Applied Intelligence (2021) 51:6810–6822
https://doi.org/10.1007/s10489-021-02237-1

Iterative rating prediction for neighborhood-based collaborative filtering

Li Zhang[1] · **Zepeng Li**[1,2] · **Xiaohan Sun**[1,2]

Accepted: 25 January 2021 / Published online: 14 February 2021

Abstract

This paper investigates the issue of rating prediction for neighborhood-based collaborative filtering in recommendation systems. A novel rating prediction algorithm, called iterative rating prediction (IRP), is proposed for neighborhood-based collaborative filtering. The main idea behind IRP is neighborhood propagation. To predict ratings of items for target users, IRP relies on not only the rating information of direct neighbors but also that of indirect neighbors with different propagation depth. To implement the idea, IRP iteratively updates the ratings of items for users. The efficiency of the proposed method is examined through extensive experiments. Experimental results demonstrate the superior performance of our method, especially on small-scaled and sparse datasets.

Keywords Collaborative filtering · Neighborhood propagation · Rating prediction · Iteration · Recommender systems

1 Introduction

Nowadays, the E-commerce and social networking websites are very active in our daily life [1–3]. For the fast growth in the World Wide Web (WWW), it is not easy for users to find appropriate items from WWW quickly. Fortunately, recommender systems (RSs) can assist users in handling the vast quality of information, which can provide an accurate recommendation of products [4–7]. E-commerce websites, like Netfix.com and Amazon.com, personalize recommendations for users and show them products related to their interests and preferences. In such a situation, RS can not only save the valuable time for users but also increase profitability of the business by increasing sales at online stores. The websites related to tourism [8], movies [9], music [10], and news [11], and restaurants can recommend items to users based on their past history [12].

The family tree of RSs is shown in Fig. 1 [13]. There are three main categories of RSs' techniques, including collaborative filtering (CF), content-based filtering, and hybrid filtering [12]. Content-based filtering algorithms try to analyze the profile of a target user and the profile of a target item based on the user's historical behavior, and then determine whether or not to recommend the target item to the target user. However, it is hard to analyze profiles in many applications such as multimedia data. CF algorithms are the most successful and widely used in RSs [14–16]. The assumption in CF is that if some users have similar interesting items up to now, these users would have similar interests in future. Generally, CF is domain independent and more accurate than content-based filtering. There are two main branches for recommending items in CF: memory-based and model-based algorithms. The hybrid filtering methods combine the approaches of collaborative filtering and content-based filtering in different ways. This paper focuses on memory-based algorithms of CF.

Supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No. 19KJA550002, the Six Talent Peak Project of Jiangsu Province of China under Grant No. XYDXX-054, and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

✉ Li Zhang
zhangliml@suda.edu.cn

Zepeng Li
zpli520@stu.suda.edu.cn

Xiaohan Sun
20195227090@stu.suda.edu.cn

[1] School of Computer Science and Technology, Joint International Research Laboratory of Machine Learning and Neuromorphic Computing, Soochow University, Suzhou 215006, Jiangsu, China

[2] Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, Jiangsu, China

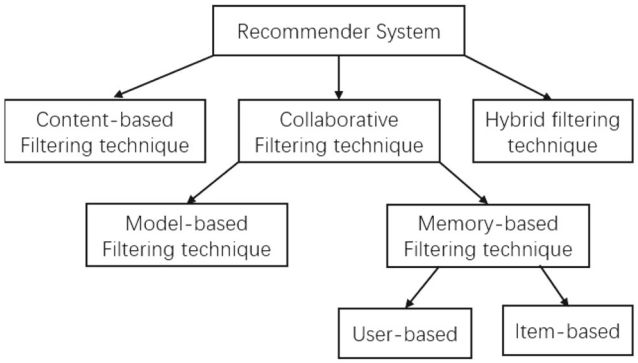

Fig. 1 Family tree of RSs

Memory-based collaborative filtering algorithms, also referred to as neighborhood-based collaborative filtering (NCF) ones, are simple and intuitive and can provide an immediate response to a new user after receiving upon his/her feedback. NCF depends on a simple intuition that an active user may have preference behaviors on some items if these items are appreciated by a set of similar users (neighbors). NCF involves three main steps: (1) Calculating a similarity value between two users or items by using a similarity measurement; (2) Finding the nearest neighbors of an active user or item according to the similarity value; (3) Generating a predicted rating value for an item according to the rating prediction algorithm. Because previous researches have indicated that the similarity measurement plays an important role in recommendation performance of NCF, a lot of work about NCF have been done on similarity measurements [15, 17–22]. The first two steps of NCF are out of the scope of this paper, so we do not discuss them further.

The rating prediction algorithm in the last step of NCF is another factor considered by researchers. The average prediction (Ave-pred) method [23] is the first method for rating prediction. Ave-pred is simple and takes only the average scores of neighbor users as ratings. Thus, the recommendation performance of Ave-pred is not satisfactory owing to ignoring the similarity between different neighbors and the target user in the prediction stage. Similarity-based prediction (Sim-pred) method [24] predicts the rating of a target item based on the similarity between the target user and its neighbors, which is a commonly used method in NCF. However, Sim-pred simply uses the rating information of neighbors such that the prediction performance is related to the number of neighbors. Moreover, Sim-pred cannot screen out high-quality neighbors, its computational cost is relatively high. Besides, in the domains of E-commerce and social networks, the rating matrix is very sparse, which makes the co-rated items of neighbor users extremely rare and would lower the prediction accuracy. To remedy it, Zhang and Pu [25] proposed a recursive prediction algorithm (RPA) that relaxes the issue of the data sparseness. However, RPA assigns ratings to items, but never backtracks and modifies them again. As a result, the ratings predicted by RPA may be partly incomplete. Kumar et al. [26] proposed a new user rating prediction (URP) algorithm that uses a fixed similarity algorithm. URP cannot transplant other commonly used similarity measurements, otherwise the recommendation performance of URP cannot be guaranteed.

To overcome the data sparseness problem and provide more accurate ratings, a novel iterative rating prediction (IRP) method is presented to predict the ratings of items for target users. Similar to RPA, IRP breaks the constraint of relying on only direct neighbors and tries to utilize more rating information. Our contributions are as follows:

- A general prediction method (IRP) is proposed for estimate the ratings of users on items in RS. IRP is designed independently of similarity measurement, so it can be combined with existing similarity measurements, such as Pearson correlation coefficient (PCC).
- By using the propagation of nearest neighbors, IRP utilizes more rating information from indirect neighbors with different propagation depth. In this way, IRP efficiently extends the set of nearest neighbors.
- IRP iteratively backtracks and modifies the predicted ratings. In the iterative process, we define the propagation reliability for each entry in the user-item rating matrix, which can show the reliability of each rating in matrix and the propagation depth. In our method, the greater the propagation depth or iteration, the smaller the propagation reliability is.

The rest of the paper is organized as follows. In Section 2, PCC and the related work on rating prediction algorithms is reviewed. Section 3 proposes the new rating prediction method. In Section 4, experimental results on real datasets are reported and analyzed. Finally, Section 5 concludes this paper.

2 Related work

In this section, we describe the similarity measurement (PCC) and the rating prediction algorithms used in NCF. In fact, our method mainly deals with the prediction stage; thus, our method can be combined with arbitrary similarity measurements, including traditional and new hybrid ones. In addition, our method is a rating prediction method. For comparison, we also introduce commonly used prediction algorithms in NCF.

First, we provide some notations that are used in this paper. Let $U = \{u_1, u_2, \cdots, u_m\}$ and $T = \{t_1, t_2, \cdots, t_n\}$

be the sets of users and items, respectively, where m is the number of users and n is the number of items. The user-item rating matrix can be expressed as $\mathbf{R} \in \mathbb{R}^{m \times n}$, where the ith row and jth column element r_{ij} is the rating score made by the ith user on the jth item. Table 1 lists related notations.

Assume that $u_v \in U$ is the target user and t_j is the target item. In the following, we discuss four methods for calculating the prediction value of the user u_v on the item t_j.

