Journal of Computers Vol. 33 No. 3, June 2022, pp. 207-223
doi: 10.53106/199115992022063303017

Bayesian Personalized Ranking with the Synthesis of Multiple User and Item Classification

Yang Gu[1], Fei Xiong[1,2*], Zixing Wang[1], Ran Lu[1], Yang Zhang[2]

[1] School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
{19231192, xiongf, 19211121, 19231197, zhang.yang}@bjtu.edu.cn
[2] Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing 100044, China

Received 16 April 2022; Revised 16 May 2022; Accepted 16 June 2022

Abstract. In recent years, rapidly evolving networks have facilitated data interactions between users and goods, and recommender systems have emerged due to the needs of the times. Implicit feedback in recommender systems always is divided into the observed item set and unobserved item set. However, items in the unobserved set often are treated equally, and the items that are of potential interest to users are not sufficiently exploited from unobserved items. In this paper, we come up with a method to solve the problem of the insufficient exploration of unrated items, and we contribute to the additional alleviation of rating sparseness. To do so, we propose a novel Bayesian personalized ranking with the synthesis of multiple users and item classification (BPRS). For each user, we divide the items into three categories, which are positive, interest and negative. We conduct multiple user classification and item classification to exploit the items that are of interest and generate the final ranking. Experiments on three real-world datasets demonstrate the effectiveness of our algorithm for greatly improving the accuracy of recommendation results and alleviating the cold-start problem.

Keywords: Bayesian personalized ranking, multiple user and item classification, interest item detection, implicit feedback

1 Introduction

Currently, the large amount of information and the large number of various products in modern society can be overwhelming to some people and make them hesitate in making choices. While recommender systems can provide personalized recommendations of users in minutes according to their interests, they may address information overload in online e-commerce transactions and social networking platforms. In recent years, recommender systems have brought huge benefits to major manufacturers, such as Taobao, Netflix, NetEase CloudMusic, Amazon, and YouTube [1]. From a practical perspective, the key to a personalized recommender system is to model users' preferences for items based on their past interactions, and the process is known as collaborative filtering [2-3]. Among them, matrix factorization is a common method [4] which represents a user or an item by a latent eigenvector, and users' interactions with items are approximated by an inner product of their latent vectors [5]. As a representative task in recommender systems, recommendation ranking applies collaborative filtering to provide a list of potential items for each user.

Usually, recommendation ranking algorithms are divided into two types, i.e., point-wise and pairwise algorithms [6]. Among them, the purpose of point-wise algorithms is to minimize the loss between the true ratings and the predicted ratings [7]. While pair-wise algorithms, such as Bayesian Personalized Ranking (BPR) [8], are designed to maximize the difference in the relative preferences between a pair of rated and unrated items.

Although some progress has been made in collaborative ranking methods based on implicit feedback, we believe that current modeling effects of unobserved items and users' preferences still are unsatisfactory. Recent studies have divided unrated items in more detail and demonstrated that further division of unrated items can improve the effectiveness of recommendations, thereby resulting in a better experience for users. However, existing research does not adequately describe how to divide unobserved items, and users' potential interests need sufficient exploration. Observed ratings should be exploited from more perspectives to characterize users' interests. Therefore, the main issue we address is how to solve the distinction between potential interest items and negative items in unobserved items.

Facing the challenges describes above, we focus on implicit feedback in recommender systems. Also, we divide unrated items of each user into two sets, i.e., the potential items in which users may be interested and the items in which users have no interest at all. Therefore, items are divided into three subsets, i.e., the rated item set

* Corresponding Author

potential interest item set and negative item set. By partitioning a subset of similar items and similar users, we can rank different items from multiple aspects. As shown in Fig. 1, we propose a new method of discovering the collections of potential interest items with specific processes.

We explore the potential interests of users in two aspects. For the first aspect, we find similarity relationships between users. Then, we set users with similarity greater than $threshold1_{user}$ as similar users, and then summarize the rated items of similar users into a collection. According to the frequency of the occurrence for each item, items with a frequency of occurrence that is greater than $threshold2_{user}$ are considered as interest items set_1 excavated by $threshold2_{user}$. For the second aspect, we find similarity relationships between items and set the items with similarity greater than $threshold1_{user}$ as similar items. Then, according to each user's rated items, we summarize a similar item collection. Items with a frequency of occurrence that is greater than $threshold2_{item}$ are considered as interest items set_2. Then, we merge those two sets as interest items. Finally, for each user, we choose rated items as positive items, whereas items other than interest items and positive items are regarded as negative items. After dividing the three categories, we can learn user preferences and item features through Bayesian inference and predict the sort of items. Our contribution is summarized in the following points:

(1) We use user similarity and item similarity to determine whether an unrated item is an interest item. In addition, we design different similarity thresholds to find similar users and similar items.

(2) We propose a new algorithm of Bayesian recommendation ranking with the synthesis of multiple user and item classification (BPRS). We combine similarity of users and similarity of items and filter interest items. Then, we synthesize potential interest items obtained from two aspects, and sort them with the Bayesian inference framework.

(3) We perform extensive experiments using real-world datasets and compare the results with many state-of-the-art algorithms. The results show that BPRS improves the accuracy and can alleviate the cold-start problem effectively.

The rest of this paper is organized as follows. Section 2 discusses related works about the BPR algorithm and introduces improved algorithms based on BPR; it also recommends some methods of calculating user and item similarity. Our problem definition and models are introduced in Section 3. Section 4 illustrates the dataset, experimental process, and the analysis of the results. Section 5 presents our conclusions and the directions of our future work.

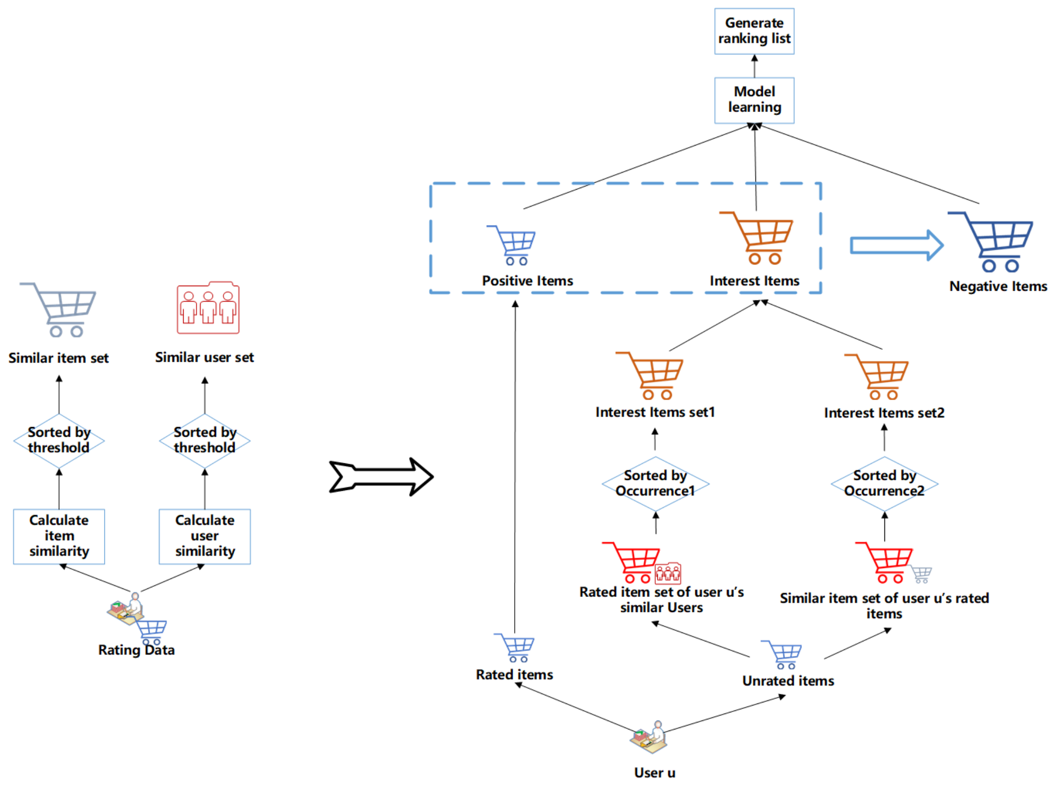


Fig. 1. The procedure of the BPRS model

2 Related Work

In this section, first, we introduce the BPR algorithm that our work correlates with, then we introduce some BPR-based algorithms, and finally we summarize the methods that calculate user or item similar relationships.

