Knowl Inf Syst
DOI 10.1007/s10115-016-0920-5

REGULAR PAPER

Measure prediction capability of data for collaborative filtering

Xijun Liang[1] · **Zhonghang Xia**[2] · **Liping Pang**[3] · **Liwei Zhang**[3] · **Hongwei Zhang**[3]

Received: 28 January 2015 / Revised: 29 October 2015 / Accepted: 20 January 2016

Abstract Collaborative filtering (CF) approaches have been widely been employed in e-commerce to help users find items they like. Whereas most of existing work focuses on improving algorithmic performance, it is important to know whether the recommendation for users and items can be trustworthy. In this paper, we propose a metric, "relatedness," to measure the potential that a user's preference on an item can be accurately predicted. The relatedness of a user–item pair is determined by a community which consists of users and items most related to the pair. The relatedness is computed by solving a constrained ℓ_1-regularized least square problem with a generalized homotopy algorithm, and we design the homotopy-based community search algorithm to identify the community by alternately selecting the most related users and items. As an application of the relatedness metric, we develop the data-oriented combination (DOC) method for recommender systems by integrating a group of benchmark CF methods based on the relatedness of user–item pairs. In

This work is partially supported by the Natural Science Foundation of China under Grant 11171049 and 61503412, Natural Science Foundation of Shandong Province under Grant No. ZR2014AP004, and Fundamental Research Funds for the Central Universities under Grant No. 15CX02051A.

✉ Xijun Liang
liangxijunsd@163.com

[1] College of Science, China University of Petroleum, Qingdao 266555, China

[2] Department of Computer Science, Western Kentucky University, Bowling Green, KY 42101, USA

[3] School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China

experimental studies, we examine the effectiveness of the relatedness metric and validate the performance of the DOC method by comparing it with benchmark methods.

Keywords Collaborative filtering · Community · Relatedness · Homotopy algorithm · ℓ_1-Regularized least squares problem

1 Introduction

Collaborative Filtering has extensively been studied [2,17,34] since 1990's and successfully applied for recommending items (such as books, movies, or television programs) to users by e-commerce providers, such as Amazon, Netflix and TiVo. While most CF approaches focus on improving algorithmic performance in terms of evaluation metrics, such as accuracy [15,17], mean absolute error [5,34], root-mean-squared error [41], correlation [39], precision/recall [38], confidence [1,27], limited attention has been given to the influence of users' ratings on the algorithmic performance. Actually, the prediction accuracy of a CF algorithm highly depends on how "good" a pair of user and item is for prediction. For instance, a CF algorithm may provide a trustworthy prediction for a user who has a constant rating on similar items, but untrustworthy one for a new user who has few rating records stored in the system. Another instance is the unsolicited commercial e-mails, which has become a serious problem annoying users and overloading e-mail systems. Although the recommendation system does help service providers send their commercial e-mails more specially, a lot of recommendations are not interesting to users at all. Actually, with the pre-knowledge of prediction capability, service providers can only recommend their services to users when they are confident, resulting in better marketing reputation and trust-based relationships with customers.

The aim of measuring the prediction capability of a user–item pair is twofold. First, with the pre-knowledge of prediction capability, users can be more confident on the recommendation of CF algorithms about those "good" data. The prediction capability reflects the potential of the user–item pair, whose rating can be correctly predicted. A "good" user–item pair usually contains sufficient rating information for prediction, and thus its predicted rating is trustworthy. On the other hand, a "bad" user–item pair lacks adequate rating information to formulate accurate recommendations, and thus the prediction is not trustworthy. Consequently, users may not even trust the recommendation for those "good" user–item pairs if "good" and "bad" data cannot be distinguished. Second, the knowledge about the prediction capability of a user–item pair can help design a CF algorithm to improve its performance. According to experimental results based on multiple well-known CF algorithms, prediction accuracy for "good" user–item pairs is generally higher than those "bad" ones. This observation provides us a guideline in the algorithm design: "good" and "bad" data should be dealt with different methods.