2.1 Similarity measurement

Many similarity measurements have been proposed and used in NCF for RSs, including traditional similarity measurements (cosine similarity, Pearson correlation coefficient [17]) and new hybrid ones [18–22]. Although new hybrid similarity measurements can indeed improve the prediction accuracy of RSs, these methods suffer from the issue of high computational complexity. For simplicity, we use one of simple traditional measurements: PCC.

PCC is often used to compute linear correlation between a pair of objects. The similarity between users u_v and u_i can be defined as follows:

$$sim(u_v, u_i) = \frac{\sum_{t_j \in T_{vi}} (r_{vj} - \overline{r}_v)(r_{ij} - \overline{r}_i)}{\sqrt{\sum_{t_j \in T_{vi}} (r_{vj} - \overline{r}_v)^2} \sqrt{\sum_{t_j \in T_{vi}} (r_{ij} - \overline{r}_i)^2}} \tag{1}$$

where T_{vi} represents the set of items rated by users u_v and u_i together, r_{vj} indicates the rating value of user u_v on item t_j, and $\overline{r}_v$ is the average rate of user u_v.

2.2 Prediction algorithms

2.2.1 Ave-pred

The most simple method for rating prediction is the maximizing average satisfaction method, or Ave-pred, which was proposed in [23]. Ave-pred can use arbitrary similarity measurements to calculate the correlation between users. However, when predicting the rating of the user u_v on the item t_j, Ave-pred does not consider the distance relationship between the target user u_v and its neighbors but assigns the same weight to all neighbors.

In Ave-pred, the calculation formula for prediction rating is as follows:

$$\widehat{r}_{vj} = \frac{1}{|NU_v|} \sum_{u_i \in NU_v} r_{ij} \tag{2}$$

where NU_v represents the set of K nearest neighbors of the target user u_v, r_{ij} means the rating of user u_i on item t_j and $|\cdot|$ denotes the cardinality of a set $\cdot$.

2.2.2 Sim-pred

On the basis of Ave-pred, Jannach proposed Sim-pred [24]. Similar to Ave-pred, Sim-pred needs to find the set NU_v of similar users for u_v. Unlike to Ave-pred, Sim-pred takes into account similarity between users besides the average ratings of users. The average score of users can represent their rating preference. As a result, the prediction is made as a weighted average of neighbors. Although Sim-pred has been widely applied to NCF methods, its recommendation performance is related to the number of neighbors, and the computational complexity is relatively high.

In Sim-pred, the prediction formula can be expressed as

$$\widehat{r}_{vj} = \overline{r}_v + \frac{\sum_{u_i \in NU_v} sim(u_v, u_i)(r_{ij} - \overline{r}_i)}{\sum_{u_i \in NU_v} |sim(u_v, u_i)|} \tag{3}$$

where $sim(u_v, u_i)$ is a function of computing the similarity between u_v and u_i, r_{ij} represents the rating value of item t_j given by user u_i, $\overline{r}_v$ is the average rating score of user u_v, and $|\cdot|$ is the absolute value of a scalar $\cdot$.

2.2.3 URP

Kumar et al. [26] designed a new user rating prediction (URP) algorithm for predicting ratings of items.

In URP, there is an assumption that if two users rate similar types of items and give similar ratings to these items, then these users are similar to each other.

The similarity between the target user u_v and user u_i can be calculated by

$$sim(u_v, u_i) = \sum_{t_p \in T_{vi}} r_h - \left| r_{vp} - r_{ip} \right| \tag{4}$$

Table 1 Description of notations

Notation	Description
r_h	The highest rating value
n	The maximum of items
$\overline{r}_i$	The average rating score of user u_i
r_{ij}	The rating of user u_i on item t_j
T_{vi}	The collection of items co-rated by user u_v and user u_i
NU_v	The set of K nearest neighbors for the user u_v

where r_h is the highest rating among all ratings, r_{vp} is the rating of user u_v on item t_p, and T_{vi} is the set of items rated by both users u_v and u_i. If $sim(u_v, u_i)$ is greater than a given threshold, then user u_i is included in the list NU_v. The prediction formula of URP is as follows

$$\widehat{r}_{vj} = \frac{\sum_{u_i \in NU_v} r_{ij}}{|NU_v|} + \frac{\sum_{u_i \in NU_v} sim(u_v, u_i)}{n\,|NU_v|\,r_h} \tag{5}$$

where n is the number of total items.

Although URP is very well implemented, the prediction process is not suitable for other similarity calculation methods except for (4). Even so, the similarity measurement (4) is relatively simple and does not consider the user's rating preference. Besides, the similarity measurement greatly depends on the co-rated items, the calculated similarity would be biased if there are very few common rated items owing to data sparseness.

2.2.4 RPA

Zhang and Pu [25] proposed a recursive prediction algorithm that relaxes the constraint that nearest neighbors must have rated the target item. If one nearest neighbor of the target user has not rated the given item yet, then RPA recursively estimates the rating value of this neighbor based on the neighbor's nearest neighbors and joins the estimated rating value in the prediction of the target user on the given item. In this way, RPA makes more information contribute to the prediction process.

The predicted rating of RPA can be represented as the following:

$$\widehat{r}_{vj} = \overline{r}_v + \frac{\sum_{u_i \in NU_v} w(u_i, t_j) sim(u_v, u_i)(r_{ij} - \overline{r}_i)}{\sum_{u_i \in NU_v} \left| w(u_i, t_j) sim(u_v, u_i) \right|} \tag{6}$$

where $\overline{r}_v$ represents the average rating of u_v, NU_v is the set of nearest neighbors of user u_v, r_{ij} means the score of item t_j given by user u_i, and $w(u_i, t_j)$ is the weight value of user u_v on item t_j and defined by

$$w(u_i, t_j) = \begin{cases} 1, & \text{if } r_{ij} \text{ is given} \\ \lambda, & \text{otherwise} \end{cases}$$

with the weight threshold $\lambda \in [0, 1]$.

3 Iterative rating prediction algorithm

In Section 2, we discuss four prediction methods for NCF, all of which are direct and intuitive. Because Ave-pred, Sim-pred and URP adopt only the rating information of neighbor users who have rated the given item, these methods would have bad prediction performance in the case of data sparsity. Although RPA could avoid the issue of sparse data, RPA has its own limitations: (1) RPA never backtracks and modifies the predicted ratings; (2) It is hard to determine the weight value $w(u_i, t_j)$ if r_{ij} is not given. On the basis of neighborhood propagation and the iteration idea from RPA, this section presents an iterative rating prediction algorithm.

It is well known that users may rate only on a small amount of items owing to a huge number of items. Therefore, there are many entries missed in the user-item rating matrix $\mathbf{R}$. Without loss of generality, let $r_{ij} > 0$ for all rated items and $r_{ij} = 0$ for unrated ones. In other words, if $r_{ij} = 0$, we thought that r_{ij} is not given by user u_i. The goal of IRP is to iteratively update the predicted rating $\widehat{r}_{ij}$ if r_{ij} is not given in the original rating matrix.

3.1 Neighbor propagation

Before we discuss IRP, we first describe some related concepts. Given the user set U and the item set T, the neighborhood relation R on U and T can be described as:

$$R = \{(u_v, u_i) | u_i \in NU_v \subset U, u_v, u_i \in U\} \tag{7}$$

where NU_v is the set of K nearest neighbors of u_v; that is, $|NU_v| = K$. The properties of neighborhood relation can be found in Theorem 1.

Theorem 1 *Let U be a user set and T be an item set. If R is a neighborhood relation on U and T, then*

1. *R is reflexive;*
2. *R is not symmetric;*
3. *R is not transitive.*

The proof of Theorem 1 is shown in Appendix A. The system (U, R) can provide a directed graph G, where users are taken as vertexes, and neighborhood relations are edges. Usually, G can be represented by an adjacent matrix. Let G be a directed graph induced by a user set U and its neighborhood relation R. The adjacent matrix $\mathbf{A}$ of G can be defined as

$$a_{vi} = \begin{cases} 1, & if\ (u_v, u_i) \in R \\ 0, & otherwise \end{cases} \tag{8}$$

where a_{vi} is the v-th row and i-th column entry of $\mathbf{A}$.