2.1 Bayesian Personalized Ranking

Rendle et al. proposed the Bayesian Personalized Ranking algorithm in 2009, which predicts rankings based on pairwise preferences among two items for a single user. It aims to provide each user with a sorted list of items. By changing the prediction of the rating of a single item to a pairwise rating between items, we can alleviate the problem of data sparsity to a large extent.

There are two assumptions in the BPR algorithm. One is that the ratings of user *u* on different items are independent of each other, which means, the ratings of user *u* on items *i* and *j* have nothing to do with other items. The other assumption is that the ratings of user *u* on items *i* and *j* have nothing to do with other users.

Then, the likelihood of the BPR algorithm for all users can be expressed as:

$$BPR = \prod_{u \in U} \prod_{i \in I_u^+} \prod_{j \in I \setminus I_u^+} P\left(r_{ui} > r_{uj}\right)\left(1 - P\left(r_{uj} > r_{ui}\right)\right). \quad (1)$$

where U is the user set, I_u^+ is the set of positive items that have been rated by user u, and r_{ui} represents user u's rating on item i.

2.2 BPR-related Algorithm

In recent years, many people have proposed relevant algorithms based on BPR. Studies have shown that most of collaborative filtering algorithms based on matrix decomposition are improved mainly in the following, i.e., weighting, sampling mode, the loss function, and constraint terms [9]. For the improvement of BPR, these aspects also are included, and new strategies are introduced to alleviate various problems, such as data sparsity and cold start.

The group preference-based Bayesian personalized ranking algorithm (GBPR) [10] linearly blends individual preferences with group preferences. The ranking method with a fast adaptive and context-dependent sampling algorithm (AOBPR) [11], accelerates the convergence rate, especially with long-tail effects. Social Bayesian Personalized Ranking (SBPR) [12] considers the impact of real-life relationships and introduces social trust relationships into the recommendation process. Group Bayesian Personalized Ranking with Rich Interactions (GBPR+) [13] changes pair-wise levels between individual items in GBPR to pair-wise relationships between group items. Trinity Preference-based Bayesian Personalized Ranking (TBPR) [14] further alleviates the sparseness of item information by dividing items into four sets based on user-related location information and the strength and weakness of interactions. BPR for heterogeneous implicit feedback (BPRH) [15] considers user auxiliary actions as well as deletion actions as implicit feedback. By using these user actions, BPRH increases the accuracy of item recommendations.

In recent years, various scholars also have made some efforts to alleviate the sparsity of data, and solve the cold-start problem. Double Bayesian Pairwise Learning (DBPL) [16] synthesizes new pair-order preference relationships through a preference between two pairs of items. Rating Bayesian Personalized Ranking (RBPR) [17] combines the matrix decomposition [18] with the BPR algorithm, thereby improving the effect to some extent in consideration of implicit feedback and display feedback.

2.3 Similarity Calculation

In recent years, similarity characteristics of users and items have been applied extensively to recommender systems in collaborative filtering. The most popular similarity metrics are Cosine, Pearson correlation coefficient (PCC), Constrained Pearson Correlation Coefficient (CPCC), Jaccard, etc.

Among these, the PCC considers a linear correlation between two rating vectors, which is defined as:

$$Sim_{pcc}(a,b) = \frac{\sum_{j \in I_a \cap I_b} (R_{a,j} - \bar{R}_a)(R_{b,j} - \bar{R}_b)}{\sqrt{\sum_{j \in I_a} (R_{a,j} - \bar{R}_a)^2} \sqrt{\sum_{j \in I_b} (R_{b,j} - \bar{R}_b)^2}}. \quad (2)$$

where j is a collection of items jointly evaluated by users a and b; $\bar{R}_a$ is the average rating given by users a; $R_{a,j}$ is the rating of item j given by user a. PCC is used extensively to measure the degree of correlation between two variables, with values between -1 and +1.

CPCC, based on the PCC, we use the median of the rating to represent the boundary of user interest, which is defined as:

$$Sim_{cpcc}(a,b) = \frac{\sum_{j \in I_a \cap I_b} (R_{a,j} - R_{med,a})(R_{b,j} - R_{med,b})}{\sqrt{\sum_{j \in I_a} (R_{a,j} - R_{med,a})^2} \sqrt{\sum_{j \in I_b} (R_{b,j} - R_{med,b})^2}}. \quad (3)$$

where $R_{med,a}$ is the median rating given by users a, and $R_{a,j}$ indicates that user a's rating on item j.

Many scholars also are innovating constantly on this basis. A new item similarity based on the α-divergence method [19] is calculated based on the probability density distribution of ratings, which greatly reduces the dependence on co-rated cases. The method also considers the influence of scores and the ratio of co-rated cases on the results, which effectively improve the accuracy of recommendations. In addition, a new method derived from the subspace clustering approach was proposed to construct the neighbor user tree, and a new similarity method was used to find similar users. Triangle multiplying Jaccard (TMJ) similarity [20] considers the length and angle of rating vectors and non-co-rating users. Improve_CF [21] consists of three parts, i.e., Jaccrd, CPCC, and CF-Coefficient [21], to further improve the effect.

3 Recommendation with The Synthesis of Multiple User and Item Classification

In this section, first we elaborate on our problem and explain the variables we used. Then, we provide a concrete approach on how to find a collection of interest items. Finally, we introduce the new recommendation algorithm, BPRS, and detail its partitioning process. The entire process of BPRS is shown in Section 1, Fig. 1.

3.1 Problem Definition

For each user u, in addition to rated items, numerous unknown items include the items which user may like potentially and the items users do not like. Based on this, for each user u, we divide the remaining items into two categories, i.e., potential interest and negative items. Hence, we can define that:

$$item\ i \in I_u^{positive}, item\ j \in I_u^{interest}, item\ k \in I_u^{negtive}$$

where $I_u^{positive}$ refers to rated items of user u, $I_u^{interest}$ refers to potential interest items of user u, and $I_u^{negtive}$ refers to negative items of user u. Then, we use $i >_u j$ to represent that user u prefers item i to item j, and we define $i >_u j$, $j >_u k$. Because of the transitivity of $>_u$, we can get $i >_u k$. Since we can differentiate positive items based on the rating data, our model mainly extracts the interest items in the intermediate layer. After dividing the interest items, we will choose the remaining items as negative items.

We define U as the set of users, and I as the set of items. Users and items constitute the interactions with a matrix R, where we assume there are M users and N items, and therefore, we have $R \in R^{M \times N}$. Then, we define the user-related matrix P as an $M \times f$ matrix $P^{M \times f}$ and the item-related matrix Q as an $N \times f$ matrix $Q^{N \times f}$ where f is the number of latent factors [22]. In addition, we define the global preference on items is $b^{1 \times N}$, and each element in the interaction matrix R as r_{ui}, representing user u's rating on item i. In building the model, we use $u \in [1, M]$ to represent a user and $i \in [1, N]$ to represent an item. Our goal is to predict the hidden matrix $\hat{P}^{M\times f}$, the $\hat{Q}^{N\times f}$ and the global preference $b^{1\times N}$ through existing rating information in order to obtain the predicted rating matrix. And for each user u we have $\hat{R}_u^{1\times N} = \hat{U}_u^{1\times f} \times \hat{I}^{f\times N} + b^{1\times N}$.