As the CF algorithm exploits ratings of other user–item pairs for computing prediction, the prediction capability of a user–item pair can be measured by its similarity to these user–item pairs. Pearson correlation [34] and cosine similarity [5,37] are typical metrics for measuring the similarity between two users/items. Some variations [17,26] of the two metrics were extended to describe more accurate user–item relationships. Both Pearson correlation and cosine similarity are defined over the entire set of items, which is neither necessary nor realistic. It may not be well defined for pairwise similarity in the sparse data setting when most users rate only few items in a large-scale information system. Also, some users may either disagree with others or have very unique tastes. Finding common rating patterns over entire item sets for these types of users is extremely difficult. To overcome these problems, some works [43,44] have studied partial similarity over a subset of users or items for computing prediction.

The bicluster concept [40,46] and graphical models [20] were introduced in the CF system to characterize partial similarity over both users and items. The bicluster method employs only already-known ratings in the user–item matrix and thus alleviates sparsity problem. However, these biclusters do not work as indicators to reflect the prediction capability of a user–item pair since they only identify subsets of users and items having certain coherent relationships. Some user–item pairs may not be included in any bicluster even though they have sufficient information for computation of prediction. Although the graphical model alleviates this problem by including users and items from different clusters, it is mainly used to generate more accurate recommendation. All the user–item pairs in a cluster are associated with the same cluster, regardless of the rating variance of each user–item pair, and thus the prediction capability of each individual user–item pair remains unclear.

In this paper, we develop a metric, called relatedness, to indicate the prediction capability of a user–item pair. The goal is to identify those user–item pairs whose preferences can be accurately predicted. The value of relatedness actually reflects prediction capability of a specific community, which is composed of the users and items most related to the user–item pair. The relatedness of the user–item pair is computed from two aspects: users and items. For the user relatedness, the user's rating vector in the user–item matrix is represented as a linear combination of other users' rating vectors. The task of finding the most related users is formulated as a constrained ℓ_1-regularized least square (ℓ_1-RLS) problem, in which the ℓ_1-norm constraint is added as a regularization term to remove those unrelated users. Similarly, we can compute the item relatedness. When a user–item pair has multiple communities, we seek for the one which maximizes the relatedness value.

We design an efficient greedy strategy, homotopy-based community search (HCS) algorithm, to search a good community. The active user–item pair is assumed to have a certain number of explicit ratings, and the HCS algorithm runs offline to calculate its relatedness value by solving an formulated optimization problem. At each iteration, HCS finds the most related users and items alternately and deletes those unrelated users and items from the current community. Motivated by homotopy algorithm [12,30], we develop the generalized homotopy (GH) algorithm to solve the ℓ_1-RLS problem for selecting related users and items. Both user and item relatedness are calculated, and the smaller one is defined as the relatedness of the user–item pair. This process is repeated until the community size reaches a threshold. Although the entire user–item matrix in a large-scale problem is huge, known ratings for a particular active user–item pair are usually very limited and thus the optimization problem for computing relatedness is a small-sized problem. Moreover, user–item pairs are independent, and thus, computation of relatedness for different user–item pairs can be implemented in parallel.

In [42], authors have shown that a linear combination scheme has significantly improved prediction accuracy on the Netflix dataset. This scheme combined multiple CF prediction results with a set of suitable weights. In fact, the quality of those known ratings is different and thus weights assigned to the ratings should change according to the confidence. In this paper, we propose the data-oriented combination (DOC) method as an application of the relatedness metric, which classifies the ratings based on their prediction quality and assign a weight to each CF algorithm according to the category of the user–item pair. DOC first calculates predictions of multiple classical CF algorithms individually and then combines these predictions based on the relatedness of user–item pairs. As the prediction capability of a user–item pair can be computed beforehand, we can choose a proper CF algorithm to get better prediction. Our experimental studies show that the relatedness metric contributes a lot in the DOC method to improve prediction accuracy over MovieLens [35], EachMovie [8] and Yahoo! Music [45] datasets.

The main contributions of this work are summarized as follows:

– Introduces the relatedness metric to indicate the potential that a user–item pair can be predicted accurately.
– Designs HCS and GH algorithms to compute the relatedness of a user–item pair.
– Develops the DOC CF method based on relatedness to improve prediction performance.