The adjacent matrix $\mathbf{A}$ can describe the direct neighbor relationship. For user u_v, its direct neighbor is u_i if and only if $a_{vi} = 1$. In the directed graph G, u_v is directly connected and pointed to u_i. It is possible that u_v is indirectly connected and pointed to u_i in G. In this case, u_v is said to be propagated to u_i and u_i is called an indirected neighbor of u_v. To describe the indirect neighbor relationship, we first define the concept of propagation formally.

Definition 1 Let G be a directed graph induced by the user set U and its neighborhood relation R, and $\mathbf{A}$ be the adjacent matrix of G.

1. Let $\mathbf{A}^1 = \mathbf{A}$. If $a_{vi}^{(1)} = 1$ in $\mathbf{A}^1$, then u_v can be propagated to u_i.
2. Let $\mathbf{A}^2 = \mathbf{A} \otimes \mathbf{A}$, where $\otimes$ denotes the logical multiplication of matrices. If $a_{vi}^{(2)} = 1$ in $\mathbf{A}^2$, then u_v can be propagated to u_i.
3. Let $\mathbf{A}^q = \mathbf{A}^{q-1} \otimes \mathbf{A}$ with $q \in \mathbb{N}^+$. If $a_{vi}^{(q)} = 1$ in $\mathbf{A}^q$, then u_v can be propagated to u_i.

From Definition 1, we can see that the neighborhood propagation can be reflected by the relation composition of $\mathbf{A}$ itself. In graph theory, $a_{vi}^{(q)} = 1$ in $\mathbf{A}^q$ also denotes that there exists at least one path with length q from u_v to u_i. In this way, we define the concept of indirect neighbors.

Definition 2 Let U be the user set, and $\mathbf{A}$ be the adjacent matrix of G constructed on U. For $u_v \in U$, u_i is its indirect neighbor if and only if $a_{vi} = 0$ and u_v can be propagated to u_i.

In the following, we illustrate direct neighbors, indirect ones and neighborhood propagation.

Example 1 Assume that there are five users and eight items. Let $U = \{u_1, u_2, u_3, u_4, u_5\}$ and $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$. The original rating information is given in Table 2, where rating scores are between 1 to 5 and "-" means that users do not rate items. Let $K = 2$. We analyze direct neighbors, indirect ones and neighborhood propagation based on sets U and T.

First, we represent the user-item rating matrix as follows:

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 0 & 5 & 0 & 4 & 3 & 0 \\ 0 & 3 & 2 & 3 & 0 & 0 & 0 & 2 \\ 4 & 0 & 4 & 0 & 1 & 0 & 2 & 0 \\ 0 & 2 & 4 & 2 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 5 & 0 & 3 \end{bmatrix} \tag{9}$$

Second, we use PCC to measure the similarity between five users. The similarity matrix $\mathbf{S}$ has the form:

$$\mathbf{S} = \begin{bmatrix} \infty & 1 & 1 & -0.707 & 0 \\ 1 & \infty & -1 & -1 & 0 \\ 1 & -1 & \infty & 1 & 0.164 \\ -0.707 & -1 & 1 & \infty & 1 \\ 0 & 0 & 0.164 & 1.0 & \infty \end{bmatrix} \tag{10}$$

where the similarity between u_i and itself is set to ∞.

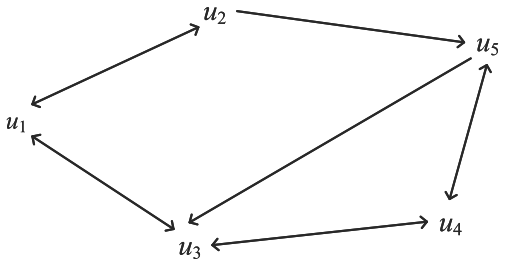

Fig. 2 Directed graph of Example 1

Then, we show the directed graph G on U in Fig. 2. From the Fig. 2, we can observe the propagation of users. Without loss of generality, we takes u_1 as the target user. The set of nearest neighbor users of u_1 is $NU_1 = \{u_2, u_3\}$, which can be directly observed from both (10) and Fig. 2. In other words, both u_2 and u_3 are direct neighbors of u_1. Also, we can see that u_1 and u_5 are direct neighbors of u_2; thus, u_5 should be one of indirect neighbors of u_1 according to Definition 2. Similarly, u_4 is also indirect neighbors of u_1. From Fig. 2, we can see that u_1 can be propagated to both u_4 and u_5. In terms of Definition 1, we calculate $\mathbf{A}^2$ and have

$$\mathbf{A}^2 = \mathbf{A} \otimes \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

which indicates that $a_{14} = 1$ and $a_{15} = 1$. In other words, u_4 and u_5 are indirect neighbors of u_1. We can continue to find indirect neighbors for u_1 by $\mathbf{A}^3$, $\mathbf{A}^4$ and so on.

3.2 Prediction process

It is well known that the user-item matrix may be very sparse. When we try to recommend a target item to a target user, IRP predicts its rating value on this target item using scores given by its direct neighbors. If some or all nearest neighbors of the target user have not rated the target item, IRP first ignores these neighbors until they have ratings. The scores of these neighbors could be estimated from the rating information of their own direct neighbors. In this

Table 2 Rating information in Example 1

User\Item	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
u_1	–	–	–	5	–	4	3	–
u_2	–	3	2	3	–	–	–	2
u_3	4	–	4	–	1	–	2	–
u_4	–	2	4	2	–	4	–	–
u_5	2	–	–	–	2	5	–	3

way, the rating information is propagated among in the neighborhood.

Let p be the current iteration and $\widetilde{\mathbf{R}}^{(p)}$ be the rating matrix in the p-th iteration. In each iteration, IRP utilizes the rating information provided by the previous iteration to update the rating matrix. The element of v-th row and j-th column in $\widetilde{\mathbf{R}}^{(p)}$ is denoted as $\widetilde{r}_{vj}^{(p)}$ that can be updated as:

$$\widetilde{r}_{vj}^{(p)} = \begin{cases} r_{vj}, & if\ r_{vj} > 0 \\ \bar{r}_v^{(p-1)} + \triangle\widetilde{r}_{vj}^{(p)}, & otherwise \end{cases} \tag{11}$$

where r_{vj} is the original rating of user u_v on item t_j, $\bar{r}_v^{(p-1)}$ is the average rating score of user u_v in the $(p-1)$-th iteration, and $\triangle\widetilde{r}_{vj}^{(p)}$ is the predicted rating increment in the p-th iteration. The prediction formulation for $\triangle\widetilde{r}_{vj}^{(p)}$ in IRP can be described as

$$\triangle\widetilde{r}_{vj}^{(p)} = \frac{\sum_{u_i \in NU_v} w_{ij}^{(p-1)} sim(u_v, u_i) \left(\widetilde{r}_{ij}^{(p-1)} - \bar{r}_i^{(p-1)}\right)}{\sum_{u_i \in NU_v} |sim(u_v, u_i)|} \tag{12}$$

where $w_{ij}^{(p-1)}$ is the propagation reliability of the rating $\widetilde{r}_{ij}^{(p-1)}$ in the $(p-1)$-th iteration, and $\widetilde{r}_{ij}^{(p-1)}$ is the score of user u_i on item t_j in the $(p-1)$-th iteration.

Equation (11) implies that the predicted ratings would be updated only for items that users do not rate on. According to (12), $\triangle\widetilde{r}_{vj}^{(p)}$ depends on the rating information of direct neighbors of u_v on t_j. It is possible that all neighbors of u_v do not rate item t_j. In this case, $\triangle\widetilde{r}_{vj}^{(p)}$ is equal to zero, and $\widetilde{r}_{vj}^{(p)}$ is equal to the average rating of u_v at this iteration. However, $\triangle\widetilde{r}_{vj}^{(p)}$ can be changed as long as one of indirected neighbors of u_v has rated item t_j such that $\widetilde{r}_{vj}^{(p)}$ could be modified accordingly. Therefore, our method can iteratively backtrack the predicted ratings.