3.2 Dividing the Intermediate Layer with Multiple User and Item Classification

It is obvious that, in the unrated items, there are still many items that are of interest to users. Therefore, for each user, we divide all items into three categories, i.e., a set of positive items, a set of items of interest, and a set of negative items.

To extract the intermediate layer of items, we have to analyze which items users might like. First, user u prefers items purchased by users who have similar characteristics of its own interests. Therefore, we can think that user u may buy the items that are similar to the similar users' interests. Meanwhile, users also will purchase the items with similar characteristics to the items they have already purchased. Therefore, we will divide the middle layer by dealing with both similar users and purchased items. First, we should judge the relationship between two users or two items. In that case, we choose CPCC similarity to measure the similarity relationship between two users or two items. Then, we select the users or items that have similarities greater than a certain threshold. Considering that the interests of different users are independent, and the number of similar users per user varies greatly, we delineate similar users with a single threshold, so that the numbers of similar users for different users are different. At the same time, since there maybe common similarities and universalities between items, the number of similar items per item is close, and then we select the same number of similar items for each rated item. To do so, the similarity threshold of each item should be different and unique. Then, we use two methods to get the appropriate thresholds. We designate the threshold for dividing similar users as $threshold1_{user}$ and the threshold for dividing similar items as $threshold1_{item}$. For the calculation of $threshold1_{user}$, first we rank the calculation results of each user's similarities with all of the other users, so that we can get a ranked similarity matrix, which we refer to as the $similar_{ranked\text{-}user}$ matrix. Note that, in this ranked similarity matrix, each column belongs to one user, and, in each column, her/his similarities with other users are ranked among the highest to the lowest. Then, we average the similarity value of the T_1 th row in the $similar_{ranked\text{-}user}$ matrix to obtain $threshold1_{user}$, as shown in Fig. 2. The calculation process is as follows:

$$thereshold1_{user} = \frac{\sum_{i=1}^{N_{user}} similar_{ranked-user}(T_1, i)}{M}. \tag{4}$$

where M is the number of users, i is the index of a user, T_1, is the row number of the $similar_{ranked\text{-}user}$ matrix, and $similar_{ranked\text{-}user}(T_1, i)$ is the value of the $T_1 th$ row and ith column in this matrix. Then, we select users whose similarities are greater than $thereshold1_{user}$ as similar users.

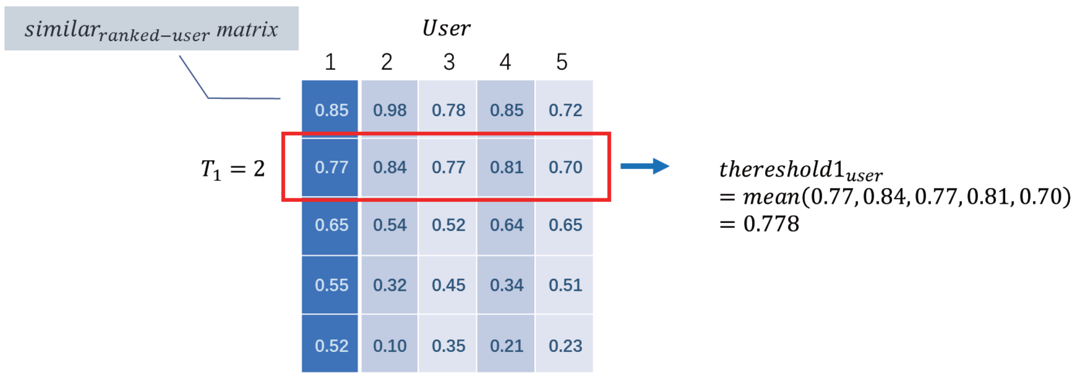


Fig. 2. The calculation process of $threshold1_{user}$

For the calculation of $threshold1_{item}$, we also rank the calculation results for each item by similarity values, so that we can get the $similar_{ranked\text{-}item}$ matrix, where each column belongs to one item, and the similarities in each column are ranked from the highest to the lowest. Then, for each item, we choose the value of the T_2th row in each column of the $similar_{ranked\text{-}item}$ matrix as $threshold1_{item}$. Note that $threshold1_{item}$ is a one-dimensional array, which means each item has its unique threshold, as shown in Fig. 3. The calculation process is as follows:

$$threshold1_{item}(i) = similar_{ranked-item}(T_2, i)\,. \tag{5}$$

where i is the item index, $threshold1_{item}(i)$ is the unique threshold of item i, and $similar_{ranked\text{-}item}(T_2, i)$ is the value of T_2th row and the ith column in the matrix. Then, we select items whose similarities are greater than $theresold1_{item}$ as similar items.

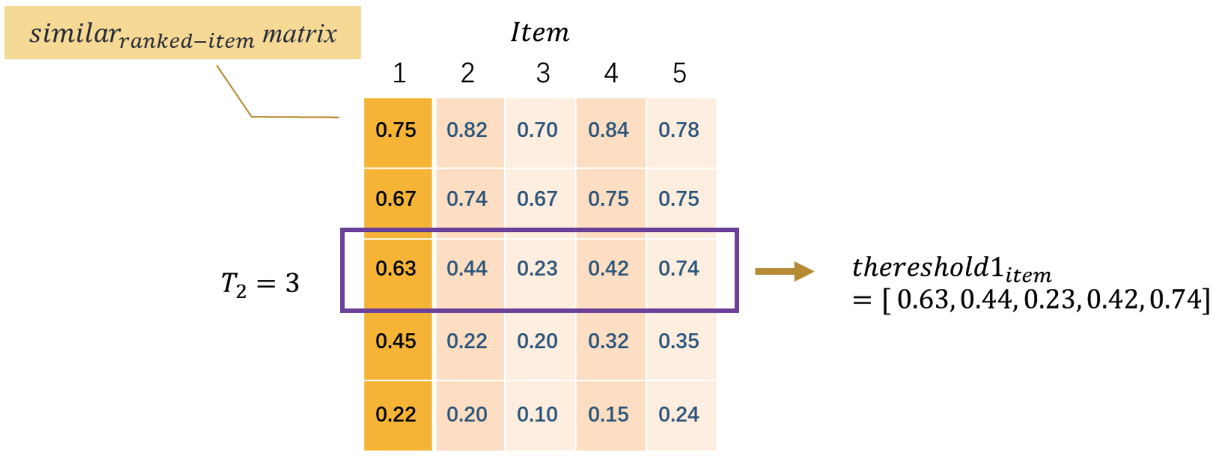


Fig. 3. The process of calculating $threshold1_{item}$

Next, we summarize the rated items of each similar user and record the frequency of occurrence for these items. If the frequency of occurrence for an item is greater than $threshold2_{user}$, we identify the item as an interest item explored in the user aspect, as shown in Fig. 4. For the calculation of $threshold2_{user}$, we multiply the number of similar users by a specific percentage, β_1 to get $threshold2_{user}$. The calculation process is as follows:

$$threshold2_{user}(u) = N_{sim-user}(u) \times \beta_1\,. \tag{6}$$

where $N_{sim-user}$ is the number of similar users, and u is the index of a user.