The rest of the paper is organized as follows: In Sect. 2, we introduce the preliminary knowledge of the CF problem and linear regression. In Sect. 3, we define the relatedness metric for a user–item pair and formulate the computation of the relatedness as an optimization problem. Section 4 presents the HCS algorithm to search good communities and calculate the relatedness value. Details of the GH algorithm for selecting the most related users and items are discussed in Sect. 5. We study the DOC method in Sect. 6. Experimental results are shown in Sect. 7. Related work is discussed in Sect. 8. Section 9 states the conclusion of the paper.

2 Collaborative filtering and linear regression

Given a recommender system consisting of m users from user set U and n items from item set I, users' opinions on items are represented by ratings. These ratings form a user–item matrix $R = (R(u, i))$, where $u \in U$, $i \in I$, and $R(u, i)$ denotes the numeric rating corresponding to user–item pair (u, i). Note that an entry in R is missing if no rating has been given by the corresponding user–item pair. The task of collaborative filtering is to predict missing rating $R(u_0, i_0)$ for active user–item pair (u_0, i_0). Let $U_0 \subseteq \{1, \ldots, m\}$ be the set of users who have rated item i_0, $I_0 \subseteq \{1, \ldots, n\}$ the set of items that user u_0 has rated, and $R_0 = R(U_0, I_0)$ the matrix consisting of ratings given by user set U_0 on item set I_0. Denote by $R(u_0, I_0)$ the vector of ratings given by user u_0 on I_0 and $R(U_0, i_0)$ the vector of ratings received by item i_0 from U_0.

Typically, the prediction of $R(u_0, i_0)$ depends on the users who have rating patterns similar to u_0's. The user-based nearest neighbor (nNbr-user) method [34] estimates $R(u_0, i_0)$ by

$$\hat{R}(u_0, i_0) = \bar{R}_{u_0} + \frac{\sum_{v \in U_0} \mathrm{sim}(u_0, v)(R(v, i_0) - \bar{R}_v)}{\sum_{v \in U_0} |\mathrm{sim}(u_0, v)|}, \tag{1}$$

where $\bar{R}_{u_0}$ and $\bar{R}_v$ are the average ratings of user u_0 and v, respectively, and $\mathrm{sim}(u_0, v)$ denotes the Pearson correlation similarity between users u_0 and v. Subtracting user's average rating from user–item matrix R, i.e.,

$$R(v, i_0) := R(v, i_0) - \bar{R}_v,$$

we can rewrite the estimation of $\hat{R}(u_0, i_0)$ as a linear combination of $R(v, i_0)$,

$$\hat{R}(u_0, i_0) = \sum_{v \in U_0} w_{u_0 v} R(v, i_0),$$

where $w_{u_0 v} = (\sum_{v \in U_0} |\mathrm{sim}(u_0, v)|)^{-1} \cdot \mathrm{sim}(u_0, v)$.

Table 1 Notations

R	The user–item rating matrix
(u_0, i_0)	The active user–item pair
$R(u, i)$	The numeric rating given by user u on item i
U, I	The set of all users, and the set of all items
Ω	The set of all user–item pairs, i.e., $\{(u, i)\|u \in U, i \in I\}$
U_0, I_0	The set of users who have rated item i_0, and the set of items that user u_0 has rated
U_1, I_1	The set of users in the community, and the set of items in the community
R_0	$R(U_0, I_0)$; the $p_0 \times q_0$ submatrix of R corresponding to U_0 and I_0
$R(u_0, I_0)$	The vector of ratings given by u_0 on I_0
$R(U_0, i_0)$	The vector of ratings given by U_0 on i_0
A	$R(U_1, I_1)$; the community of (u_0, i_0), which is a $p \times q$ submatrix of R_0
r_{i_0}	$R(U_1, i_0)$; the vector of ratings given by U_1 on i_0
r_{u_0}	$R(u_0, I_1)$; the vector of ratings given by u_0 on I_1

