa_u|\times|I|)+O(|\kappa_u|)$. The space complexity for storing the array variables *PDI* and *APDU*, set variables *I* and κ_u, set variables S_1 and S_2, and other variables such as f_1, f_2, k_1, and k_2 is $O(|I|)+O(|\kappa_u|)$, $O(|I|)+O(|\kappa_u|)$, $O(|\kappa_u|)$, and $O(1)$, respectively. Thus, the space complexity of Algorithm 1 is $O(|\kappa_u|\times|I|)+O(|\kappa_u|)+2\times[O(|I|)+O(|\kappa_u|)]+O(|\kappa_u|)+O(1)\approx O(|\kappa_u|\times|I|)$.

3.1.2 Identifying The Attack Users

As the cluster of anomaly rating users obtained by Algorithm 1 may contain genuine users, we need to further identify the attack users in this cluster.

Firstly, we seek the attacked item from the user rating database, which can be done by calculating the *PDI* of items in the rating database. For the case of push attacks, the attacked item is usually selected from unpopular items. The set of unpopular items is defined as follows:

$$UI=\left\{ i \middle| i\in I, PDI(i)<\frac{\sum_{j\in I} PDI(j)}{|I|}\right\} \tag{5}$$

where *I* denotes the set of items.

Taking into account the ratings given by users in the cluster of anomaly rating users, we recalculate *PDI* of each item in the set of unpopular items, the attacked item is the item with the largest *PDI* in this set.

Secondly, we further identify attackers in the cluster of anomaly rating users according to the attacked item. If a user *u* in the cluster of anomaly rating users rates the attacked item, then the user *u* is viewed as an attack user and the flag of user *u* is set to 1.

According to the above analysis, we present an algorithm to further identify the attackers.

Algorithm 2 identifying the attackers
Input: user rating matrix *R*
Output: the indicator matrix of attackers *Z*

```
1: $UI \leftarrow \varnothing$, Sum_PDI $\leftarrow$ 0
2: for each item $i \in I$ do
3:   $PDI(i)=\sum_{u\in\kappa_u}\Gamma(r_{ui})$
4:   Sum_PDI $\leftarrow$ Sum_PDI+PDI(i)
5: end for
6: Avg_PDI $\leftarrow$ Sum_PDI / |I|
 7: for each item $i \in I$ do    /* obtain the set of unpopular items */
 8:   if (PDI(i)< Avg_PDI) then
 9:     $UI \leftarrow UI \cup \{i\}$
10:   endif
11: end for
12: $C_s \leftarrow$ call Algorithm 1   /* obtain the cluster of anomaly rating users */
13: $j \leftarrow getAttackedItem(UI,C_s)$   /* get the attacked item */
14: for each user $u \in \kappa_u$ do     /* obtain the indicator matrix of attackers */
15:   if $u \in C_s$ and $r_{uj}=r_{max}$ then /* $r_{max}$ is the maximum rating allowed */
16:     $Z(u)=1$
17:   else
18:     $Z(u)=0$
19:   end if
20: end for
21: return $Z$
```

In Algorithm 2, Lines 1-11 obtain the set of unpopular items by calculating *PDI* of items in the rating database according to Eq. (5). Lines 12-13 obtain the cluster of anomaly rating users by calling Algorithm 1 and get the attacked item by function *getAttackedItem*(*UI*,*C*$_s$). Lines 14-21 obtain the indicator matrix of attackers.

The time complexity for Algorithm 2 is analyzed below. The time complexity for Lines 1-11, Lines 12-13, and Lines 14-21 is $O(|I|\times|\kappa_u|)$, $O(|\kappa_u|\times|I|)$, and $O(|\kappa_u|)$, respectively. Thus, the time complexity for Algorithm 2 is $O(|\kappa_u|\times|I|)$.

The space complexity analysis for Algorithm 2 is as follows. The space complexity for storing the input data (i.e., the user rating matrix *R*) and output data (i.e., the indicator matrix of attackers *Z*) is $O(|\kappa_u|\times|I|)+O(|\kappa_u|)$. The space complexity for Lines 1-11, Lines 12-13, and Lines 14-21 is $3\times O(|I|)+O(|\kappa_u|)$, $O(|\kappa_u|\times|I|)$, and $O(|\kappa_u|)$, respectively. Therefore, The space complexity for Algorithm 2 is $O(|\kappa_u|\times|I|)$.

3.2 Robust BPMF Model for Recommendation

In MF model [30], the user rating matrix $R\in\Re^{m\times n}$ is decomposed into two low-rank matrices $U=(U_1,U_2,\ldots,U_m)\in\Re^{m\times d}$ and $V=(V_1,V_2,\ldots,V_n)\in\Re^{n\times d}$ which are called the user and item feature matrices, where U_i and V_j are *d*-dimensional user and item feature vectors, *m* and *n* are the number of users and items, *d* is the feature dimension. The expression is below:

$$R\approx U*V^{\mathrm{T}} \tag{6}$$

PMF is a special matrix factorization of analyzing low dimensional factorization from the view of statistics [23], which supposes the user ratings, item and user feature vectors obey Gaussian distribution. Compared with the traditional matrix factorization (such as SVD), PMF is easier to deal with big data and sparse data. Because it no longer looks for the optimal low rank, but rebuilds the model to train the user and item feature vectors from the angle of probability.

BPMF is a full Bayesian treatment of PMF model by integrating all model parameters and hyper-parameters to avoid tuning parameters [24], [25]. The conditional probability over the observed ratings is given by:

$$p(R|U,V,\sigma^2)=\prod_{i=1}^{m}\prod_{j=1}^{n}[N(R_{ij}|U_iV_j^{\mathrm{T}},\sigma^2)]^{I_{ij}} \tag{7}$$

where $N(x|\mu,\sigma^2)$ is the probability density function of Gaussian distribution with mean μ and variance σ^2, R_{ij} is the rating of user i for item j, I_{ij} is an indicator function that is 1 if user i rates on item j and 0, otherwise. The prior distributions over U and V are given by:

$$p(U|\mu_U,\Lambda_U)=\prod_{i=1}^{m}N(U_i|\mu_U,\Lambda_U^{-1}) \tag{8}$$

$$p(V|\mu_V,\Lambda_V)=\prod_{j=1}^{n}N(V_j|\mu_V,\Lambda_V^{-1}) \tag{9}$$

Bayesian probabilistic matrix factorization supposes the user hyper-parameters $\Theta_U=\{\mu_U,\Lambda_U\}$ and item hyper-parameters $\Theta_V=\{\mu_V,\Lambda_V\}$ obey the Gaussian-Wishart distribution. The prior distributions of Θ_U and Θ_V are given by:

$$p(\Theta_U|\Theta_0)=p(\mu_U|\Lambda_U)p(\Lambda_U)=N(\mu_U|\mu_0,(\beta_0\Lambda_U)^{-1})W(\Lambda_U|W_0,v_0) \tag{10}$$

$$p(\Theta_V|\Theta_0)=p(\mu_V|\Lambda_V)p(\Lambda_V)=N(\mu_V|\mu_0,(\beta_0\Lambda_V)^{-1})W(\Lambda_V|W_0,v_0) \tag{11}$$

where $W(x|W_0,v_0)$ is the Wishart distribution with v_0 degrees of freedom and a $d\times d$ scale matrix W_0, $\Theta_0=\{\mu_0,v_0,W_0\}$.