In (12), it requires determine the propagation reliability with iterations. Let $\mathbf{W}^{(p)}$ be the propagation reliability matrix in the p-th iteration. According to the description above, IRP dynamically updates two matrices: the predicted rating matrix $\widetilde{\mathbf{R}}^{(p)}$ and the propagation reliability matrix $\mathbf{W}^{(p)}$. Naturally, the initial rating matrix $\widetilde{\mathbf{R}}^{(0)}$ is the original rating matrix $\mathbf{R}$ when $p = 0$. For $p > 0$, we follow (11) and (12) to update $\widetilde{\mathbf{R}}^{(p)}$ based on both $\widetilde{\mathbf{R}}^{(p-1)}$ and $\mathbf{W}^{(p-1)}$. The initialization of $\mathbf{W}^{(0)}$ has the following form:

$$w_{ij}^{(0)} = \begin{cases} 1, & if\ r_{ij} > 0 \\ 0, & otherwise \end{cases} \tag{13}$$

On the basis of $\mathbf{W}^{(p-1)}$, $\widetilde{\mathbf{R}}^{(p-1)}$ and $\widetilde{\mathbf{R}}^{(p)}$, the updating formulation of $\mathbf{W}^{(p)}$ can be expressed as:

$$w_{ij}^{(p)} = \begin{cases} \lambda^p, & if\ \triangle\widetilde{r}_{ij}^{(p-1)} = 0 \wedge \triangle\widetilde{r}_{ij}^{(p)} \neq 0 \\ w_{ij}^{(p-1)}, & otherwise \end{cases} \tag{14}$$

where the weight factor $0 \leq \lambda \leq 1$. The formula (14) indicates that the update of propagation reliability depends on the predicted increment both the p-th iteration and the $(p-1)$-th iteration. Moreover, the propagation reliability on a rating r_{ij} indicates that the reliability of this rating and the propagation depth. For example, $w_{ij}^{(p)} = 1$ means that $\widetilde{r}_{ij}^{(p)}$ is directly given by user u_i; thus, this rating is very reliable and has a propagation depth of 0. While $w_{ij}^{(p)} = \lambda^p$ means that $\widetilde{r}_{ij}^{(p)}$ is estimated from neighbors of u_i; thus, the reliability of this rating is discounted and has a propagation depth of p. The greater the propagation depth is, the larger the neighborhood.

3.3 Algorithm description and analysis

In fact, IRP can not only provide a predicted rating score for a given target user on a given item, but also fill the whole rating matrix with predicted ratings. We summary the procedure of filling the rating matrix using IRP in Algorithm 1. Basically, IRP has four steps. Step 1 is to initialize two matrices. In Step 2, we need to know the similarity between any two users in the user set U. Here, the similarity measurement available can be applied to our algorithm. In experiments, we adopt PCC (1) as the similarity measurement, which is simple and efficient. Step 3 is the core of IRP, which is to alternately update $\widetilde{\mathbf{R}}^{(p)}$ and $\mathbf{W}^{(p)}$. The main iteration would be stopped when the number of iteration is greater than the maximum number of iterations allowed, or $\|\widetilde{\mathbf{R}}^{(p)} - \widetilde{\mathbf{R}}^{(p-1)}\|_F \leq \theta$ holds true, where $\|\cdot\|$ denotes the Frobenius norm of a matrix. Finally, Step 4 returns the new rating matrix $\widetilde{\mathbf{R}}^*$.

According to Algorithm 1, we discuss the computational complexity of IRP. Among four steps of IRP, the first step is initialization and the fourth step is return; thus, these two steps are neglected. Step 2 calculating the similarity between users has the computational complexity of $O(m^2 n)$ in theory, where m is the number of users and n is the number of items. It is worth noting that the number of ratings given by users is far less than n, so the computational complexity of this step should be less than $O(m^2 n)$. In the iteration process, there are at most $MaxIter$ iterations. Each iteration uses K nearest neighbors to locally update the values in the rating matrix. Both the rating matrix and the propagation reliability matrix have the size of $m \times n$; thus,

the number of updated entries is much less than mn. Step 3 has the computational complexity of $O((MaxIter)Kmn)$, where $MaxIter \ll m$ and $K \ll m$. In summary, the computational complexity of IRP is $O(m^2n)$.

Algorithm 1 Iterative rating prediction.

Input: U: the set of users; T: the set of items; $\mathbf{R}$: the user-item rating matrix; λ: the weight factor; K: the number of neighbors; θ: the threshold of stop updating; $MaxIter$: the maximum number of iterations allowed.

Output: The updated rating matrix $\widetilde{\mathbf{R}}^*$.

begin

1. Initialization: the rating matrix $\widetilde{\mathbf{R}}^{(0)} = \mathbf{R}$, the propagation matrix $\mathbf{W}^{(0)}$ using (13), and let $p = 1$;
2. Calculate similarities between users $sim(u_v, u_i)$ where $u_v, u_i \in U$, and generate the set of direct neighbors for each user $NU_i, u_i \in U$;
3. **while** $p \leq MaxIter$ **do**
   - Update $\widetilde{\mathbf{R}}^{(p)}$ using (11) and (12);
   - Update $\mathbf{W}^{(p)}$ using (14);
   - **if** $\|\widetilde{\mathbf{R}}^{(p)} - \widetilde{\mathbf{R}}^{(p-1)}\|_F \leq \theta$ **then**
     - **break**;
   - **else**
     - Let $p = p + 1$;
4. **return** $\widetilde{\mathbf{R}}^*$, where $\widetilde{\mathbf{R}}^* = \widetilde{\mathbf{R}}^{(p)}$.

end

In the following, we continue Example 1 to comprehend the procedure of IRP.

Example 2 (Continued from Example 1) In the user-item rating matrix (9), the unrated items have ratings of 0. Here, the task is to assign predicted ratings for those unrated items in terms of IRP. According to Algorithm 1, we need to set some parameters in advance. Let $MaxIter = 10$, $\lambda = 0.9$, and $\theta = 0.1$. The similarity matrix has been given by (10) in Example 1.

We first initialize the predicted user-item rating matrix by

$$\mathbf{R}^{(0)} = \mathbf{R} = \begin{bmatrix} 0 & 0 & 0 & 5 & 0 & 4 & 3 & 0 \\ 0 & 3 & 2 & 3 & 0 & 0 & 0 & 2 \\ 4 & 0 & 4 & 0 & 1 & 0 & 2 & 0 \\ 0 & 2 & 4 & 2 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 5 & 0 & 3 \end{bmatrix}$$

and the propagation reliability matrix by

$$\mathbf{W}^{(0)} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Next, we begin to alternately update the rating matrix and the propagation reliability matrix. For $p = 1$, we have

$$\widetilde{\mathbf{R}}^{(1)} = \begin{bmatrix} \mathbf{5} & \mathbf{5} & \mathbf{4} & 5 & \mathbf{2} & 4 & 3 & \mathbf{4} \\ 0 & 3 & 2 & 3 & 0 & \mathbf{3} & \mathbf{2} & 2 \\ 4 & \mathbf{2} & 4 & \mathbf{3} & 1 & \mathbf{3} & 2 & 0 \\ \mathbf{3} & 2 & 4 & 2 & \mathbf{2} & 4 & \mathbf{2} & \mathbf{3} \\ 2 & \mathbf{2} & \mathbf{4} & \mathbf{2} & 2 & 5 & \mathbf{2} & 3 \end{bmatrix}$$