In the viewpoint of linear regression, weight vector w can be treated as regression coefficients of least squares

$$\min_w \|R(U_0, I_0)w - R(U_0, i_0)\|^2. \tag{2}$$

Similarly, the item-based nearest neighbor (nNbr-item) method [37] estimates $R(u_0, i_0)$ by

$$\hat{R}(u_0, i_0) = \frac{\sum_{j \in I_0} \text{sim}(i_0, j) \cdot R(u_0, j)}{\sum_{j \in I_0} |\text{sim}(i_0, j)|}$$

where $\text{sim}(i_0, j)$ indicates the similarity between item i_0 and item j. The least squares for regression coefficients w is

$$\min_w \|R(U_0, I_0)^T w - R(u_0, I_0)\|^2. \tag{3}$$

Both nNbr-user and nNbr-item methods exploit linear relationship between users and items to predict unknown ratings.

A list of notations is given in Table 1 for convenient references. Some of them will be clearly explained when they are used.

3 Community and relatedness

A group of users with similar interests tend to have similar rating preference on a set of items, and thus, as rating reference for a user in the group. Such a group of users and items corresponds to a submatrix in the user–item matrix, called a community. For an example of a CF system with 5 users and 8 items, the user–item matrix is shown in Fig. 1. Although the similarity among these users is very small over all 8 items, the rating patterns given by u_1, u_2 and u_3 are very similar to active user u_0's on items i_4, i_5, i_6 and i_7. These users and items construct a community (shadow in Fig. 1) of (u_0, i_0), denoted by $((u_1, u_2, u_3), (i_4, i_5, i_6, i_7))$. A "good" community is more likely to have a strong support for the prediction of the active user–item

Fig. 1 User–item matrix and community. *Question mark* represents the active user–item pair. *White box* represents a missing rating. A "good" community of (u_0, i_0) is *colored* in *shadow*

$$\begin{array}{c|ccccccc|c}
 & i_1 & i_2 & i_3 & i_4 & i_5 & i_6 & i_7 & i_0 \\
u_0 & 2 & 5 & 1 & 4 & 4 & 4 & 2 & \boxed{?} \\
u_1 & 2 & 3 & 3 & 5 & 5 & 5 & 3 & 3 \\
u_2 & 5 & 2 & 2 & 4 & 4 & 4 & 2 & 2 \\
u_3 & 3 & 1 & 1 & 2 & 2 & 2 & 1 & 1 \\
u_4 & 4 & \square & 3 & 5 & 2 & 1 & \square & 5
\end{array}$$

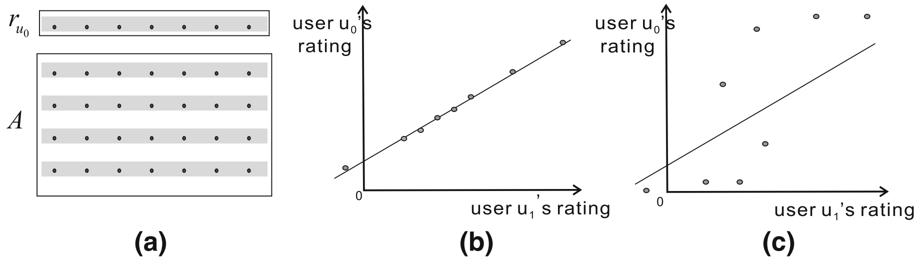


Fig. 2 User relatedness. **a** Vector representation of users. Each user is represented by a row in the user–item matrix. **b** A case of large user relatedness. Two users contain strong linear relationship. A point in graph represents a pair of ratings $(R(u_0, i),\ R(u_1, i))$ given by two users u_0 and u_1 on an item i. **c** A case of small user relatedness. Two users contain weak linear relationship

pair. In this example, community $((u_1, u_2, u_3), (i_4, i_5, i_6, i_7))$ safely implies $R(u_0, i_0) = 2$. On the other hand, a "bad" community is less helpful with the prediction. In Fig. 1, users $\{u_2, u_3, u_4\}$ and items $\{i_1, i_2, i_3\}$ form another community, but it cannot safely imply the rating of $R(u_0, i_0)$.

Intuitively, a "good" community is formed by a set of users highly related to the active user and a set of items highly related to the active item. Motivated by the linear regression representation in Sect. 2, we develop a new metric, "relatedness," for the active user–item pair to indicate the potential that it can be predicted beforehand according to its community.