To develop attack-resistant CF recommendation algorithms, we construct a robust CF model by incorporating the detection results of user anomaly rating behaviors into the BPMF model. The reason for adopting BPMF model is that this model itself has better accuracy and robustness than other models. Particularly, we incorporate the indicator matrix of attackers *Z* obtained by Algorithm 2 into the prior distribution of item feature matrix *V* to decrease the negative impact of attackers. At the same time, we reserve the ratings of attackers on the unattacked items to alleviate the data sparsity, which is helpful to improve the recommendation accuracy.

Depending on whether or not the item is an attacked target item, the following two cases should be taken into account.

For the attacked item, we suppose the prior distribution of user ratings as follows:

$$p(R|U,V,\sigma^2)=\prod_{i=1}^{m}\prod_{j=1}^{n}[N(R_{ij}|U_iV_j^{\mathrm{T}},\sigma^2)]^{I_{ij}(1-Z_i)} \tag{12}$$

For the unattacked items, we use Eq. (7) as the prior distribution of user ratings. That is to say, we ignore the impact of attackers on the unattacked items.

Based on the constructed robust BPMF model, we design a robust CF algorithm, namely RBPMF-CF, which is described below:

Algorithm 3 RBPMF-CF
Input: user rating matrix *R*, the feature dimension *d*, the indicator matrix of attackers *Z*
Output: feature matrices *U*, *V*

```
1: Initialize the feature matrices U, V
2: for k=1 to loop do  /* loop is the number of iterations*/
```

```
3:   sample the hyper-parameters:
     $\Theta_U^k \sim p(\Theta_U \mid U^k, \Theta_0)$, $\Theta_V^k \sim p(\Theta_V \mid V^k, \Theta_0)$
4:   for each user $u \in \kappa_u$ do
5:     sample user feature: $U_u^{k+1} \sim p(U_u \mid R, V^k, \Theta_U^k)$
6:   end for
7:   for each item $i \in I$ do
8:     if item $i$ is an attacked item then
9:       sample item feature: $V_i^{k+1} \sim p(V_i \mid R, Z, U^{k+1}, \Theta_V^k)$
10:    else
11:      sample item feature: $V_i^{k+1} \sim p(V_i \mid R, U^{k+1}, \Theta_V^k)$
12:    end if
13:  end for
14: end for
15: return U, V
```

In Algorithm 3, Lines 1-6 initialize the feature matrices *U, V* and sample the user features. Lines 7-14 sample the item features. If the sampled item is an attacked item, then we exclude the ratings of attackers on the attacked item. Line 15 is to return the feature matrices *U, V*.

The time complexity for Algorithm 3 is analyzed below. The time complexity for Line 1, Lines 2-14, and Line 15 is $O(d \times |\kappa_u|) + O(d \times |I|)$, $loop \times [O(|\kappa_u|) + O(|I|)]$, and $O(1)$, respectively. Since *d* and *loop* are far less than $|\kappa_u|$ and $|I|$, the time complexity for Algorithm 3 is $O(|\kappa_u| + |I|)$.

The space complexity analysis for Algorithm 3 is as follows. The space complexity for storing the input data and output data is $O(|\kappa_u| \times |I|) + O(1) + O(|\kappa_u|)$ and $O(d \times |\kappa_u|) + O(d \times |I|)$, respectively. The space complexity for storing the set variables *I*, κ_u and other variables such as *k*, *u*, and *i* is $O(|I|) + O(|\kappa_u|)$ and $O(1)$, respectively. Therefore, the space complexity for Algorithm 3 is $O(|\kappa_u| \times |I|)$.

4. Experimental Evaluation

4.1 Experimental Data and Settings

We use the following datasets to evaluate our RBPMF-CF algorithm.

(1) MovieLens 100K dataset.[1] It includes 100000 ratings from 943 users on 1682 movies. The ratings are all integer values between 1 and 5, where 1 denotes disliked and 5 denotes the most liked. The sparsity level of this dataset is 93.7%. We randomly extract 80% ratings from this dataset as training set and the remaining 20% are used as test set.

(2) Netflix dataset.[2] It includes 103297638 ratings on 17770 movies by 480189 users and its sparsity level is 98.8%. All ratings are integer values between 1 and 5. We randomly extract 214690 ratings on 4000 movies by 2000 users as the sampled dataset whose sparsity level is

[1] http://grouplens.org/datasets/movielens/100k/
[2] It was constructed to support participants in the Netflix prize (http://netflixprize.com)

97.3%. The partition method of training and test sets for the sampled dataset is the same as that of MovieLens 100K dataset.

To evaluate the robustness of RBPMF-CF, the attack profiles generated by average attack, random attack, and AoP attack, respectively are injected into the training sets. We set the filler size to 3% and 5%, the attack size to 2%, 4% , 6%, 8%, and 10%, respectively. The target item is randomly chosen from unpopular items and all attack profiles are generated for push attacks.

In our experiments, we set the dimension of features *d* to 10 and the number of iterations *loop* to 50 for RBPMF-CF algorithm.

4.2 Evaluation Metrics

We use root mean squared error (RMSE) and prediction shift (PS) to measure the algorithm's performance.

RMSE is a metric to measure the algorithm's prediction accuracy and it is defined below [31]

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i)\in T}(r_{ui} - \hat{r}_{ui})^2}{|T|}} \tag{13}$$

where T is the test set, r_{ui} and $\hat{r}_{ui}$ are the real and predicted ratings of user u on item i, respectively.

PS is a metric to measure the algorithm's robustness, which is defined below [10]

$$\text{PS} = \frac{\sum_{u\in\kappa_u}\left|\hat{r}_{ui}^{'} - \hat{r}_{ui}\right|}{|\kappa_u|} \tag{14}$$

where $|\kappa_u|$ is the number of users in the test set, $\hat{r}_{ui}$ and $\hat{r}_{ui}^{'}$ are the predicted ratings of user u on item i before and after attacks, respectively.