and

$$\mathbf{W}^{(1)} = \begin{bmatrix} \mathbf{0.9} & \mathbf{0.9} & \mathbf{0.9} & 1.0 & \mathbf{0.9} & 1.0 & 1.0\text{‘} & \mathbf{0.9} \\ 0.0 & 1.0 & 1.0 & 1.0 & 0.0 & \mathbf{0.9} & \mathbf{0.9} & 1.0 \\ 1.0 & \mathbf{0.9} & 1.0 & \mathbf{0.9} & 1.0 & \mathbf{0.9} & 1.0 & 0.0 \\ \mathbf{0.9} & 1.0 & 1.0 & 1.0 & \mathbf{0.9} & 1.0 & \mathbf{0.9} & \mathbf{0.9} \\ 1.0 & \mathbf{0.9} & \mathbf{0.9} & \mathbf{0.9} & 1.0 & 1.0 & \mathbf{0.9} & 1.0 \end{bmatrix}$$

where the updated entries are in bold type. The stop conditions are not satisfied. For $p = 2$, we have

$$\widetilde{\mathbf{R}}^{(2)} = \begin{bmatrix} 5 & \mathbf{4} & 4 & 5 & 2 & 4 & 3 & 4 \\ \mathbf{4} & 3 & 2 & 3 & \mathbf{1} & \mathbf{2} & \mathbf{1} & 2 \\ 4 & 2 & 4 & 3 & 1 & 3 & 2 & \mathbf{3} \\ 3 & 2 & 4 & 2 & 2 & 4 & 2 & 3 \\ 2 & 2 & 4 & 2 & 2 & 5 & 2 & 3 \end{bmatrix}$$

and

$$\mathbf{W}^{(2)} = \begin{bmatrix} 0.90 & 0.90 & 0.90 & 1.00 & 0.90 & 1.00 & 1.00 & 0.90 \\ \mathbf{0.81} & 1.00 & 1.00 & 1.00 & \mathbf{0.81} & 0.90 & 0.90 & 1.00 \\ 1.00 & 0.90 & 1.00 & 0.90 & 1.00 & 0.90 & 1.00 & \mathbf{0.81} \\ 0.90 & 1.00 & 1.00 & 1.00 & 0.90 & 1.00 & 0.90 & 0.90 \\ 1.00 & 0.90 & 0.90 & 0.90 & 1.00 & 1.00 & 0.90 & 1.00 \end{bmatrix}$$

The stop conditions are still not satisfied. In addition, since all elements in $\widetilde{\mathbf{R}}^{(2)}$ are greater than zero, the propagation reliability matrix would not be updated again. In other words, $\mathbf{W}^{(p)} = \mathbf{W}^{(2)}$ for $p > 2$. In the following iterations, we have

$$\widetilde{\mathbf{R}}^{(3)} = \begin{bmatrix} 5 & 4 & 4 & 5 & 2 & 4 & 3 & 4 \\ 4 & 3 & 2 & 3 & 1 & 2 & 1 & 2 \\ 4 & \mathbf{3} & 4 & 3 & 1 & 3 & 2 & 3 \\ 3 & 2 & 4 & 2 & \mathbf{1} & 4 & 2 & 3 \\ 2 & 2 & 4 & 2 & 2 & 5 & 2 & 3 \end{bmatrix}, \widetilde{\mathbf{R}}^{(4)} = \begin{bmatrix} 5 & 4 & 4 & 5 & 2 & 4 & 3 & 4 \\ \mathbf{3} & 3 & 2 & 3 & 1 & 2 & 1 & 2 \\ 4 & 3 & 4 & 3 & 1 & 4 & 2 & 3 \\ 3 & 2 & 4 & 2 & 1 & 4 & 2 & 3 \\ 2 & 2 & 4 & 2 & 2 & 5 & 2 & 3 \end{bmatrix},$$

$$\text{and } \widetilde{\mathbf{R}}^{(5)} = \begin{bmatrix} 5 & 4 & 4 & 5 & 2 & 4 & 3 & 4 \\ 3 & 3 & 2 & 3 & 1 & 2 & 1 & 2 \\ 4 & 3 & 4 & 3 & 1 & 4 & 2 & 3 \\ 3 & 2 & 4 & 2 & 1 & 4 & 2 & 3 \\ 2 & 2 & 4 & 2 & 2 & 5 & 2 & 3 \end{bmatrix}$$

When $p = 5$, the condition $\|\widetilde{\mathbf{R}}^{(5)} - \widetilde{\mathbf{R}}^{(4)}\| < \theta$ is satisfied. Thus, IRP stops. The final rating matrix is $\widetilde{\mathbf{R}}^{(5)}$.

From Example 2, we see that values in the predicted rating matrix are iteratively updated. Once the rating information of neighbor users of u_v on item t_j is changed in the current iteration, r_{vj} would be updated in the next iteration. After the fourth iteration, the rating scores

Table 3 Description of datasets used in the experiments

Dataset	Purpose	#User	#Item	#Rating	Sparsity	Rating domain
ML-Latest_Small	Movie	671	9066	100,004	98.3%	$\{0.5, 1, \cdots, 5\}$
ML-100K	Movie	943	1682	100,000	93.7%	$\{1, 2, \cdots, 5\}$
FilmTrust	Movie	1508	2071	35,497	98.8%	$\{0.5, 1, \cdots, 4\}$
Netflix	Movie	5000	16,295	1,066,148	98.7%	$\{1, 2, \cdots, 5\}$

are stabilized and no longer updated, which shows the convergence of the IRP algorithm.

4 Experiments and analysis

To validate the performance of our method, we perform experiments on four public datasets: ML-Latest-Small [27], ML-100k [27], FilmTrust [28] and Netflix[1] which have been frequently used in literatures to test recommendation algorithms. In this section, we first describe these datasets and performance evaluation metrics, and then report experimental results of our approach and other compared ones. All numerical experiments are performed on a personal computer with an Intel Core I5 processor with 8 GB RAM. This computer runs Windows 7, with Matlab R2012b.

4.1 Datasets

Both ML-Latest-Small and ML-100k are from MovieLens that was collected by the GroupLens Research Project at the University of Minnesota [27]. A brief description of these datasets is given in Table 3.

- **ML-Latest-Small**: This dataset describes a five-star rating that movies were rated on a floating point scale of 0.5 (bad) to 5 (excellent) with scale of 0.5 and contains 100,004 ratings across 9,066 movies. The dataset was created by 671 users from January 09, 1995 to October 16, 2016. Users were selected at random for inclusion. All selected users had rated at least 20 movies. The sparsity of ML-Latest-Small is 98.3%.
- **ML-100k**: The dataset was collected through the MovieLens website (movielens.umn.edu) and has been cleaned up, where users who had less than 20 ratings or did not have complete demographic information were removed from this dataset. This dataset contains 100,000 ratings provided by 943 users for 1,682 movies. Movies were rated on an integer scale 1 (bad) to 5 (excellent). The sparsity of this matrix is 93.7%.
- **FilmTrust**: It is not a pre-packaged dataset and is a small dataset crawled from the entire FilmTrust website in June 2011, which includes 35,497 ratings provided by 1,508 users for 2,071 movies. Movies were rated on a floating point scale 0.5 (bad) to 4 (excellent) with scale of 0.5. The sparsity of FilmTrust reaches 98.8%. Note that this dataset is very sparse and imbalanced in that one user might have rated one item and another might have rated dozens of items (and same is true for items as well).
- **Netflix**: The dataset brings movies together that have been rated from 1999 to 2005, including 480,189 users, 17,770 movies and more than 100 million rating data. Users can choose different rating values from 1 to 5 to rate a movie. Besides, considering the size of the dataset, we randomly select 5000 users who gave about 1,066,148 rating data for 16,295 items. The sparsity of Netflix is 98.7%.

To evaluate the effectiveness of the proposed scheme, each dataset is categorized into five subsets by applying five-fold cross-validation, which follows the method described in [29] and [30]. In each trial, four subsets are used for training and the remaining for test. As a result, we report the average performance of five trials.