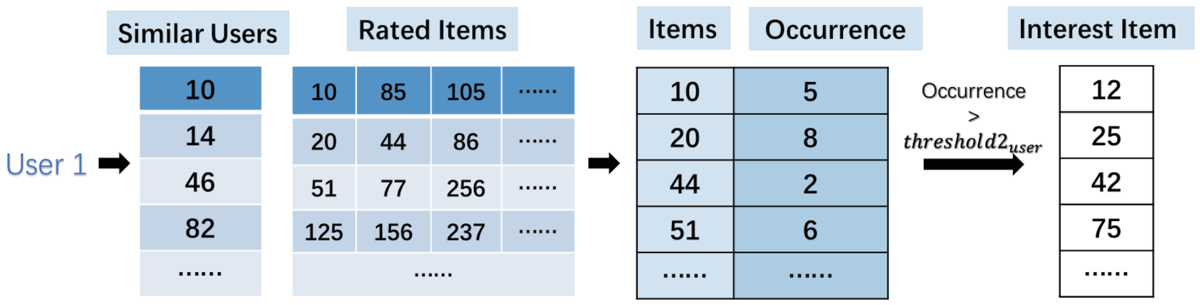


Fig. 4. Collection of items of interest from the user perspective

In the same way, we summarize the set of similar items for all the rated items of user u, and we record the frequency of occurrence for each item. If the frequency of occurrence for an item is greater than $threshold2_{item}$, we identify the item as an item of interest explored in the item aspect, as shown in Fig. 5. Also, we calculate $threshold2_{item}$ in a similar way, and we multiply the number of similar items by a specific percentage, β_2, to get $threshold2_{item}$. The calculation process is as follows:

$$threshold2_{item}(i) = N_{rated-item}(u) \times \beta_2 \,. \tag{7}$$

where $N_{rated\text{-}item}(u)$ is the number of user u's rated items.

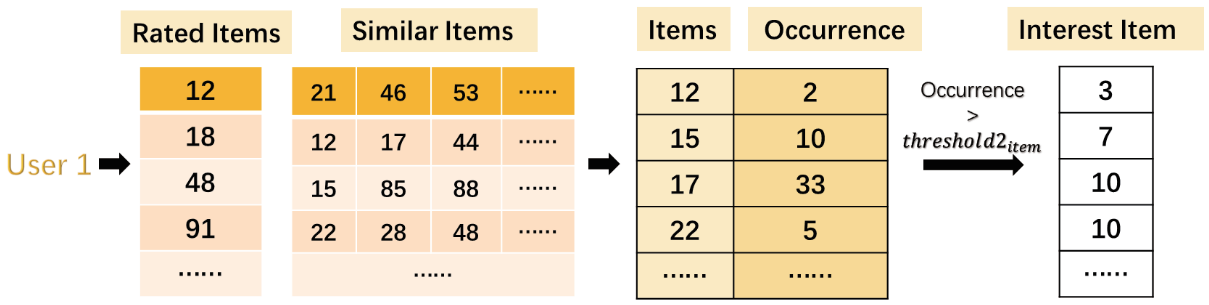


Fig. 5. Collection of interest items in the item aspect

To summarize the process, we combine the items explored from the user aspect with the items explored in the item aspect, so that we can get the set of items of interest. Then, items other than the items of interest and positive items are treated as negative items. Generally, for each user, u, total items can be divided into three sets of items as follows:

- $I_u^{positive}$: *Rated items*
- $I_u^{interest}$: *Interest items*
- $I_u^{negtive}$: *Negative items*

We calculate ratings of items by matrix decomposition and they are estimated as follows:

$$\begin{aligned} \hat{r}_{ui} &= b_i + p_u^T q_i . \\ \hat{r}_{uj} &= b_j + p_u^T q_j \,. \\ \hat{r}_{uk} &= b_k + p_u^T q_k \,. \end{aligned} \tag{8}$$

Among them, b_i, b_j, b_k are the inherent biases of item i, j, and k, respectively. Each user u is associated with the latent vector $p_u \in R^{f\times1}$, and item i, j and k are related to the vectors q_i, q_j, $q_k \in R^{f\times1}$. Hence, each rating is based on the inner product of these two vectors, so missing values in the dataset can be predicted.

3.3 Sorting Algorithm

For an individual user u, the set of items is divided into three parts, which are the positive item set, the interest item set, and the negative set, so the item-set group for each user is given by

$$G(u) = \{(i,j,k) | i \subseteq I_u^{positive}, I_u^{interest}, I_u^{negtive}\} . \tag{9}$$

Therefore, we use a multi-layer Bayesian personalized rank model to represent the hierarchical relationships among items of different grades. We define that

$$pref(u,i) \succ pref(u,j) \succ pref(u,j)$$

Based on the above conditions, because the posterior probability is proportional to the likelihood function, the posterior probability of all user preferences can be written as the following likelihood function:

$$\prod_{u\in U}\prod_{(i,j,k)\in\mathcal{I}} p\left(\hat{r}_{ui} > \hat{r}_{uj} > \hat{r}_{uk}\middle|\theta\right) = \prod_{u\in U}\prod_{(i,j,k)\in G(u)} p\left(\hat{r}_{ui} > \hat{r}_{uj}\middle|\theta\right)\cdot p\left(\hat{r}_{uj} > \hat{r}_{uk}\middle|\theta\right). \quad (10)$$

Among them, we have

$$p(i >_u j|\theta) \coloneqq \sigma\left(\hat{r}_{uij}(\theta)\right). \quad (11)$$

At the same time, it holds true that

$$\sigma\left(\hat{r}_{uij}(\theta)\right) = \frac{1}{1+e^{-\left(\hat{r}_{uij}(\theta)\right)}}. \quad (12)$$

$$\hat{r}_{uij}(\theta) = \hat{r}_{ui}(\theta) - \hat{r}_{uj}(\theta). \quad (13)$$

where $\hat{r}_{uij}(\theta)$ represents the preference between two items of an individual user, and θ represents the variables that have to be optimized.

To reduce the complexity of the computations, we use log likelihood and finally obtain the following equation:

$$\max_{\Theta} F = \sum_u\sum_{i,j}\alpha\ln\sigma\left(\hat{r}_{uij}(\theta)\right) + \sum_u\sum_{j,k}(1-\alpha)\ln\sigma\left(\hat{r}_{ujk}(\theta)\right). \quad (14)$$

Among them, $\Theta = \{\boldsymbol{p}_u, \boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{b}_i, \boldsymbol{b}_j, \boldsymbol{b}_k, u \in \boldsymbol{U}, i, j, k \in \boldsymbol{I}\}$, α is the weight parameter, which is used to control the weight of the impact of the first and second terms. To avoid overfitting, we add the regularization term to obtain the final objective function.

$$\max_{\Theta} F = \sum_u\sum_{i,j}\alpha\ln\sigma\left(\hat{r}_{uij}(\theta)\right) + \sum_u\sum_{j,k}(1-\alpha)\ln\sigma\left(\hat{r}_{ujk}(\theta)\right) + \sum_u\sum_{i,j,k}\frac{\lambda}{2}\left(\|\boldsymbol{p_u}\|^2 + \|\boldsymbol{q_i}\|^2 + \|\boldsymbol{q_j}\|^2 + \|\boldsymbol{q_k}\|^2 + b_i^2 + b_j^2 + b_k^2\right). \quad (15)$$

3.4 Learning Model

We find the parametric optimal solution by using stochastic gradient descent. But to learn the set of parameters $\Theta = \{\boldsymbol{p}_u, \boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, b_i, b_j, b_k\}$ that maximize the posterior probability, we use a stochastic gradient rise approach:

$$\theta = \theta + \eta\,\Delta\theta. \quad (16)$$

where η is the learning rate, and $\Delta\theta$ is a differential of the model parameters. The final calculation yields the following results:

$$
\begin{aligned}
b_i^1 &= b_i^0 + \eta\left(\alpha\frac{1}{1+e^{\hat{r}_{uij}}} - \lambda\, b_i\right). \\
b_j^1 &= b_j^0 + \eta\left(\alpha\frac{-1}{1+e^{\hat{r}_{uij}}} + (1-\alpha)\frac{1}{1+e^{\hat{r}_{ujk}}} - \lambda\, b_j\right). \\
b_k^1 &= b_k^0 + \eta\left((1-\alpha)\frac{-1}{1+e^{\hat{r}_{ujk}}} - \lambda\, b_k\right). \\
p_u^1 &= p_u^0 + \eta\left(\alpha\,\frac{1}{1+e^{\hat{r}_{uij}}}\,(q_i - q_j) + (1-\alpha)\frac{1}{1+e^{\hat{r}_{ujk}}}(q_j - q_k) - \lambda\, p_u\right). \\
q_i^1 &= q_i^0 + \eta\left(\alpha\frac{1}{1+e^{\hat{r}_{uij}}}\,p_u - \lambda\, q_i\right). \\
q_j^1 &= q_j^0 + \eta\left(\alpha\frac{-1}{1+e^{\hat{r}_{uij}}}\,p_u + (1-\alpha)\frac{1}{1+e^{\hat{r}_{ujk}}} - \lambda\, q_j\right). \\
q_k^1 &= q_k^0 + \eta\left((1-\alpha)\frac{-1}{1+e^{\hat{r}_{ujk}}}\,p_u - \lambda\, q_k\right).
\end{aligned}
\tag{17}
$$