4.3 Experimental Results and Analysis

To illustrate the superiority of RBPMF-CF, we compare it with three baseline algorithms.

(1) MMF: A robust MF algorithm based on M-estimator [18]. In our experiments, we set the feature dimension, the number of iterations, and the learning rate to 10, 25, and 0.01, respectively for MMF algorithm.

(2) LTSMF: A robust MF algorithm based on least trimmed square estimator [19]. In our experiments, we set the feature dimension, the number of iterations, and the learning rate to 10, 25, and 0.01, respectively for LTSMF algorithm.

(3) VarSelect SVD: A robust recommendation algorithm based on principal component analysis and singular value decomposition [20]. In our experiments, we set the feature dimension, the number of iterations, and the learning rate to 10, 100, and 0.01, respectively for VarSelect SVD algorithm.

4.3.1 Comparison of Performance on The MovieLens Dataset

Table 1 shows the comparison of performance for MMF, LTSMF, VarSelect SVD, and RBPMF-CF on the MovieLens dataset under various attacks at different attack sizes across different filler sizes.

As shown in **Table 1**, under three attacks, the RMSE of MMF and LTSMF is above 0.96, and the RMSE values of both algorithms fluctuate between 0.96 and 0.97 as the attack and filler sizes increase. On the whole, the accuracy of two algorithms is relatively close. The RMSE of VarSelect SVD is about 0.95. Nevertheless, the RMSE of VarSelect SVD under average and AoP attacks is almost above 0.95. Moreover, the majority RMSE values of VarSelect SVD under AoP attack are greater than those of it under average attack. The reason is that AoP attack profiles have very high similarity with genuine profiles so that parts of them are viewed as genuine profiles. By contrast, the RMSE of RBPMF-CF under three attacks is reduced obviously, which is below 0.92. This indicates that the combination of attack detection and BPMF model can further improve the accuracy of algorithm. Furthermore, the RMSE of RBPMF-CF under three attacks has little change at various attack sizes and filler sizes, which means RBPMF-CF is stable. Thus, RBPMF-CF has better accuracy than MMF, LTSMF, and VarSelect SVD on the MovieLens dataset.

Table 1. Comparison of performance on the MovieLens dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	MMF	RMSE	0.9635	0.9651	0.9606	0.9644	0.9609	0.9653	0.9631	0.9669	0.9638	0.9629
		PS	0.6051	0.8007	0.8543	0.8861	0.9249	0.6469	0.8108	0.896	0.95	0.9672
	LTSMF	RMSE	0.9626	0.9662	0.9639	0.9641	0.9616	0.9672	0.9646	0.9651	0.964	0.9609
		PS	0.6079	0.7667	0.8603	0.8873	0.9293	0.6539	0.7896	0.8786	0.9758	0.9857
	VarSelect SVD	RMSE	0.9553	0.952	0.9469	0.9455	0.9427	0.9488	0.9472	0.9449	0.9415	0.9414
		PS	0.3739	0.5832	0.6373	0.7085	0.7206	0.4327	0.6698	0.7268	0.7337	0.7362
	RBPMF-CF	RMSE	**0.9148**	**0.9134**	**0.9146**	**0.9145**	**0.914**	**0.9147**	**0.914**	**0.9147**	**0.9141**	**0.9148**
		PS	**0.3597**	**0.3247**	**0.3521**	**0.416**	**0.3966**	**0.3608**	**0.3413**	**0.3539**	**0.4103**	**0.4111**
Average attack	MMF	RMSE	0.9656	0.9612	0.9665	0.9624	0.9615	0.9627	0.9619	0.9661	0.9662	0.9609
		PS	0.8029	0.9493	1.039	1.1173	1.1569	0.8526	0.9679	1.0632	1.1635	1.1642
	LTSMF	RMSE	0.9606	0.9686	0.9692	0.9654	0.9635	0.9644	0.9663	0.9636	0.9628	0.9653
		PS	0.8586	1.0056	1.0896	1.1881	1.1924	0.8551	1.0637	1.1219	1.1961	1.2073
	VarSelect SVD	RMSE	0.9572	0.9556	0.9534	0.9562	0.9524	0.9527	0.9513	0.9494	0.9506	0.9511
		PS	0.5219	0.7246	0.8832	0.9711	1.0258	0.6181	0.8668	0.9426	1.117	1.1398
	RBPMF-CF	RMSE	**0.9147**	**0.9141**	**0.9146**	**0.9144**	**0.9151**	**0.9149**	**0.9144**	**0.9164**	**0.9158**	**0.9162**
		PS	**0.3574**	**0.3195**	**0.3685**	**0.4212**	**0.376**	**0.3594**	**0.3246**	**0.3547**	**0.4245**	**0.3903**
AoP attack	MMF	RMSE	0.9666	0.9608	0.9613	0.9618	0.9658	0.9632	0.9644	0.9694	0.9668	0.9606
		PS	0.8934	1.0207	1.134	1.1894	1.2278	0.9095	1.0658	1.1923	1.2262	1.2513
	LTSMF	RMSE	0.9665	0.9673	0.9637	0.965	0.9669	0.9635	0.9612	0.9694	0.9655	0.9654
		PS	0.8975	1.0321	1.1335	1.1754	1.2127	0.8992	1.1156	1.2297	1.2304	1.2808
	VarSelect SVD	RMSE	0.9571	0.958	0.9583	0.9558	0.9592	0.9547	0.9557	0.9546	0.9563	0.9585
		PS	0.6459	0.9614	0.9815	1.081	1.1431	0.7537	0.9811	1.0892	1.1942	1.2918
	RBPMF-CF	RMSE	**0.9147**	**0.9137**	**0.9146**	**0.9144**	**0.9149**	**0.915**	**0.9146**	**0.9155**	**0.915**	**0.915**
		PS	**0.3605**	**0.3149**	**0.3686**	**0.4165**	**0.3982**	**0.3584**	**0.3259**	**0.3414**	**0.4142**	**0.3726**

It can be seen from **Table 1**, for the same filler size and attack size, the PS of RBPMF-CF under three attacks is smaller than that of MMF, LTSMF, and VarSelect SVD, which means the robustness of RBPMF-CF is better than that of other algorithms. It can also be seen from **Table 1**, the PS of MMF, LTSMF and VarSelect SVD under AoP attack is slightly greater than that of them under average attack. The reason is that AoP attack profiles use a certain percentage of popular items as filler items, which makes them look like genuine ones. Due to the high similarity between them, some AoP attack profiles are regarded as genuine profiles by three algorithms, thus leading to a great prediction shift. However, the PS values of RBPMF-CF under AoP attack are basically consistent with those of it under average attack. This indicates that RBPMF-CF is still robust against AoP attack. Therefore, RBPMF-CF is more robust than MMF, LTSMF and VarSelect SVD on the MovieLens dataset.