4.2 Experimental setting

Our proposed IRP method is a kind of prediction rating algorithms, so we compare it with the existing rating prediction algorithms mentioned in the related work, including Sim-pred [24], Ave-pred [23], URP [26], and RPA [25]. Besides, we also compare with other NCF algorithms, such as Users' tree accessed on subspace (UTAOS) [31], Neighbor users by subspace clustering on collaborative filtering (NUSCCF) [32], and Collaborative filtering method based on the concept of "friend of a friend" (CFfoaf) [33]. Here, we list the comparison methods that are not mentioned in the related work.

- **UTAOS**[31]: UTAOS is a NCF method and is designed for fast finding nearest neighbors for a given user by constructing the nearest neighbor tree in the item space of interest. In the prediction stage, UTAOS also uses Sim-pred to predict the score of items.
- **NUSCCF**[32]: NUSCCF defines a new similarity measurement method related to the construction of

[1] http://www.netflixprize.com

nearest neighbor tree and then constructs three nearest neighbor sets in three item subspaces. NUSCCF is an extension of UTAOS.

– **CFfoaf**[33]: CFfoaf alleviates the "grey sheep" problem by extending the set of nearest neighbors; that is, there is no nearest neighbors for the target user. For users who do not have any common rating items, CFfoaf establishes the relationship between them through intermediate friends. CFfoaf also adopts Simpred to predict ratings.

As mentioned in Section 2.2.2, URP has its own similarity measurement scheme (4). For other methods, PCC (1) is adopted. In general, NCF algorithms provide each user with an ordered list of unrated items, which is termed as the recommendation list [34]. If the predicted rating scores of items in the recommendation list are greater than or equal to θ, they should be recommended to the target user. We select three indexes to evaluate the recommendation performance of compared algorithms, including mean absolute error (MAE), recall, and coverage [18, 35, 36].

MAE is the most commonly used measure to evaluate the accuracy of recommendation algorithms and can be calculated by

$$MAE = \frac{1}{m'} \sum_{v=1}^{m'} \frac{1}{n'_v} \sum_{j=1}^{n'_v} \left| r_{vj} - \widetilde{r}^*_{vj} \right| \tag{15}$$

where r_{vj} is the actual rating that user u_v assigns to item t_j, $\widetilde{r}^*_{vj}$ represents the predicted rating of user u_v on item t_j, m' is the total number of users in the test set, and n'_v is the number of items that user u_v can predict.

Recall indicates how many positive examples are in the recommended list. The higher the recall rate is, the better the recommendation performance. The formula of calculating recall can be expressed as:

$$recall = \frac{1}{m'} \sum_{v=1}^{m'} \frac{\left| IR_{vp} \cap IR_{va} \right|}{\left| IR_{va} \right|} \tag{16}$$

where IR_{vp} means the list of items that could be possibly recommended to user u_v, and IR_{va} is the list of items that user u_v really likes in the test set.

Coverage is the ratio of recommended items to those evaluated by target users, and can be defined as follows:

$$coverage = \frac{\sum_{v=1}^{m'} \left| IR_{vp} \cap T_{vt} \right|}{\sum_{v=1}^{m'} \left| T_{vt} \right|} \tag{17}$$

where T_{vt} represents the collection of items rated by user u_v in the test set.

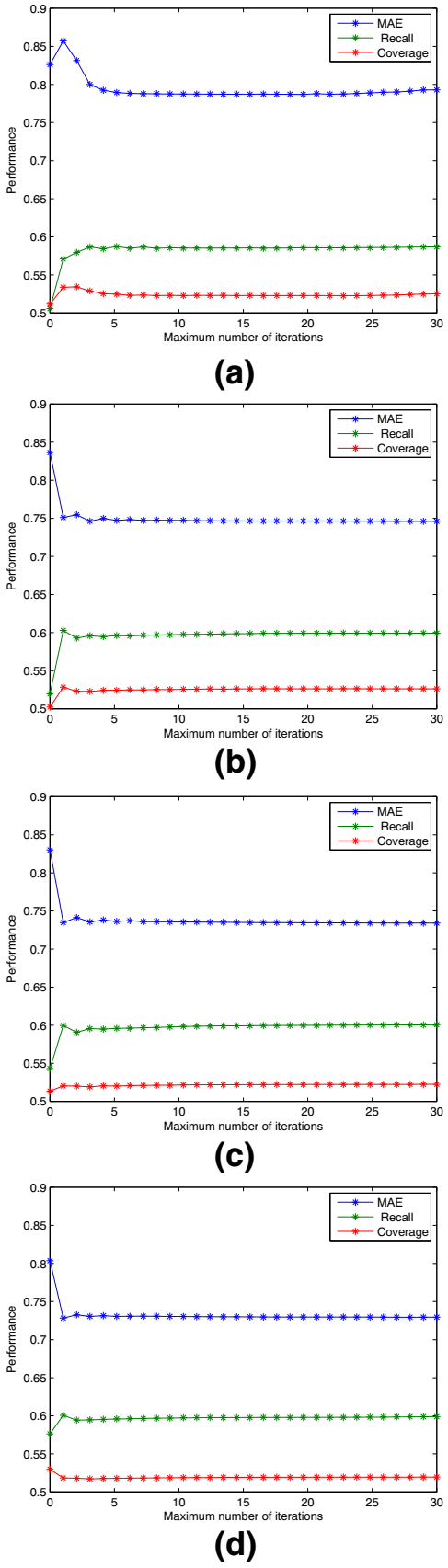

Fig. 3 Performance vs. $MaxIter$ under different K

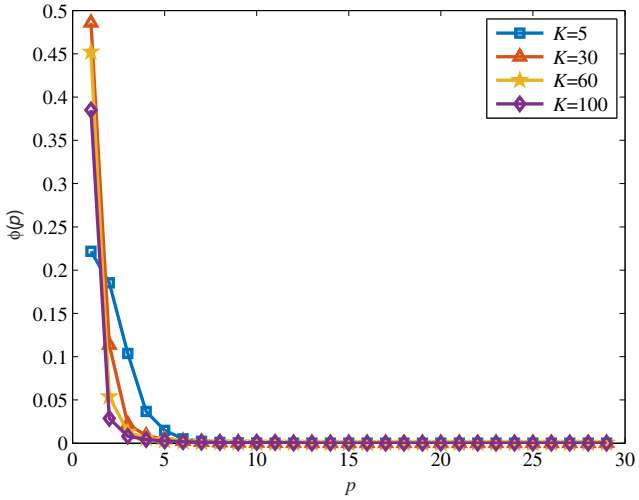

Fig. 4 Difference $\phi(p)$ vs. iteration number p under different K

4.3 Analysis of parameters for IRP

Our method has two parameters: the number of neighbors K and the maximum number of iterations $MaxIter$. To analyze the effect of parameters on the prediction performance of our method, we first perform 5-fold cross validation on the ML-100K dataset.

Let K vary in the set $\{5, 10, 20, \cdots, 100\}$, and the maximum number of iterations in the range of 1 to 30. The variation of average performance indexes vs. $MaxIter$ is given in Fig. 3. The situations under different neighbor numbers $K = 5$, $K = 30$, $K = 60$, and $K = 100$ are shown in Fig. 3a, b, c, and d, respectively. The horizontal axis of these sub-figures is the maximum number of iterations, and the vertical axis is the performance index. It can be seen from Fig. 3 that no matter what the value of K is, performance curves tend to stabilize after the sixth iteration, which proves that IRP is convergent.

To further validate the convergence of IRP, we calculate the difference between the predicted rating matrices in the current and previous iterations. Let $MaxIter = 30$ and $\phi(p) = \|\widetilde{\mathbf{R}}^{(p)} - \widetilde{\mathbf{R}}^{(p-1)}\|_F$, $p = 1, \cdots, MaxIter$. The variation of $\phi(p)$ on p is shown in Fig. 4. We see that the difference $\phi(p)$ approaches to 0 under different neighbor numbers with the increase of iterations. The greater the value of K is, the faster the difference $\phi(p)$ approaches to 0. In addition, the difference $\phi(p)$ is infinitely close to 0 when $p = 6$. As a result, IRP is convergent based on experimental results. In the following experiments, let $MaxIter = 10$ just to be sure.