The algorithms are shown in Table 1 as follows:

Table 1. Learning in the proposed algorithm BPRS

Algorithm 1. Learning in the proposed algorithm BPRS

Input: user set *U*, item *I*, ratings *R*, the dimension of feature space *f*, regularization parameters λ_θ, weight parameter α, learning rate η.

Output: $\Theta \in \{\boldsymbol{p}_u, \boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{b}_i, \boldsymbol{b}_j, \boldsymbol{b}_k\}$

```
1.  Randomly generate training set and test set from the dataset
2.  Do item classification
3.  Initialize the parameters $\Theta \sim N(0, 0.001)$
4.  for iterations do
5.    for training sample do
6.      Randomly sample a user $u \in U$
7.      Randomly sample an event $i \in I_u^{positive}$
8.      Randomly sample an event $j \in I_u^{interest}$
9.      Randomly sample an event $k \in I_u^{negtive}$
10.     $\boldsymbol{p}_u \leftarrow \boldsymbol{p}_u + \eta\,\Delta\,\boldsymbol{p}_u$
11.     $\boldsymbol{q}_i \leftarrow \boldsymbol{q}_i + \eta\Delta\,\boldsymbol{q}_i$
12.     $\boldsymbol{q}_j \leftarrow \boldsymbol{q}_j + \eta\,\Delta\,\boldsymbol{q}_j$
13.     $\boldsymbol{q}_k \leftarrow \boldsymbol{q}_k + \eta\,\Delta\boldsymbol{q}_k$
14.     $b_i \leftarrow b_i + \eta\,\Delta b_i$
15.     $b_j \leftarrow b_j + \eta\,\Delta b_j$
16.     $b_k \leftarrow b_k + \eta\,\Delta b_k$
17.   end
18. end
19. return $\Theta$
```

3.5 Complexity

The computational complexity of BPRS consists of three main parts, i.e., pre-classification, training the model, and rating the prediction. The complexity of the classification is equal to $O(\sum_u |\bar{I_u}|)$, where $|\bar{I_u}|$ is the number of items that are not rated by each user. The total number of unrated items for all users is equal to $|\bar{R}|$. In this section, we use two methods to classify the items, so the complexity of the pre-classification is $O(2|\bar{R}|)$. The complexity of the process of training the model can be divided into three parts. In every iteration, the computational complexity of p_u is $O(f \cdot |R|$, where f is the dimension of the feature space for the user and the item and $|R|$ is the number of training ratings. And the computational complexity of b_i, b_j, b_k is $O(3 \cdot f \cdot |R|$, and the computational complexity of q_i, q_j, q_k is $O(3 \cdot f \cdot |R|)$. Then the complexity of the second part is $O(7 \cdot f \cdot |R| \cdot N)$, where N is the number of iterations. For the rating prediction, the complexity is $O(m \cdot n \cdot f)$, where m is the number of users, and n is the number of items. Therefore, according to the steps of the algorithm, the time complexity of our algorithm is $O(7 \cdot f \cdot |R| \cdot N + m \cdot n \cdot f + 2|\bar{R}|)$.

4 Experimental Setup

In this section, we describe our experimental setup and analyze the results. In addition, in order to show the superiority of our algorithm, we compare our model with other algorithms.

4.1 Dataset

In this part, we evaluate the effectiveness of our model on three real-world datasets of our experiments. The three datasets we selected are Movielens 100k (ML-100K), Movielens 1M(ML-1M), and Epinions. Among them, Movielens is published by Grouplens Lab [23], and Epinions.com is a well-known creative common site and review site, where users can review items and assign the ratings in the range of 1 to 5 [24]. The statistics of these datasets are shown in Table 2. Movielens 100k, which includes 100,000 ratings, consists of 943 users and 1,682 items. Movielens 1M includes 1,000,209 ratings, with 6,040 users and 3,706 items. Epinions consists of 276,116 ratings, and it has 7,411 users and 8,728 items. All of the scores are integers in the range of 1 to 5, and the higher scores indicate higher preferences.

For each dataset, we list items rated 1-5 as positive items for users. In this experiment, we used 5-fold cross-validation with five cross-validations per dataset, and we use the average as the experimental result.

Table 2. Details of the dataset

Dataset	Movielens 100K	Movielens 1M	Epinions
User	943	6040	7411
Item	1682	3706	8728
Ratings	100,000	1,000,209	276,116
Sparsity	93.7%	95.5%	99.6%
Average number of ratings per user	106	166	37
Average number of ratings per item	25	253	32

For the evaluation of the models, we use three traditional and general methods, i.e., precision [25], recall [26], and F1 [27], to evaluate the effect of our ranking results, top-N.

For, precision, recall, and F1, the algorithm is defined as follows:

$$P@L = \frac{1}{m}\sum_{u=1}^{m}\frac{D_u(L)}{L}. \tag{18}$$

$$R@L = \frac{1}{m}\sum_{u=1}^{m}\frac{D_u(L)}{C_u(L)}. \tag{19}$$

$$F1@L = \frac{2Pre@L \times Rec@L}{Pre@L + Rec@L}. \tag{20}$$

where L is the length of the recommendation list, m is the total number of users, $D_u(L)$ represents the number of recommendation results verified in the test set, and $C_u(L)$ indicates the number of items selected by the user u in the test set.

4.2 Baseline

To compare the superiority of the algorithms, we compare our algorithm with the existing state-of-the-art algorithms. Therefore, we select five baseline algorithms and implemented them through librec or the open-source code on github, including BPR, AOBPR, EALS, GBPR, PNMF, NMFItemItem, and BUIR, which are described as follows:

BPR [8]: The algorithm transforms the original rating of a single user on a single item into a pair sort between items within a single user, thereby optimizing the sorting of the product preferences of each user.

AOBPR [11]: The algorithm accelerates the convergence of SGD mainly by non-uniformly sampling the non-implicit feedback data, and it avoids long-tail effects.

EALS [4]: The algorithm is based on an element alternating least squares algorithm that can effectively optimize the matrix decomposition model with variable weighted missing data.

GBPR [13]: The algorithm proposes group preferences between users, applying the SGD algorithm to train the model.

PNMF [28]: The algorithm combines NMF and conventional SVD or PCA decomposition to have better matrix orthogonality and sparsity.

NMFItemItem [29-30]: The algorithm differs from the original non-negative matrix decomposition in that it uses a decomposition of the item-item matrix, which is much easier to use for fast online recommendations with a lot of new or fast-changing users.

BUIR [31]: The algorithm proposes a novel OCCF framework, which does not require negative sampling.

Since there are many parameters in the recommended algorithm, its selection will greatly affect the effect of the algorithm. In our experiments, we modulate the parameters of the algorithm appropriately before comparison. In addition, we set the number of hidden features f for the user and the item to 50, and for the BPR, AOBPR, EALS, GBPR, PNMF, and NMFItemItem we set the learning rate to 0.1, and the regularization coefficient for the user and the item to 0.01. For BUIR, we set the learning rate to 0.001 and the regularization coefficient for the user and the item to 0.0001. For the GBPR, we set the group size to 3. For the AOBPR, we set the distribution to 5.