4.3.2 Comparison of Performance on The Netflix Dataset

Table 2 shows the comparison of performance for MMF, LTSMF, VarSelect SVD, and RBPMF-CF on the Netflix dataset under various attacks at different attack sizes across different filler sizes.

Table 2. Comparison of performance on the Netflix dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	MMF	RMSE	0.9624	0.9598	0.9606	0.9583	0.9581	0.9589	0.961	0.9586	0.9566	0.9576
		PS	0.8324	1.0147	1.0056	1.1558	1.0543	0.7255	1.0043	1.1202	1.0015	1.0681
	LTSMF	RMSE	0.9747	0.9721	0.971	0.9719	0.9691	0.9669	0.9666	0.9674	0.975	0.9691
		PS	0.8468	0.8426	0.8831	0.9787	0.9738	0.6016	0.6564	0.7325	0.9077	0.98
	VarSelect SVD	RMSE	0.9518	0.947	0.9429	0.9393	0.9383	0.9426	0.9383	0.9361	0.9351	0.9342
		PS	0.5496	0.7037	0.6994	0.7786	0.8249	0.5183	0.7095	0.7818	0.8566	0.9184
	RBPMF-CF	RMSE	**0.9061**	**0.9058**	**0.906**	**0.9067**	**0.9068**	**0.9062**	**0.906**	**0.9068**	**0.9068**	**0.907**
		PS	**0.3469**	**0.3319**	**0.3767**	**0.361**	**0.321**	**0.3552**	**0.3069**	**0.385**	**0.3466**	**0.3249**
Average attack	MMF	RMSE	0.9641	0.9665	0.9655	0.9686	0.969	0.9649	0.9695	0.9682	0.9662	0.9711
		PS	0.9543	1.1305	1.1835	1.2224	1.2472	0.892	1.1962	1.1988	1.2233	1.2981
	LTSMF	RMSE	0.9792	0.9725	0.9783	0.9785	0.9753	0.9802	0.9758	0.976	0.9784	0.9774
		PS	0.9497	1.0668	1.1723	1.1943	1.1334	0.9167	1.0361	1.0768	1.1526	1.1896
	VarSelect SVD	RMSE	0.9563	0.954	0.9506	0.9507	0.9489	0.9497	0.9484	0.9469	0.944	0.9439
		PS	0.5215	0.7547	0.8437	0.9898	1.0009	0.6509	0.8412	0.9467	1.079	1.1076
	RBPMF-CF	RMSE	**0.9062**	**0.9069**	**0.9071**	**0.908**	**0.9085**	**0.9072**	**0.9073**	**0.9089**	**0.9088**	**0.9103**
		PS	**0.3616**	**0.3169**	**0.3463**	**0.344**	**0.2978**	**0.336**	**0.3024**	**0.3577**	**0.3345**	**0.2977**
AoP attack	MMF	RMSE	0.9717	0.9727	0.9724	0.9804	0.9765	0.9716	0.9746	0.9748	0.979	0.9765
		PS	1.0801	1.2333	1.3134	1.2751	1.2999	1.0634	1.23544	1.245	1.3004	1.2891
	LTSMF	RMSE	0.9807	0.9797	0.975	0.9813	0.9802	0.9799	0.9812	0.9858	0.9782	0.9861
		PS	1.0028	1.1415	1.2454	1.2366	1.2486	0.9497	1.0578	1.115	1.2164	1.2069
	VarSelect SVD	RMSE	0.9583	0.9595	0.9599	0.9613	0.9606	0.9552	0.957	0.956	0.957	0.9584
		PS	0.7333	0.919	1.0404	1.1559	1.1865	0.7825	1.0628	1.1252	1.2555	1.2965
	RBPMF-CF	RMSE	**0.9062**	**0.9063**	**0.9068**	**0.9074**	**0.9075**	**0.9063**	**0.9062**	**0.9071**	**0.9079**	**0.9087**
		PS	**0.3749**	**0.3249**	**0.3793**	**0.351**	**0.3107**	**0.3557**	**0.33**	**0.3621**	**0.3368**	**0.293**

As shown in **Table 2**, under random attack, the RMSE of RBPMF-CF is below 0.91, which is better than that MMF, LTSMF, and VarSelect SVD. Moreover, the change of its RMSE is relatively stable. This means that the increase of attack and filler sizes has little impact on the accuracy of RBPMF-CF. Under average attack, the RMSE of MMF and LTSMF is close to 0.97 and 0.98, respectively, which is slightly greater than that of them on the MovieLens dataset. This means the accuracy of two algorithms is affected to some extent by the sparsity of Netflix dataset. The RMSE of VarSelect SVD under average attack is about 0.95, which is better than that of MMF and LTSMF. The RMSE of RBPMF-CF under average attack is about 0.91, which is slightly smaller than that of it on the MovieLens dataset. This indicates that the sparsity of Netflix dataset has little impact on the accuracy of RBPMF-CF. For the case of AoP attack, the RMSE of MMF and LTSMF is above 0.97, the RMSE of VarSelect SVD is close to 0.96, which is slightly greater than that of them under average attack. This is because parts of AoP attack profiles are viewed as genuine ones due to their high similarity, resulting in a decline in accuracy for three algorithms. The RMSE of RBPMF-CF is about 0.91, which is better than that of baselines. Thus, RBPMF-CF also has better accuracy than MMF, LTSMF, and VarSelect SVD on the Netflix dataset.

It can be seen from **Table 2**, the PS values of RBPMF-CF under three attacks are smaller than those of MMF, LTSMF, and VarSelect SVD, which indicates that RBPMF-CF is more robust against three attacks than the baseline algorithms. For the case of AoP attack, the PS values of MMF, LTSMF, and VarSelect SVD are slightly greater than those of them under average attack. This is because parts of AoP attack profiles are regarded as genuine profiles by three algorithms. The PS values of RBPMF-CF under AoP attack are basically consistent with the results under average attack, which means RBPMF-CF is also robust against AoP attack. Therefore, the robustness of RBPMF-CF is also better than that of MMF, LTSMF, and VarSelect SVD under three attacks on the Netflix dataset.