The variation of average performance indexes vs. K on the prediction performance of our method is given in Fig. 5, which is a diagram with dual coordinates where curves of MAE, recall, and coverage are on the left, and that of time on the right. Observation on the curve of time indicates that the running time has an approximately linear relationship with K. The greater K is, the more time IRP needs. Thus, K should not be too greater. From curves of three performance indexes of MAE, recall, and coverage, we can see a commonality of them. These indexes fluctuate so little as K increases so that there is no significant change from the curves. Taking into account the performance factors of time, we set the number of neighbors K to 60 according to the comparative experimental results.

4.4 Experimental results

To validate the performance of IRP, we compare our methods with prediction methods on other four datasets. Besides, we also compare it with other novel NCF methods, UTAOS, NUSCCF, and CFfoaf that are mentioned in Section 4.2. These methods all use the prediction scheme of Sim-pred.

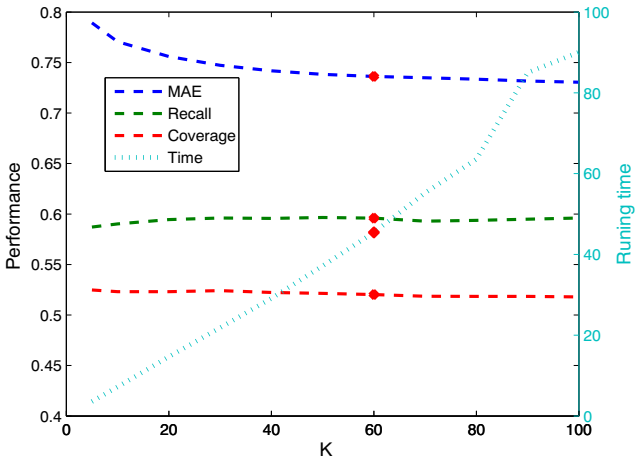

Fig. 5 Performance indexes vs. K

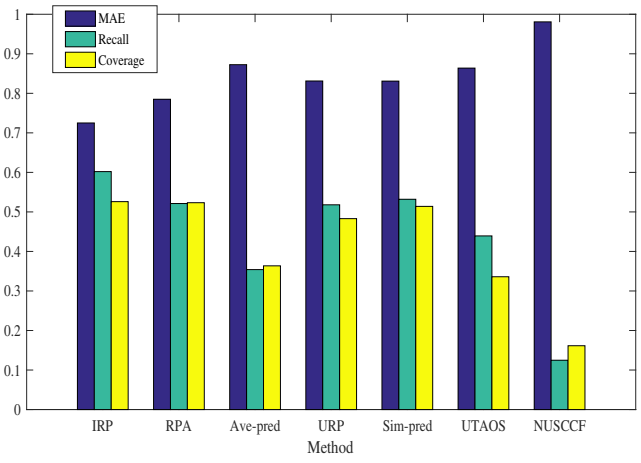

Fig. 6 Performance comparison on the ML-100k dataset

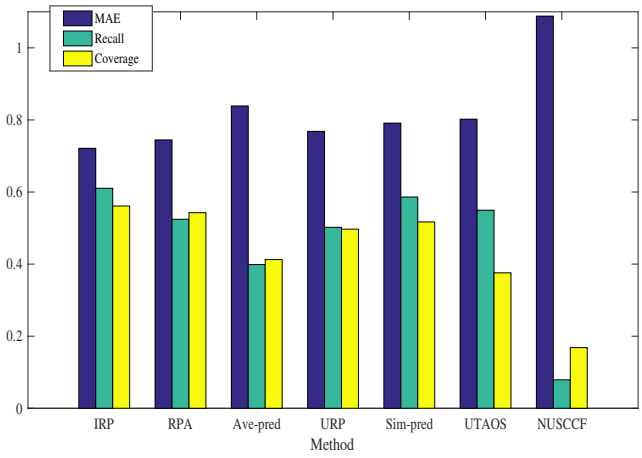

Fig. 7 Performance comparison on the ML-Latest-Small dataset

Figure 6 shows the performance comparison of eight methods on the ML-100k dataset, including average MAE, recall, and coverage. From Fig. 6, we can have the following conclusions. It can be seen from the MAE index that IRP is the best (MAE is the lower the better in recommendation systems), and it reduces the error rate by 6% compared to RPA. From the recall index (the higher the better), our method also achieves the best result. Although there is no obvious improvement in coverage, we can see that our method is still the best.

On the ML-Latest-Small dataset, We show the test the prediction performance (MAE, coverage, recall) of eight methods in Fig. 7. As can be seen from Fig. 7, we have the similar conclusions as those from Fig. 6. MAE, recall and coverage of our method are much better than compared methods. On this dataset, MAE of IRP has a 10% lower error rate than Ave-pred, 7% lower than Sim-pred, 2% lower than RPA, which shows the validity of the neighborhood propagation.

On the FilmTrust dataset, we list the comparison results in Table 4, where the best results are in bold type. We test the prediction performance (MAE, coverage, recall) of eight NCF methods. Observation on Table 4 indicates that IRP is much better in MAE, recall and coverage than all compared methods. On this dataset, MAE of IRP is lower than all other methods, and recall and coverage are higher. These can prove the validity of our method.

Table 4 Performance comparison on the FilmTrust dataset

Method	MAE	Recall	Coverage
IRP	**0.6426**	**0.5876**	**0.8400**
RPA	0.6592	0.3276	0.4670
Ave-pred	0.7612	0.2519	0.2937
URP	0.6556	0.5200	0.7900
Sim-pred	0.6884	0.5477	0.8225
UTAOS	0.7415	0.2545	0.3451
NUSCCF	1.0407	0.0660	0.0801
CFfoaf	0.6855	0.3909	0.3512

Table 5 Performance comparison on the Netflix dataset

Method	MAE	Recall	Coverage
IRP	**0.7554**	**0.5958**	**0.4943**
RPA	0.8359	0.3045	0.4269
Ave-pred	0.9926	0.2358	0.3730
URP	0.9392	0.2027	0.3177
Sim-pred	0.9492	0.2380	0.3729
UTAOS	0.8638	0.4393	0.3360
NUSCCF	0.9808	0.1248	0.1615
CFfoaf	0.9676	0.2020	0.3192

On the Netflix dataset, we show the comparison results of IPR with other methods in Table 5. Results in this table lead to the same conclusion that is obtained from the FilmTrust dataset. On each index, IRP achieves its best among eight methods, followed by RPA. Compared with RPA, IRP improves the prediction performance by 9.63% in MAE, 19.13% in recall, and 6.74% in coverage.

5 Conclusion

This paper proposes a novel iterative rating prediction algorithm for collaborative filtering-based recommender systems. The key contribution of this work is to enable a larger range of neighbor users to participate into the prediction process. IRP iteratively updates the predicted rating matrix using the rating information provided by direct and indirect neighbors. At the same, the propagation reliability matrix is updated with iterations, an element in which represents the reliable degree of the corresponding rating in the predicted rating matrix. Experiments are executed on four datasets with different sparsity levels. On the ML-100k, we analyze the effect of parameters on the algorithm performance. The experimental results imply that IRP can achieve to a stable state in a finite iterative number. It is necessary to determine an appropriate neighbor number considering the running time and the recommendation performance. On ML-Latest-Small dataset, FilmTrust and Netflix datasets, we validate the effectiveness of IRP. Our method achieves the best MAE, recall, and coverage among compared methods. Experiment results show that our scheme is a promising approach for recommendation.

Although IRP has an excellent performance on four datasets, there still are some points need study further. In theory, IRP can use any similarity measurements. In this paper, we choose PCC as the similarity method to measure

the correlation between users, but it is undeniable that PCC also has some defects, such as there is a negative score in the dataset and PCC only considers absolute value. In the future work, we would test our algorithms on other similarity measurements to check the performance of IRP. Besides, we know that the user-based CF is very similar to the item-based CF. The difference is that the user-based CF is to calculate the similarity between users, while the item-based CF calculates the similarity between items. Thus, it is worth investigating the performance of IRP when applying it to item-based recommendation.