Given community $A = R(U_1, I_1)$ for (u_0, i_0), which is composed of p users from set $U_1 \subseteq U_0$, and q items from set $I_1 \subseteq I_0$. Without loss of generality, we assume $p < q$. For simplicity, we only seek the linear relationship among users and items. Note that the relatedness of (u_0, i_0) is determined by its user relatedness and item relatedness.

The user relatedness reflects the degree of linear relationship between the active user and users within the community. As illustrated in Fig. 2a, a user can be represented as a vector of ratings over item set I_1, corresponding to a row in A, and the active user is denoted by $r_{u_0} := R(u_0, I_1)$. With linear regression, we can model this relationship by fitting a linear function to observed data (users in the community). Specifically, the active user is a dependent variable, and other users are independent variables. Moreover, the degree of linear relationship is consistent with the magnitude of fitness. Figure 2b, c shows two cases of the linear relationship in a community having two users. The community in Fig. 2b has a larger user "relatedness" than that in (c), since users in Fig. 2b fit the line better.

The sum of squared residuals in linear regression reflects the degree of the linear relationship, and the regression coefficients can be computed by solving the following least squares problem

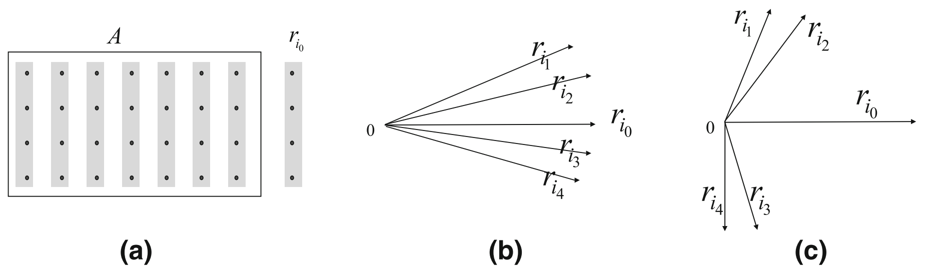


Fig. 3 Item relatedness. **a** Vector representation of items. Each item is represented by a column in the user–item matrix. **b** Four vectors close to r_{i_0} linearly represent it. **c** Four vectors far away from r_{i_0} linearly represent it

$$\min_{\alpha} \frac{1}{2}\|A^\top \alpha - r_{u_0}\|_2^2, \tag{4}$$

where α is the vector of regression coefficients, indicating the contribution of each user in linear representation of r_{u_0}. In some solutions, the regression coefficients are dominated only by a couple of users. For example, if $R(u_1, I_1) = r_{u_0}$, then one has solution $\alpha_1 = 1$ and $\alpha_i = 0$, for $i \neq 1$ to Problem (4), and the sum of squared residuals exactly equals 0. In this case, only user u_1 contributes in the linear representation. Our observation is that the contribution from sufficient users in the community is necessary to reduce the bias in prediction. To balance the contribution from users in the community, we add a constraint on α in Problem (4) as follows,

$$\begin{array}{ll} \min_\alpha & \frac{1}{2}\|A^\top \alpha - r_{u_0}\|_2^2 \\ \text{s. t.} & \|\alpha\|_\infty \leq \kappa_\alpha, \end{array} \tag{5}$$

where $\kappa_\alpha > 0$ is a constant.

The item relatedness reflects the degree of linear relationship between the active item and items within the community. As shown in Fig. 3a, an item is represented as a vector of ratings given by users in U_1, corresponding to a column in A, and the active item is denoted by $r_{i_0} := R(U_1, i_0)$. Similarly to Problem (4), the item relatedness can be characterized by the sum of squared residuals of problem

$$\min_\beta \frac{1}{2}\|A\beta - r_{i_0}\|^2, \tag{6}$$

where β is the vector of regression coefficients. However, the solution to Problem (6) does not work as an indicator for the item relatedness since linear system

$$A\beta = r_{i_0} \tag{7}$$

is underdetermined ($p < q$), and thus the sum of squared residuals is usually 0. In this case, the relationship between items and r_{i_0} can be determined by those item vectors in A whose linear combination represents r_{i_0}.