4.3.3 The Effectiveness of Our Anomaly Rating Behavior Detection Method

To show the effectiveness of our anomaly rating behavior detection method (i.e., the first step of the proposed model, which is called ARBD for short), we conduct further experiments on two datasets from three aspects. Firstly, we compare ARBD with two existing shilling attack detection methods in the literature of recommender systems using precision and recall metrics. Secondly, we substitute ARBD with an existing detection method and perform the proposed Bayesian probabilistic matrix factorization model after removing the detected anomaly ratings. Thirdly, we combine ARBD with the basic matrix factorization (MF) model, which is denoted as ARBD+MF for convenience, and compare it with the basic MF.

(1) *Comparison of detection performance*. To show the performance of ARBD in detecting anomaly rating users, the precision and recall metrics are used for evaluating its performance, which are defined below:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

where TP and FN are the number of attack profiles correctly identified and misclassified, respectively, FP is the number of genuine profiles misclassified.

To show the superiority of ARBD, we compare it with two baselines.

a) CBS (Catch the Black Sheep) [14]: An unsupervised approach for detecting shilling attacks, which needs labeling seed attack users. In the experiment, 20% attack users for each attack size are used as the seeds.
b) UD-HMM [16]: An unsupervised approach for detecting shilling attacks based on hidden Markov model and hierarchical clustering. In the experiment, the parameters N and α are set to 5 and 0.7, respectively.

Tables 3 and **4** list the precision and recall of three methods with various attacks on the MovieLens and Netflix datasets, respectively.

Table 3. Precision and recall of three methods on the Movielens dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	CBS	Precision	0.6842	0.8205	0.9000	0.8765	0.8932	0.6842	0.8974	0.9000	0.9259	0.9126
		Recall	0.6842	0.8421	0.9474	0.9467	0.9787	0.6842	0.9211	0.9474	1.0000	1.0000
	UD-HMM	Precision	0.6667	0.9744	0.7917	0.9615	1	0.6786	0.8444	0.9333	0.9737	0.9691
		Recall	0.9649	1	1	1	0.9929	1	1	0.9942	0.9867	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1
Average attack	CBS	Precision	0.5263	0.8205	0.9000	0.9136	0.8932	0.7895	0.8974	0.9000	0.9259	0.9126
		Recall	0.5263	0.8421	0.9474	0.9867	0.9787	0.7895	0.9211	0.9474	1.0000	1.0000
	UD-HMM	Precision	0.6781	1	0.9828	0.9868	0.9792	0.7261	0.8261	0.9661	0.9868	0.9895
		Recall	0.9649	1	1	1	1	1	1	1	1	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1
AoP attack	CBS	Precision	0.6842	0.8205	0.9167	0.8765	0.9029	0.6667	0.8421	0.8667	0.9259	0.8835
		Recall	0.6842	0.8421	0.9649	0.9467	0.9894	0.6842	0.8421	0.9123	1.0000	0.9681
	UD-HMM	Precision	0.1597	0.6491	0.5729	0.8242	0.7899	0.1027	0.3396	0.5534	0.4054	0.4253
		Recall	1	0.9912	0.9883	1	0.9929	1	0.9474	1	1	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1

Table 4. Precision and recall of three methods on the Netflix dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	CBS	Precision	0.4000	0.7470	0.8189	0.8605	0.8955	0.5250	0.7229	0.8976	0.8895	0.9000
		Recall	0.4000	0.7750	0.8667	0.9250	0.9850	0.5250	0.7500	0.9500	0.9563	0.9900
	UD-HMM	Precision	0.5634	0.3791	0.8451	0.9877	0.9901	0.3500	0.4598	0.9375	0.9938	0.9217
		Recall	1	1	1	1	1	0.9583	1	1	1	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1
Average attack	CBS	Precision	0.4500	0.6988	0.8346	0.8547	0.8818	0.6000	0.7590	0.8425	0.8721	0.9045
		Recall	0.4500	0.7250	0.8833	0.9188	0.9700	0.6000	0.7875	0.8917	0.9375	0.9950
	UD-HMM	Precision	0.1498	0.8989	0.9677	0.9877	0.9901	0.7843	0.8602	0.7362	0.9938	0.9852
		Recall	1	1	1	1	1	1	1	1	1	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1
AoP attack	CBS	Precision	0.5250	0.6988	0.8583	0.8721	0.8864	0.4000	0.8193	0.8189	0.8663	0.9045
		Recall	0.5250	0.7250	0.9083	0.9375	0.9750	0.4000	0.8500	0.8667	0.9313	0.9950
	UD-HMM	Precision	0.0756	0.4545	0.5941	0.8939	0.9132	0.0659	0.4145	0.3750	0.8602	0.9009
		Recall	1	1	1	1	1	1	1	1	1	1
	ARBD	Precision	1	1	1	1	1	1	1	1	1	1
		Recall	1	1	1	1	1	1	1	1	1	1

As shown in **Tables 3** and **4**, CBS maintains high precision and recall in detecting attacks with large attack sizes, which indicates that CBS can correctly detect most attack profiles. However, it performs poorly when detecting attacks with small attack sizes. The reason is that the number of seed attack users for CBS is small at low attack sizes. As to UD-HMM, it can effectively detect random and average attacks with large attack sizes, but it performs poorly in detecting the two attacks with small attack sizes. When detecting AoP attack, the overall performance of UD-HMM is not very good because a number of genuine profiles are misidentified. All the precision and recall values of ARBD are 1, which indicates that ARBD can correctly detect attack profiles and none of them are misidentified. These results illustrate the effectiveness of ARBD in detecting attacks. Therefore, ARBD outperforms CBS and UD-HMM in detecting three attacks.

(2) *Comparison of RMSE and PS by substituting ARBD with UD-HMM.* To show the effectiveness of ARBD, we substitute ARBD with UD-HMM and combine UD-HMM with the proposed Bayesian probabilistic matrix factorization model (denoted as UD-HMM+BPMF for convenience) to make recommendations. **Tables 5** and **6** list the RMSE and PS of UD-HMM+BPMF and RBPMF-CF on the MovieLens and Netflix datasets with various attacks, respectively.

As shown in **Table 5**, the RMSE of UD-HMM+BPMF under three attacks is between 0.9134 and 0.9164, the RMSE of RBPMF-CF under three attacks is between 0.9134 and 0.9156. These results indicate that there is little difference between UD-HMM+BPMF and RBPMF-CF in prediction accuracy on the MovieLens dataset. The PS of UD-HMM+BPMF under three attacks is between 0.3019 and 0.4746, the PS of RBPMF-CF under three attacks is between 0.3163 and 0.4216. These results illustrate that there is no big difference between UD-HMM+BPMF and RBPMF-CF in prediction shift on the MovieLens dataset.