Appendix A. Proof of Theorem 1

Proof According to the definition of the neighborhood relation, R can be represented by

$$R = \{(u_i, u_j) | u_j \in N_K(u_i) \subset U, u_i \in U\}$$

The construction of neighborhood relations depends on K nearest neighbors of users. In the following, we prove the properties of R.

1. Whatever measurement is adopted to search K nearest neighbors, the distance between a user u_i and itself is always zero. For any u_i in U, it is true that $u_i \in N_K(u_i)$. Thus, $(u_i, u_i) \in R$. In other words, R is reflexive.
2. Whatever measurement is adopted to search K nearest neighbors, the statement that if u_j is one of K nearest neighbors of u_i but u_i may be not one of K nearest neighbors of u_j is true. In other words, there are u_i and u_j in U such that $(u_i, u_j) \in R$ but $(u_j, u_i) \notin R$. Thus, R is not symmetric.
3. Whatever measurement is adopted to search K nearest neighbors, the statement that if $u_j \in U_K(u_i)$ and $u_k \in U_K(u_j)$ but $u_k \notin U_K(u_i)$ is true. In other words, there are u_i and u_j in U such that $(u_i, u_j) \in R$ and $(u_j, u_k) \in R$ but $(u_i, u_k) \notin R$. Thus, R is not transitive.

That completes the proof of Theorem 1. □

References

1. Kumar P, Thakur RS (2018) A framework for weblog data analysis using hive in hadoop framework
2. Kumar Malviya B, Agrawal J (2015) A study on web usage mining theory and applications, pp 935–939
3. Valera M (2014) A novel approach of mining frequent sequential pattern from customized web log preprocessing. Ijera
4. Kumar P, Thakur RS (2018) Recommendation system techniques and related issues: a survey. Int J Inf Technol 10(1):1–7
5. Rathod A, Indiramma M (2015) A survey of personalized recommendation system with user interest in social network. Int J Comput Sci Inf Technol 6(1):413–415
6. Lu J, Wu D, Mao M, Wang W, Zhang G (2015) Recommender system application developments: A survey. Decis Support Syst 74:12–32
7. Toledo RY, Martínez L (2017) Fuzzy tools in recommender systems: A survey. Int J Comput Intell Syst 10(1):776–803
8. Garcia I, Sebastia L, Onaindia E (2011) On the design of individual and group recommender systems for tourism. Expert Syst Appl 38(6):7683–7692
9. Choi SM, Ko SK, Han YS (2012) A movie recommendation algorithm based on genre correlations. Expert Syst Appl 39(9):8079–8085
10. Li Q, Myaeng SH, Kim BM (2007) A probabilistic music recommender considering user opinions and audio features. Inf Process Manag 43(2):473–487
11. Wen H, Fang L, Guan L (2012) A hybrid approach for personalized recommendation of news on the web. Expert Syst Appl 39(5):5806–5814
12. Gupta J, Gadge J (2015) Performance analysis of recommendation system based on collaborative filtering and demographics. 2015 International Conference on Communication, Information Computing Technology (ICCICT), pp 1–6
13. Isinkaye FO, Folajimi YO, Ojokoh BA (2015) Recommendation systems: Principles, methods and evaluation. Egyptian Inf J 16(3):261–273
14. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
15. Bobadilla J, Ortega F, Hernando A (2013) Recommender systems survey. Knowl-Based Syst 46(1):109–132
16. Su X, Khoshgoftaar TM (January 2009) A survey of collaborative filtering techniques. Adv Artif Intell 2009(1):1–19
17. Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook. Springer, Boston, pp 107–144
18. Patra BK, Launonen R, Ollikainen V, Nandi S (2014) Exploiting bhattacharyya similarity measure to diminish user cold-start problem in sparse data. In: Džeroski S, Panov P, Kocev D, Todorovski L (eds) Discovery Science. Springer International Publishing, Cham, pp 252–263
19. Bobadilla J, Ortega F, Hernando A (2012) A collaborative filtering similarity measure based on singularities. Inf Process Manag 48(2):204–217
20. Ahn HJ (2008) A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Inf Sci 178(1):37–51
21. Wang Y, Deng J, Gao J, Zhang P (2017) A hybrid user similarity model for collaborative filtering. Inf Sci 418-419:102–118
22. Liu H, Hu Z, Mian A, Tian H, Zhu X (2014) A new user similarity model to improve the accuracy of collaborative filtering. Knowl-Based Syst 56(3):156–166
23. Jameson A, Smyth B (2007) Recommendation to groups. In: Brusilovsky P, Kobsa A, Nejdl W (eds) The Adaptive Web, Methods and Strategies of Web Personalization, vol 4321. Springer, pp 596–627
24. Jannach D, Zanker M, Felfernig A, Friedrich G (2012) Recommender systems: An introduction. Inte J Hum Comput Interact 28(1):72–73

25. Zhang J, Pu P (2007) A recursive prediction algorithm for collaborative filtering recommender systems, pp 57–64
26. Kumar P, Kumar V, Thakur RS (2018) A new approach for rating prediction system using collaborative filtering. Iran J Comput Sci:1–7
27. Harper FM, Konstan JA (2016) The movielens datasets: History and context. ACM Trans Interact Intell Syst 5(4):19:1–19:19
28. Guo G, Zhang J, Yorke-Smith N (2013) A novel bayesian similarity measure for recommender systems. In: Rossi F (ed) Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI). IJCAI/AAAI, Beijing, pp 2619–2625
29. Koohi H, Kiani K (2016) User based collaborative filtering using fuzzy c-means. Measurement 91:134–139
30. Tsai CF, Hung C (2012) Cluster ensembles in collaborative filtering recommendation. Appl Soft Comput J 12(4):1417–1425
31. Ramezani M, Moradi P, Akhlaghian F (2014) A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains. Physica A Stat Mech Appl 408(32):72–84
32. Koohi H, Kiani K (2017) A new method to find neighbor users that improves the performance of collaborative filtering. Expert Syst Appl 83:30–39
33. Margaris D, Vassilakis C (2020) Improving collaborative filtering's rating prediction coverage in sparse datasets by exploiting the 'friend of a friend' concept. Int J Big Data Intell 7(1):47–57
34. Wen Y, Liu Y, Zhang ZJ, Xiong F, Cao W (2014) Compare two community-based personalized information recommendation algorithms. Physica A Stat Mech Appl 398(3):199–209
35. Sarwar BM, Konstan JA, Borchers A, Herlocker J, Miller B, Riedl J (1998) Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, CSCW '98. ACM, New York, pp 345–354
36. Zhang J, Peng Q, Sun S, Liu C (2014) Collaborative filtering recommendation algorithm based on user preference derived from item domain features. Physica A Stat Mech Appl 396(2):66–76

Li Zhang received the B.S. degree in 1997 and the Ph.D. degree in 2002 in electronic engineering from Xidian University, Xi'an, China. Now she is a full professor with the School of Computer Science and Technology, Soochow University, Suzhou, China. She was a postdoctor at the Institute of Automation, Shanghai Jiao Tong University, Shanghai, China, from 2003 to 2005. She worked as an associate professor at the Institute of Intelligent Information Processing, Xidian University, Xian, China, from 2005 to 2010. She was a visiting professor at Yuan Ze University, Taiwan, from February to May 2010. She has authored/co-authored more than 150 technical papers published in journals and conferences. Her research interests have been in the areas of machine learning, pattern recognition, neural networks and intelligent information processing.

Zepeng Li was born in Lianyungang, Jiangsu province. He received the M.S. degree in computer technology from Soochow University, Suzhou, in 2020. At present, he works in 58.com. His research interests include machine learning and data mining.

Xiaohan Sun was born in Xinyang, Henan province. She is currently pursuing the Masters degree at the School of Computer Science and Technology, Soochow University, Suzhou, China. Her research interests include machine learning and recommender system.