In our model, we use different parameters for different datasets. We set T_1 to 51 and T_2 to 280 in three datasets. For α, we set it to 0.24 in ML 100K and Epinions, we set it to 0.3 in ML 1M. For β_1, we set it to 0.20 in ML-100K, 0.14 in ML-1M, 0.013 in Epinions. For β_2 , we set it to 0.15 in ML-100K, 0.18 in ML-1M, 0.35 in Epinions. All test results were obtained using five-fold cross-validation.

4.3 Analysis of the Experimental Results

In this section, by analyzing the experimental results, we prove the superiority of our model over other algorithms, and we prove that it has better cold-start capability.

4.3.1 The Effect of Parameters

To test the performance of our model, we conduct experiments to assess the following parameters, i.e., the length, L, of the recommended list, the threshold, T_1, selected by similar users, and the weight factors, α.

Fig. 6 shows the trend of the accuracy of our algorithm with the recommended list length L, and in our experimental results, we can find that precision values decrease and recall values increase, because as L increases, precision predicts correct items are limited; the precision denominator grows rapidly, while the recall denominator test set length remains unchanged and molecular predicts correct items. But for the different datasets, we find that, as the datasets became sparser, the precision and recall values became correspondingly lower. For example, for epinions, the model predicts poorly. But for the two datasets of movielens, our model performed better because it had more data. In the following experiments, we set the value of L to 10 for convenience.

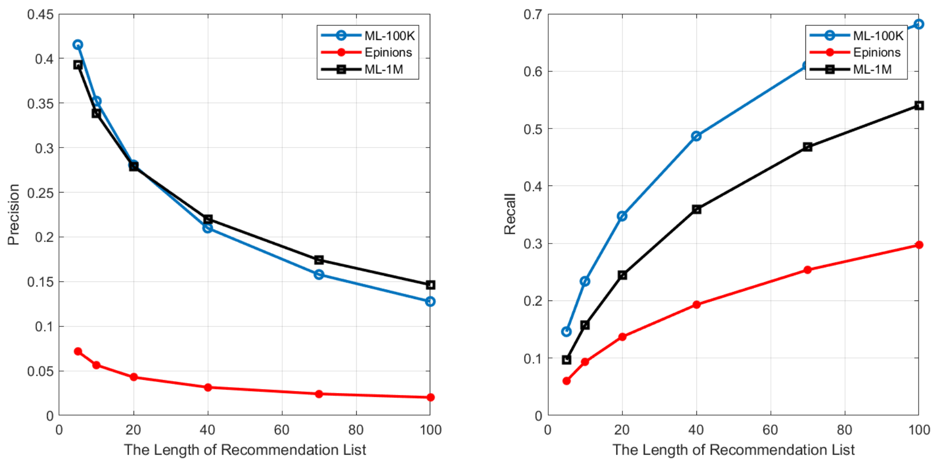


Fig. 6. Impact of the length of the recommendation list

Our model also will be affected by the hyper-parameters of the model, in which the settings of the threshold and the parameter α have a large impact on the model. Therefore, Fig. 7 shows the effect of the threshold value of T_1, and Fig. 8 shows the effect of the α.

When performing experiments to test the effect of the threshold T_1, we set T_2 to 280 and α to 0.24, and Fig. 5 shows the results of the experiment. By changing T_1, we find that the accuracy increases and then decreases or basically remains unchanged as T_1 changes, while the most suitable values of T_1 generally do not exceed 100. At the same time, we find that too few users will reduce the accuracy because too little user data is obtained, which is not much help to the model, and too many similar users will add too much noise, thus including more negative items in the potential interest item. From Epinions experiments, we find that selecting appropriate similar users can improve the accuracy of the data significantly with large sparsity. This is because the original dataset is very sparse, and by adding user similarities, the sparseness can be significantly improved, allowing the discovery of a large number of items of interest. For the Movielens 100k and Movielens 1M, because we chose the value of T_2 to be 280, we have screened a part of the items, so, although the accuracy of the prediction is improved by changing the value of T_1, it is not improved significantly.

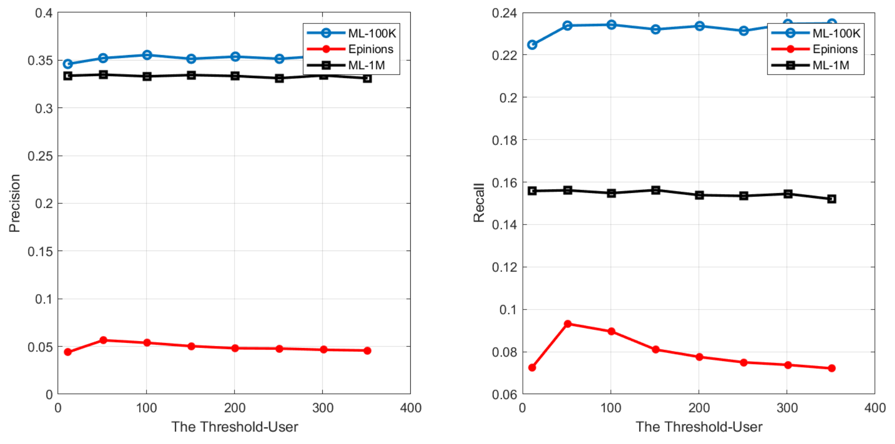


Fig. 7. Impact of T_1

When performing experiments to test the effects of α, we set T_1 to 51, and we set T_2 to 280. The results of the experiment are shown in Fig. 8, and it is apparent that the recommended accuracy increases first and then decreases as α increase. This phenomenon is evident in all three datasets. This is because we divide all the items into three categories, i.e., positive items, potential interest items, and negative items. For these three sets of items, users have similar attitudes to positive items and potential interest items, and far better attitudes to potential interest items than negative items. Therefore, for the objective function, we should add more weight to the second term, the pairwise between the user potential interest item and the negative items. However, if α is too large, it means that the whole weight is added to the latter term. Blindly expanding the pair-wise relationship between interest items and negative items will ignore the priority of positive items, thereby increasing the priority of some incorrect items of interest, resulting in the introduction of more noise, and affecting the ranking.

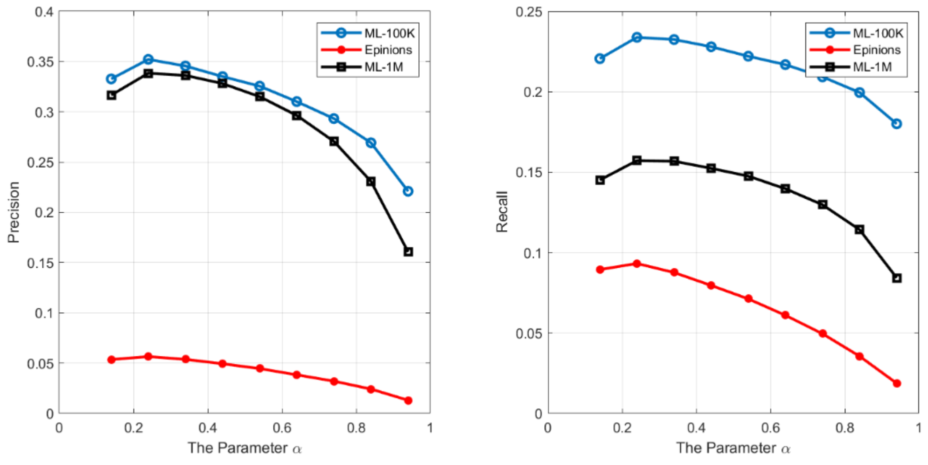


Fig. 8. Impact of α

4.3.2 Comparison with Baselines

To illustrate the performance of BPRS, we compare our model with various baselines. Table 3 shows the results of our experiments on three real-world datasets, and by analyzing the experimental results, we can draw the following conclusions:

- BPRS has achieved good results on our evaluation indicators, which are precision, recall, and F1. Its main contribution is to obtain each user's hidden interest items by simultaneously using the similarity between users and items, further alleviating the sparseness of data and including a large number of hidden interest items in the rating items.
- BPRS performed well under all conditions, and BPR-based algorithms also performed well. GBPR further improved the effect by exploiting group preferences between users. AOBPR mainly speeds up the convergence, but it has little effect on improving accuracy. In addition, we also find that PNMF has lower accuracy, because simply using PNMF dimensionality reduction will cause data points to overlap between low dimensions, which will affect the calculation of similarity. Although NMFItemItem also uses a non-negative matrix decomposition, it decomposes the item matrix into the inner product of the two matrices, taking into account the user's current purchase situation and predicting the next purchase. To some extent, the accuracy rate has been improved.
- In addition, we find that BPRS has greatly improved BUIR for two main reasons. First, BUIR is better at recommending sparse matrices that contain large numbers of users, such as user-item systems with both users and items greater than 10,000. Second, BUIR did not conduct a negative sample, but randomly generated negative items from items that had never been rated. BPRS generates and grades negative items based on existing data, thus making the hierarchical relationship of items within each user more reasonable.
- Based on the dataset, the performance of the two datasets of ML is similar due to the similarity of the features. For epinions, the results are low because the dataset is very sparse. In conclusion, BPRS improved by 11.7% on M L-1M, 15.7% on M L-100K over BPR, and 33.1% on epinions.

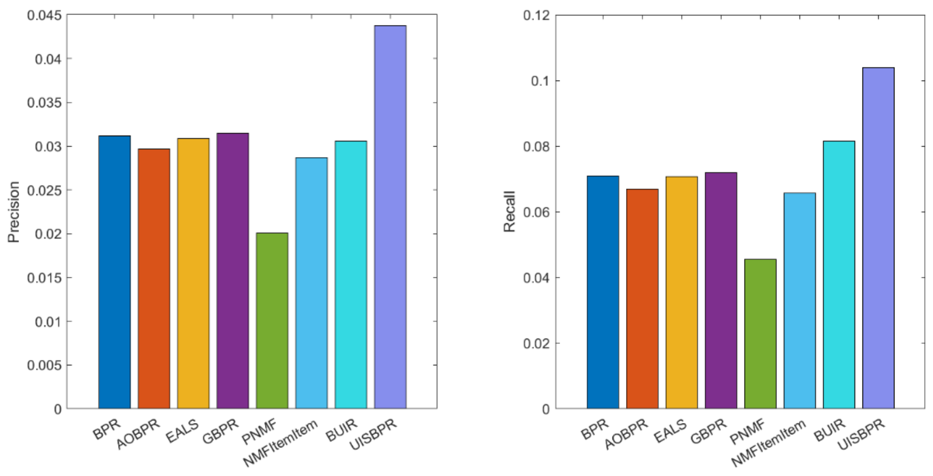
Table 3. Performance evaluation of different recommendation algorithms

	ML-100k			ML-1M			Epinions		
	R @10	P @10	F1 @10	R @10	P @10	F1 @10	R @10	P @10	F1 @10
BPR	0.2028	0.3028	0.2429	0.1407	0.3053	0.1926	0.0657	0.0443	0.0529
AOBPR	0.1875	0.2866	0.2266	0.1248	0.2865	0.1738	0.0660	0.0456	0.0539
EALS	0.1799	0.2539	0.2105	0.1476	0.2818	0.1937	0.0681	0.0439	0.0534
GBPR	0.2198	0.3326	0.2646	0.1359	0.3070	0.1884	0.0667	0.0445	0.0534
PNMF	0.1705	0.2674	0.2082	0.1031	0.2563	0.1470	0.0461	0.0335	0.0388
NMF-Item-Item	0.1949	0.3026	0.2371	0.1122	0.2616	0.1570	0.0627	0.0410	0.0496
BUIR	0.2041	0.3144	0.2475	0.1332	0.2898	0.1825	0.0774	0.0485	0.0590
BPRS	**0.2338**	**0.3522**	**0.2810**	**0.1572**	**0.3384**	**0.2147**	**0.0932**	**0.0565**	**0.0704**

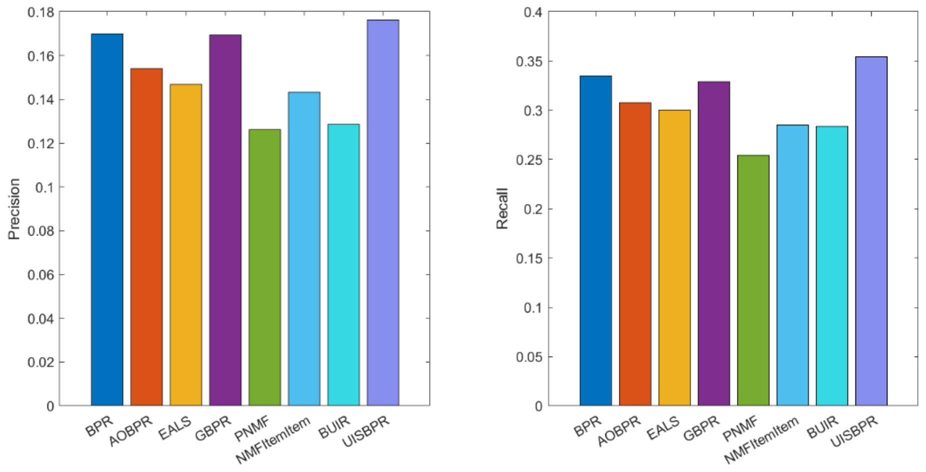
4.3.3 Results on Cold-start Users

One of the most common problems of recommender systems is their cold-start problem. For a user group, new users must join, but among new users, explicit feedback, such as purchase behavior, and implicit feedback, such as click-through, are very sparse.

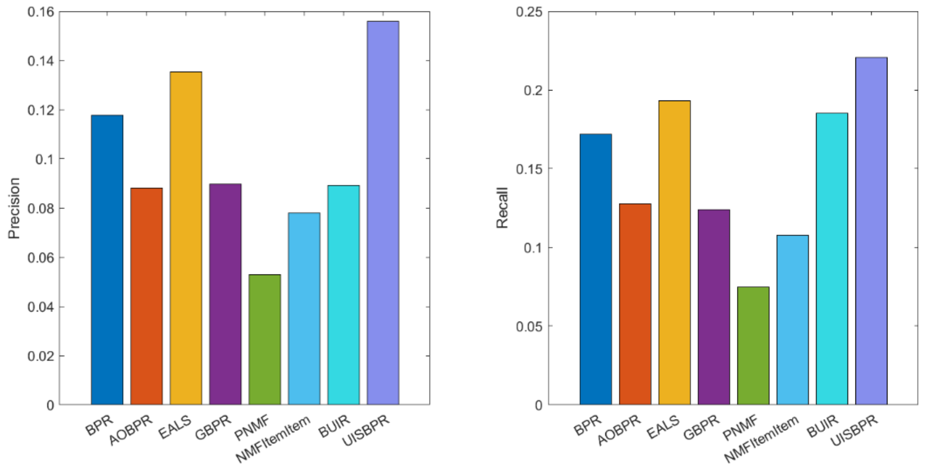
In order to verify the ability of our model to solve cold-start problems, we select users who operated on less than γ to evaluate their ability to solve users' cold-start problems by analyzing the effect of model processing and comparing it with baseline algorithms. Considering the sparseness and scale of different datasets, we set γ to 26 in ML-100K and Epinions and 16 in ML-1M. It is important to note that our cold-start users exist in the entire user-item dataset, so during the training of the model, we put all of the users together into the model for training. Finally, users with fewer than γ interactions are filtered as cold-start users, and the precision and recall of their prediction ranking will be calculated as the recommended accuracy of cold-start. The result is shown in Fig. 9.