Table 5. Comparison of RMSE and PS for UD-HMM+BPMF and RBPMF-CF on the MovieLens dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	UD-HMM+BPMF	RMSE	0.9148	0.9134	0.9147	0.9145	0.9140	0.9147	0.9140	0.9148	0.9142	0.9148
		PS	0.3539	0.3019	0.3894	0.3923	0.4370	0.3522	0.3321	0.4134	0.3872	0.4318
	RBPMF-CF	RMSE	0.9148	0.9134	0.9146	0.9145	0.9140	0.9147	0.9140	0.9147	0.9141	0.9148
		PS	0.3613	0.3276	0.3441	0.4145	0.3941	0.3637	0.3442	0.3458	0.4092	0.4084
Average attack	UD-HMM+BPMF	RMSE	0.9147	0.9141	0.9146	0.9144	0.9151	0.9149	0.9144	0.9164	0.9159	0.9162
		PS	0.3380	0.3496	0.3985	0.4091	0.4013	0.3578	0.3229	0.3887	0.4021	0.4097
	RBPMF-CF	RMSE	0.9148	0.9145	0.9147	0.9146	0.9150	0.9149	0.9146	0.9156	0.9153	0.9155
		PS	0.3592	0.3225	0.3591	0.4194	0.3724	0.3619	0.3291	0.3466	0.4216	0.3874
AoP attack	UD-HMM+BPMF	RMSE	0.9147	0.9137	0.9147	0.9144	0.9149	0.9150	0.9146	0.9156	0.9151	0.9150
		PS	0.3661	0.3127	0.4035	0.3929	0.4233	0.3518	0.3297	0.4135	0.4746	0.4665
	RBPMF-CF	RMSE	0.9148	0.9143	0.9147	0.9146	0.9149	0.9149	0.9147	0.9152	0.9149	0.9149
		PS	0.3633	0.3163	0.3595	0.4148	0.3953	0.3615	0.3271	0.3339	0.4124	0.3695

Table 6. Comparison of RMSE and PS for UD-HMM+BPMF and RBPMF-CF on the Netflix dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	UD-HMM+BPMF	RMSE	0.8896	0.8890	0.8896	0.8897	0.8903	0.8900	0.8895	0.8901	0.8900	0.8911
		PS	0.3541	0.4117	0.3130	0.4380	0.3698	0.3608	0.4542	0.3183	0.4240	0.5392
	RBPMF-CF	RMSE	0.8896	0.8889	0.8896	0.8897	0.8903	0.8900	0.8895	0.8901	0.8900	0.8911
		PS	0.3603	0.4382	0.3139	0.4380	0.3698	0.3560	0.4542	0.3183	0.4145	0.3834
Average attack	UD-HMM+BPMF	RMSE	0.8887	0.8871	0.8869	0.8862	0.8867	0.8882	0.8862	0.8862	0.8854	0.8861
		PS	0.3469	0.3972	0.3320	0.3870	0.3875	0.3539	0.4647	0.3158	0.3646	0.3679
	RBPMF-CF	RMSE	0.8880	0.8889	0.8876	0.8911	0.8878	0.8880	0.8882	0.8882	0.8880	0.8889
		PS	0.3446	0.4380	0.3235	0.3870	0.3875	0.3679	0.4572	0.3161	0.3646	0.3679
AoP attack	UD-HMM+BPMF	RMSE	0.8889	0.8881	0.8882	0.8882	0.8880	0.8890	0.8876	0.8881	0.8880	0.8888
		PS	0.8339	1.0018	0.4178	0.4602	0.3535	0.9803	0.579	0.4264	0.3665	0.5681
	RBPMF-CF	RMSE	0.8889	0.8880	0.8882	0.8882	0.8880	0.8889	0.8876	0.8911	0.8878	0.8911
		PS	0.3584	0.4471	0.3215	0.4128	0.3703	0.3380	0.4364	0.3139	0.3609	0.3904

It can be seen from **Table 6**, under three attacks, the RMSE of UD-HMM+BPMF is between 0.8854 and 0.8911, the RMSE of RBPMF-CF is between 0.8876 and 0.8911. These results illustrate that the prediction accuracy of UD-HMM+BPMF on the Netflix dataset is almost the same with that of RBPMF-CF. The PS of UD-HMM+BPMF under random and average attacks is between 0.3130 and 0.5392, the PS of RBPMF-CF under random and average attacks is between 0.3139 and 0.4572. These results illustrate that there is no obvious difference between UD-HMM+BPMF and RBPMF-CF in prediction shift under random and average attacks on the Netflix dataset. As to AoP attack, the PS of UD-HMM+BPMF is between 0.3553 and 1.0018, which is larger than that of it under random and average attacks. This is because UD-HMM performs poorly in detecting AoP attack. The PS of RBPMF-CF under AoP attack is between 0.3139 and 0.4471, which has little difference compared with that of it under random and average attacks. This again illustrates the superiority of ARBD in detecting anomaly rating users.

(3) *Comparison of RMSE and PS for the basic MF and ARBD+MF.* To further show the effectiveness of ARBD, we conduct experiments to compare RMSE and PS of the basic MF and ARBD+MF on the MovieLens and Netflix datasets with various attacks. **Tables 7** and **8** list the RMSE and PS of the basic MF and ARBD+MF on the MovieLens and Netflix datasets, respectively.

As shown in **Table 7**, under three attacks, the RMSE of basic MF is between 0.9701 and 0.9760, the RMSE of ARBD+MF is between 0.9725 and 0.9764. Clearly, there is almost no difference between the basic MF and ARBD+MF in prediction accuracy. As to the prediction shift metric, the PS of basic MF under three attacks is between 0.8009 and 1.4840, the PS of ARBD+MF under three attacks is between 0.0824 and 0.1973. Clearly, ARBD+MF is more robust against attacks than the basic MF on the MovieLens dataset.

As shown in **Table 8**, under three attacks, the RMSE of basic MF is between 0.9533 and 0.9618, the RMSE of ARBD+MF is between 0.9621 and 0.9653, which has little difference between them in prediction accuracy. Under three attacks, the PS of basic MF is between 1.1482 and 1.7988, the PS of ARBD+MF is between 0.0715 and 0.1661. Clearly, the robustness of ARBD+MF on the Netflix dataset is much better than that of basic MF.

Therefore, the combination of ARBD with the basic MF model (i.e., ARBD+MF) can significantly improve the robustness of the basic MF. This again illustrates the effectiveness of ARBD in detecting anomaly rating users.

Table 7. RMSE and PS of the basic MF and ARBD+MF on the MovieLens dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	basic MF	RMSE	0.9729	0.9715	0.9721	0.9701	0.9712	0.9702	0.9721	0.9724	0.9709	0.9720
		PS	0.8009	0.8909	1.0151	0.9595	0.9940	0.7799	0.8791	0.9669	0.9783	1.0409
	ARBD+MF	RMSE	0.9747	0.9725	0.9746	0.9744	0.9754	0.9746	0.9749	0.9729	0.9738	0.9752
		PS	0.1124	0.1630	0.1382	0.0864	0.1032	0.1011	0.1098	0.1002	0.1631	0.1350
Average attack	basic MF	RMSE	0.9743	0.9721	0.9717	0.9707	0.9718	0.9732	0.9730	0.9733	0.9734	0.9714
		PS	0.9774	1.1246	1.2018	1.2127	1.3053	0.9400	1.1299	1.1884	1.2436	1.2990
	ARBD+MF	RMSE	0.9732	0.9741	0.9745	0.9748	0.9735	0.9741	0.9733	0.9749	0.9743	0.9733
		PS	0.0976	0.1973	0.1560	0.2042	0.0824	0.1342	0.1717	0.1111	0.1556	0.1474
AoP attack	basic MF	RMSE	0.9744	0.9740	0.9760	0.9731	0.9762	0.9738	0.9757	0.9746	0.9738	0.9740
		PS	1.0230	1.3028	1.3445	1.4724	1.4840	1.0997	1.3082	1.3945	1.4477	1.4566
	ARBD+MF	RMSE	0.9744	0.9756	0.9744	0.9744	0.9737	0.9740	0.9760	0.9734	0.9764	0.9748
		PS	0.1255	0.1022	0.1142	0.0891	0.1522	0.1733	0.0887	0.1385	0.1605	0.1899

Table 8. RMSE and PS of the basic MF and ARBD+MF on the Netflix dataset

	Filler size		3%					5%				
	Attack size		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
Random attack	basic MF	RMSE	0.9607	0.9560	0.9544	0.9539	0.9547	0.9564	0.9557	0.9540	0.9533	0.9533
		PS	1.1482	1.2996	1.3733	1.4594	1.4614	1.1891	1.3353	1.4480	1.4745	1.5095
	ARBD+MF	RMSE	0.9635	0.9636	0.9639	0.9653	0.9632	0.9621	0.9641	0.9633	0.9641	0.9641
		PS	0.1428	0.1155	0.1089	0.0890	0.1167	0.1154	0.1228	0.1280	0.1140	0.0944
Average attack	basic MF	RMSE	0.9607	0.9560	0.9544	0.9539	0.9547	0.9564	0.9557	0.9540	0.9533	0.9533
		PS	1.1482	1.2996	1.3733	1.4594	1.4614	1.1891	1.3353	1.4480	1.4745	1.5095
	ARBD+MF	RMSE	0.9635	0.9636	0.9639	0.9653	0.9632	0.9621	0.9641	0.9633	0.9641	0.9641
		PS	0.1428	0.1155	0.1089	0.0890	0.1167	0.1154	0.1228	0.1280	0.1140	0.0944
AoP attack	basic MF	RMSE	0.9639	0.9608	0.9618	0.9591	0.9602	0.9601	0.9597	0.9597	0.9596	0.9580
		PS	1.4445	1.6064	1.7041	1.7291	1.7536	1.4774	1.6193	1.7026	1.7644	1.7988
	ARBD+MF	RMSE	0.9632	0.9635	0.9643	0.9645	0.9625	0.9627	0.9640	0.9625	0.9651	0.9635
		PS	0.1130	0.0961	0.1231	0.0715	0.0887	0.1273	0.1661	0.1074	0.0909	0.1200

4.3.4 Comparison of Actual Runtime for Four Algorithms

The actual runtime for a model-based recommendation algorithm consists of the training time and predictive time, which refers to the time required to train a predictive model on the training set and the time required to perform rating prediction for the target item on the test set, respectively. For RBPMF-CF algorithm, the training time also includes the time for detecting anomaly rating users. To compare the runtime of four algorithms, we carry out experiments on two datasets and calculate their training time and predictive time, respectively. **Tables 9** and **10** list the training and predictive time for four algorithms, respectively.

Table 9. The training time for four algorithms (s)

Algorithms	Datasets	
	Movielens	Netflix
MMF	9.25	438.72
LTSMF	10.28	502.95
VarSelect SVD	106.92	2518.22
RBPMF-CF	10.99	156.32

Table 10. The predictive time for four algorithms (μs)

Algorithms	Datasets	
	Movielens	Netflix
MMF	4.15	4.92
LTSMF	5.26	6.05
VarSelect SVD	16.61	42.01
RBPMF-CF	3.78	2.77

As listed in **Table 9**, the training time of VarSelect SVD on the MovieLens dataset is the largest, the training time of MMF, LTSMF and RBPMF-CF on the MovieLens 100K dataset has no obvious difference. As to Netflix dataset, the training time of VarSelect SVD is still the largest, LTSMF comes the second, MMF ranks the third, and the training time of RBPMF-CF is the smallest. Therefore, RBPMF-CF has obvious advantage in training time on the Netflix dataset.

In **Table 10**, the predictive time of four algorithms on two datasets is all in microseconds, which means four algorithms can make a predictive rating for a target item very quickly. Therefore, there is no big difference between them in predictive time while the predictive time of RBPMF-CF is the smallest and the predictive time of VarSelect SVD is the largest.

5. Conclusion

In this paper, we present a robust BPMF model for CF recommender systems based on detecting anomaly rating users. To reduce the impact of shilling attacks on recommendation results, we present a modified K-means algorithm to cluster anomaly rating users and based on it we further identify and mark the attack users. By combining the detection results with BPMF model, we construct a robust CF model and design a robust recommendation algorithm. Compared with the baseline algorithms, our algorithm is more accurate and robust.

In our future work, we will incorporate the item attribute information into BPMF model to further improve the recommendation accuracy of our algorithm. In addition, we will explore more effective ways to detect the attackers to further improve the robustness of our algorithm.