(a) Result in Epinions



(b) Result in ML-100K



(c) Result in ML-1M

Fig. 9. Results of different algorithms for cold-start users in different datasets

As shown in the Fig. 9, the accuracy of the cold-start users is reduced due to their lack of data. It shows that BUIR works better for the datasets with large sparsity, and EALS performs well when the dataset is less sparse and not large. Its practical effect in the Movielens 1M is not much lower than BPRS. For BPRS, the similar relationship between similar items and similar users is calculated in advance according to the existing data. By dividing the collection of forecast items into three levels, we can reduce the impact of the scarcity of data to a certain extent.

5 Conclusion and Future Work

To better utilize unobserved items and distinguish potential interest items from negative items, we propose a novel ranking model BPRS, which synthesizes the hidden items from user classification and item classification. Starting from the existing rating data, we explore the interest item set_1 by user similarity from the user aspect and the interest item set_2 by item similarity from the item aspect, and we synthesized both sets to expand the original sparse score set and explore the hidden items of users.

By running different baseline algorithms on the same dataset, we compare the accuracy of our model with the most advanced algorithms, and the results further proved the superiority of our algorithm. At the same time, through repeated experimental verifications, we can clearly find that the algorithm model we designed can solve the user's cold-start problem mentioned above to a great extent. This also can reflect the universality of our algorithm to a certain extent.

In addition, in our future work and research, we hope to further verify the implicit topological relationship that may exist in the actual social network in combination with the graph neural network, and we hope to conduct experiments on real datasets of different sizes, so that we can further solve the user cold-start problem and sparsity problem of the recommender system in the actual situation, enhance the robustness of the recommender system, and improve the accuracy of user personalized recommendation.

6 Acknowledgment

This work has been supported by the National Natural Science Foundation of China under Grant 61872033, the Beijing Nova Program from Beijing Municipal Science Technology Commission under Grant Z201100006820015, the Humanity and Social Science Youth Foundation of Ministry of Education of China under Grant 18YJCZH204.

References

[1] C. Yang, X. Chen, L. Liu, T. Liu, S. Geng, A hybrid movie recommendation method based on social similarity and item attributes, in: Proc. 2018 International Conference on Swarm Intelligence, 2018.
[2] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proc. 2001 international conference on World Wide Web, 2001.
[3] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, T.-S. Chua, Discrete collaborative filtering, in: Proc. 2016 International ACM SIGIR conference on Research and Development in Information Retrieval, 2016.
[4] X. He, H. Zhang, M.-Y. Kan, T.-S. Chua, Fast Matrix Factorization for Online Recommendation with Implicit Feedback, in: Proc. 2016 International ACM SIGIR conference on Research and Development in Information Retrieval, 2016.
[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural Collaborative Filtering, in: Proc. 2017 international conference on world wide web, 2017.
[6] Y. Hu, F. Xiong, S. Pan, X. Xiong, L. Wang, H. Chen, Bayesian personalized ranking based on multiple-layer neighborhoods, Information Sciences 542(2021) 156-176.
[7] Y.-C. Lee, T. Kim, J Choi, X. He, S.-W. Kim, M-BPR: A Novel Approach to Improving BPR for Recommendation with Multi-type Pair-wise Preferences, Information Sciences 547(2021) 255-270.
[8] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian Personalized Ranking from Implicit Feedback, in: Proc. 2009 Conference on Uncertainty in Artificial Intelligence, 2009.
[9] X. Chen, L. Li, W. Pan, Z. Ming, A Survey on Heterogeneous One-class Collaborative Filtering, ACM Transactions on Information Systems (TOIS) 38(4)(2020) 1-54.
[10]W. Pan, L. Chen, GBPR: group preference based Bayesian personalized ranking for one-class collaborative filtering, in: Proc: 2013 International Joint Conference on Artificial Intelligence, 2013.
[11]S. Rendle, C. Freudenthaler, Improving pairwise learning for item recommendation from implicit feedback, in: Proc. 2014 ACM international conference on Web search and data mining, 2014.
[12]T. Zhao, J. McAuley, I. King, Leveraging Social Connections to Improve Personalized Ranking for Collaborative Filtering, in: Proc. 2014 ACM international conference on conference on information and knowledge management, 2014.
[13]W. Pan, L. Chen, Group Bayesian Personalized Ranking with Rich Interactions for One-Class Collaborative Filtering, Neurocomputing 207(2016) 501-510.
[14]H. Qiu, G. Guo, J. Zhang, Z. Sun, H.-T. Nguyen, Y. Liu, TBPR: Trinity Preference based Bayesian Personalized Ranking for Multivariate Implicit Feedback, in: Proc. 2016 Conference on User Modeling Adaptation and Personalization, 2016.
[15]H. Qiu, Y. Liu, G. Guo, Z. Sun, J. Zhang, H.-T. Nguyen, BPRH: Bayesian Personalized Ranking for Heterogeneous Implicit Feedback, Information Sciences 453(2018) 80-98.
[16]Q. Zhang, F. Ren, Double bayesian pairwise learning for one-class collaborative filtering, Knowledge-Based Systems 229(2021) 107339.
[17]J. Feng, Z. Xia, X. Feng, J. Peng, RBPR: A hybrid model for the new user cold start problem in recommender systems, Knowledge-Based Systems, Knowledge-Based Systems 214(2021) 106732.
[18]A. Mnih, R. R. Salakhutdinov, Probabilistic matrix factorization, Advances in neural information processing systems 20(2007).
[19]Y. Wang, P. Wang, Z. Liu, L.-Y. Zhang, A new item similarity based on α -divergence for collaborative filtering in sparse data, Expert systems with applications 166(2021) 114074.
[20]S.-B. Sun, Z.-H. Zhang, X.-L. Dong, H.-R. Zhang, T.-J. Li, L. Zhang, F. Min, Integrating Triangle and Jaccard similarities for recommendation, PloS one 12(8)(2017) e0183570.
[21]M. Ayub, T. Khan, M. Shafqat, A New Model for Co-Ratings Based Similarity in Collaborative Filtering, in: Proc. 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), 2021.
[22]J.D.M. Rennie, N. Srebor, Fast maximum margin matrix factorization for collaborative prediction, in: Proc. 2005 international conference on Machine learning, 2005.

[23]F.M. Harper, J. Konstan, The MovieLens Datasets: History and Context, Acm transactions on interactive intelligent systems (tiis) 5(4)(2015) 1-19.
[24]P. Massa, P. Avesani, Trust-aware recommender systems, in: Proc. 2007 ACM conference on Recommender systems, 2007.
[25]D. Billsus, M. Pazzani, Learning collaborative information filters, in: Proc. 1998 International Conference on Machine Learning, 1998.
[26]K. Ji, Z. Chen, R. Sun, K. Ma, Z. Yuan, G. Xu, GIST: A generative model with individual and subgroup-based topics for group recommendation, Expert Systems with Applications 94(2018) 81-93.
[27]M. Pazzani, D. Billsus, Learning and revising user profiles: The identification of interesting web sites, Machine learning 27(3)(1997) 313-331.
[28]Z. Yuan, Z. Yang, E. Oja, Projective nonnegative matrix factorization: Sparseness, orthogonality, and clustering, Lett (2009) 11-13.
[29]D.D. Lee, H. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401(6755)(1999) 788-791.
[30]S. Kabbur, X. Ning, G. Karypis, Fism: factored item similarity models for top-n recommender systems, in: Proc. 2013 ACM SIGKDD international conference on Knowledge discovery and data mining, 2013.
[31]D. Lee, S.K. Kang, H. Ju, C. Park, H. Yu, Bootstrapping user and item representations for one-class collaborative filtering, in: Proc. 2021 International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021.