References

[1] Z. Lin, "An empirical investigation of user and system recommendations in e-commerce," *Decision Support Systems*, vol. 68, pp. 111-124, December, 2014. Article (CrossRef Link)

[2] T.T.S. Nguyen, H. Lu and J. Lu, "Web-page recommendation based on web usage and domain knowledge," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2574-2587, October, 2014. Article (CrossRef Link)

[3] H. Yin, X. Zhou, B. Cui, and et al., "Adapting to user interest drift for POI recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2566-2581, October, 2016. Article (CrossRef Link)

[4] H. Mezni and T. Abdeljaoued, "A cloud services recommendation system based on Fuzzy Formal Concept Analysis," *Data & Knowledge Engineering*, vol. 116, pp. 100-123, July, 2018. Article (CrossRef Link)

[5] G. Guo, J. Zhang and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1607-1620, July, 2016. Article (CrossRef Link)

[6] Y. Shi, M. Larson and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges," *ACM Computing Surveys*, vol. 47, no. 1, pp.1-45, July,

2014. Article (CrossRef Link)

[7] I. Gunes, C. Kaleli, A. Bilge, and et al., “Shilling attacks against recommender systems: a comprehensive survey,” *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767-799, April, 2014. Article (CrossRef Link)

[8] B. Mobasher, R. Burke, R. Bhaumik and C. Williams, “Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness,” *ACM Transactions on Internet Technology*, vol. 7, no. 4, pp. 1-23, October, 2007. Article (CrossRef Link)

[9] R. Burke, M.P. O’Mahony, and N.J. Hurley, “Robust collaborative recommendation,” in Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.(Ed): *Recommender Systems Handbook*, Springer, pp. 805-835, 2011.

[10] N. Hurley, Z. Cheng and M. Zhang, “Statistical attack detection,” in *Proc. of the 3rd ACM Conference on Recommender systems*, pp. 149-156, October 23-25, 2009. Article (CrossRef Link)

[11] C.A. Williams, B. Mobasher and R. Burke, “Defending recommender systems: detection of profile injection attacks,” *Service Oriented Computing and Applications*, vol. 1, no. 3, pp. 157-170, November, 2007. Article (CrossRef Link)

[12] B. Mehta and W. Nejdl, “Unsupervised strategies for shilling detection and robust collaborative filtering,” *User Modeling and User Adapted Interaction*, vol. 19, no. 1-2, pp. 65-97, February, 2009. Article (CrossRef Link)

[13] J. Lee and D. Zhu, “Shilling attack detection–A new approach for a trustworthy recommender system,” *Informs Journal on Computing*, vol. 24, no. 1, pp. 117-131, January, 2012. Article (CrossRef Link)

[14] Y. Zhang, Y. Tan, M. Zhang, et al., “Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation,” in *Proc. of the 24th International Conference on Artificial Intelligence*, pp. 2408-2414, July 25-31, 2015.

[15] Z. Yang, L. Xu, Z. Cai, and et al. ,“Re-scale AdaBoost for attack detection in collaborative filtering recommender systems,” *Knowledge-Based Systems*, vol. 100, pp. 74-88, May, 2016. Article (CrossRef Link)

[16] F. Zhang, Z. Zhang, P. Zhang, and et al., “UD-HMM: An unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering,” *Knowledge-Based Systems*, vol. 148, pp. 146-166, May, 2018. Article (CrossRef Link)

[17] M. O'Mahony, N. Hurley, N. Kushmerick, and et al., “Collaborative recommendation: a robustness analysis,” *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 344-377, November, 2004. Article (CrossRef Link)

[18] B. Mehta, T. Hofmann, and W. Nejdl, “Robust collaborative filtering,” in *Proc. of the 2007 ACM Conference on Recommender Systems*, pp. 49-56, October 19-20, 2007. Article(CrossRef Link)

[19] Z. Cheng and N. Hurley, “Robust collaborative recommendation by least trimmed squares matrix factorization,” in *Proc. of the 22nd International Conference on Tools with Artificial Intelligence*, pp. 105-112, October 27-29, 2010. Article (CrossRef Link)

[20] B. Mehta and W. Nejdl, “Attack resistant collaborative filtering,” in *Proc. of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 75-82, July 20-24, 2008. Article(CrossRef Link)

[21] H. Yi and F. Zhang, “Robust recommendation method based on suspicious users measurement and multidimensional trust,” *Journal of Intelligent Information Systems*, vol. 46, no. 2, pp. 349-367, April, 2016. Article (CrossRef Link)

[22] F. Zhang, Y. Lu, J. Chen and et al., “Robust collaborative filtering based on non-negative matrix factorization and R1-norm,” *Knowledge-Based Systems*, vol. 118, pp. 177-190, February, 2017. Article (CrossRef Link)

[23] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” *Advances in neural information processing systems*, pp. 1257-1264, December 3-6, 2008.

[24] J. Liu, C. Wu and W. Liu, “Bayesian probabilistic matrix factorization with social relations and item contents for recommendation,” *Decision Support Systems*, vol. 55, no. 3, pp. 838-850, June, 2013. Article (CrossRef Link)

[25] R. Salakhutdinov and A. Mnih, “Bayesian probabilistic matrix factorization using markov chain monte carlo,” in *Proc. of 25th International Conference on Machine Learning*, pp. 880-887, July 5-9, 2008. Article (CrossRef Link)

[26] B. Lakshminarayanan, G. Bouchard and C. Archambeau, “Robust Bayesian Matrix Factorisation,” in *Proc. of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 425-433, April 11-13, 2011.

[27] C. Li and Z. Luo, “A metadata-enhanced variational Bayesian matrix factorization model for robust collaborative recommendation,” *Acta Automatic Sinica*, vol. 37, no. 9, pp. 1067-1076, September, 2011.

[28] H. Li, H. He and Y. Wen, “Dynamic particle swarm optimization and K-means clustering algorithm for image segmentation,” *Optik - International Journal for Light and Electron Optics*, vol. 126, no. 24, pp. 4817-4822, December, 2015. Article (CrossRef Link)

[29] ABS. Serapiao, GS. Correa, FB. Goncalves and VO. Carvalho, “Combining K-means and K-harmonic with fish school search algorithm for data clustering task on graphics processing units,” *Applied Soft Computing*, vol. 41, pp. 290-304, April, 2016. Article (CrossRef Link)

[30] Y. Koren, R. Bell and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30-37, August, 2009. Article (CrossRef Link)

[31] Koren Y. “Factor in the Neighbors: Scalable and accurate collaborative filtering,” *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1-24, January, 2010. Article (CrossRef Link)

Hongtao Yu received his Bachelor's degree in computer application from HeFei University of Technology, China, in 1987 and Master's degree in computer application technology from Yanshan University, China, in 1993. He is now an associate professor in school of information science and engineering, Yanshan University, China. His main research interests include recommender systems, computer network, service-oriented computing, etc.

Lijun Sun received her Bachelor's degree in computer science and technology in Hebei University of Technology, City College, China, in 2013. She is currently working toward her Master's degree in computer science and technology in Yanshan University, China. Her main research interests include recommender systems, information security, etc.

Fuzhi Zhang received his PhD degree in computer application technology from Beijing Institute of Technology, China, in 2003. He is now a professor and PhD supervisor in School of Information Science and Engineering, Yanshan University, China. His main research interests include intelligent network information processing, information security, service-oriented computing, etc.