The solution to underdetermined linear system (7) is not unique. For example, vector r_{i_0} can be represented by a linear combination of four vectors $r_{i_1}, \ldots, r_{i_4}$ in either Fig. 3b or Fig. 3c. The basis pursuit method [9] suggests the set of vectors closer to r_{i_0} are more related to it. Hence, the community with items shown in Fig. 3b has a larger item "relatedness" than that shown in Fig. 3c. Here, we use a set of items closest to the active item obtained by solving problem

$$\begin{aligned}\min_\beta\ & \|\beta\|_1 \\ \text{s. t.}\ & A\beta = r_{i_0},\end{aligned} \tag{8}$$

for measuring the item "relatedness." A small value of $\|\beta\|_1$ in Problem (8) reflects a close relationship between items in the community and active item i_0. Likewise, we add a constraint on β, and the item relatedness is given by

$$\begin{aligned}\min_\beta\ & \|\beta\|_1 \\ \text{s. t.}\ & A\beta = r_{i_0}, \\ & \|\beta\|_\infty \le \kappa_\beta,\end{aligned} \tag{9}$$

where $\kappa_\beta > 0$ is a constant.

We reformulate Problem (5) and Problem (9) as constrained ℓ_1-RLS problems

$$\begin{aligned}\min_\alpha\ & \tfrac{1}{2}\|A^\top\alpha - r_{u_0}\|^2 + \lambda\|\alpha\|_1 \\ \text{s. t.}\ & \|\alpha\|_\infty \le \kappa_\alpha\end{aligned} \tag{10}$$

and

$$\begin{aligned}\min_\beta\ & \tfrac{1}{2}\|A\beta - r_{i_0}\|^2 + \lambda\|\beta\|_1 \\ \text{s. t.}\ & \|\beta\|_\infty \le \kappa_\beta\end{aligned} \tag{11}$$

where $\lambda > 0$ is a parameter. For convenience, we use a larger value of relatedness to reflect a better community and hence define the negative logarithm of optimal function values of Problem (10) and Problem (11) as the user and item relatedness for community A, respectively.

Therefore, the relatedness of community A with respect to (u_0, i_0) is determined by the smaller value of user and item relatedness, denoted by $\Gamma(A, u_0, i_0)$, i.e.,

$$\begin{aligned}&\Gamma(A, u_0, i_0) \\ &= \min\ \{-\ln\ \min\{\tfrac{1}{2}\|A\beta - r_{i_0}\|^2 + \lambda\|\beta\|_1 \mid \|\beta\|_\infty \le \kappa_\beta\}, \\ &\qquad\quad -\ln\ \min\{\tfrac{1}{2}\|A^\top\alpha - r_{u_0}\|^2 + \lambda\|\alpha\|_1 \mid \|\alpha\|_\infty \le \kappa_\alpha\}\} \\ &= \max_{\alpha,\beta}\ -\ln\max\{\tfrac{1}{2}\|A\beta - r_{i_0}\|^2 + \lambda\|\beta\|_1,\ \tfrac{1}{2}\|A^\top\alpha - r_{u_0}\|^2 + \lambda\|\alpha\|_1\} \\ &\quad\ \text{s. t.}\ \ \|\alpha\|_\infty \le \kappa_\alpha,\quad \|\beta\|_\infty \le \kappa_\beta\end{aligned} \tag{12}$$

where the last equation holds because α and β are unrelated.

Thus far, solving Problem (12) can only obtain the relatedness of a given community A. In order to identify the "best" community for active user–item pair (u_0, i_0), we search for a $p \times q$ community having the largest relatedness value in matrix $R_0 := R(U_0, I_0)$ by solving the following problem:

$$\begin{aligned}\max_{\alpha,\beta,U_1,I_1}\ & -\ln\ \max\{\tfrac{1}{2}\|R(U_1, I_1)\beta - R(U_1, i_0)\|^2 + \lambda\|\beta\|_1, \\ & \tfrac{1}{2}\|R(U_1, I_1)^\top\alpha - R(u_0, I_1)\|^2 + \lambda\|\alpha\|_1\} \\ \text{s. t.}\ \ & \|\alpha\|_\infty \le \kappa_\alpha,\quad \|\beta\|_\infty \le \kappa_\beta, \\ & U_1 \subseteq U_0,\ I_1 \subseteq I_0,\ |U_1| = p,\ |I_1| = q,\end{aligned} \tag{13}$$

where $|U_1|$ and $|I_1|$ denote the cardinality of U_1 and I_1, respectively. Here, p and q are preselected. The optimal function value of Problem (13) is defined as the relatedness of active user–item pair (u_0, i_0), denoted by $\Gamma(u_0, i_0)$. It indicates how well (u_0, i_0) can be predicted.

4 Search good communities

We develop the homotopy-based community search algorithm to find an ideal community A with p users and q items for the active user–item pair of (u_0, i_0). Starting with R_0, the HCS algorithm iteratively eliminates those items and users unrelated to the active item and user and maintains the most related items and users in two active sets, respectively. At each iteration, the two active sets are updated by the GH algorithm, discussed in Sect. 5, until the number of related users and items reaches preselected parameters p and q.

The most related items are computed by solving Problem (11) with the GH algorithm. Let $\tilde{I}$ be the set of the indexes of related items. The algorithm updates active items in community A and its corresponding r_{u_0} by

$$A := A(:, \tilde{I}), \quad r_{u_0} := r_{u_0}(\tilde{I}), \tag{14}$$

where $A(:, \tilde{I})$ denotes the submatrices of A consisting of its all rows but only those columns in $\tilde{I}$, $r_{u_0}(\tilde{I})$ denotes the vector consisting of the coordinates of r_{u_0} corresponding to $\tilde{I}$.

Similarly, the most related users are computed by solving Problem (10) with the GH algorithm. Let $\tilde{U}$ be the set of the indexes of related users. The algorithm then updates active users in community A and its corresponding r_{i_0} by

$$A := A(\tilde{U}, :), \quad r_{i_0} := r_{i_0}(\tilde{U}), \tag{15}$$

where $A(\tilde{U}, :)$ denotes the submatrices of A consisting of its all columns but only rows in $\tilde{U}$, $r_{i_0}(\tilde{U})$ denotes the vector consisting of the coordinates of r_{i_0} corresponding to $\tilde{U}$.

A critical step in the HCS algorithm is to determine the sizes of the two active sets, i.e., choosing parameters p and q. There is a trade-off between the relatedness value and the size of the community. A small community usually contains highly related users and items and thus has a large relatedness value. On the other hand, a large community usually contains some unrelated users and items and thus has small relatedness value. We are interested in the good community with a certain size range to include only the most related users and items.

We choose $p = p_0 \cdot \sigma$ and $q = q_0 \cdot \sigma$ with $\sigma \in (0, 1)$ for community A which is a submatrix of p_0-by-q_0 matrix R_0. Note that σ cannot be a constant as R_0 depends on the active user–item pair. For example, although $\sigma = 0.4$ might be a good choice for R_0 with $p_0 = 200$ users, which contains a potential community with up to $p = 200 \times 0.4 = 80$ users, it is not a good choice for R_0 with $p_0 = 20$ users as the size of the derived community is too small with $\sigma = 0.4$ ($p = 20 \times 0.4 = 8$). Hence, in order to find a community in a reasonable range, we intend to select a small σ when the size of R_0 is large, and a large σ otherwise. Toward this end, we choose $\sigma(\cdot)$ as a decreasing function and set p and q as follows:

$$p = p_0 \cdot \sigma(p_0), \quad q = q_0 \cdot \sigma(q_0).$$

Also, $\sigma(\cdot)$ is chosen such that $\phi(t) := t \cdot \sigma(t)$ is an increasing function of t, but its increasing rate decreases. Summing up these considerations, we choose function $\sigma(\cdot)$ satisfying the following properties:

(a). $\sigma(t) : \mathbb{R}_+ \longrightarrow (0, 1]$ strictly decreases with t;
(b). $\phi(t) = t \cdot \sigma(t)$ increases with t;
(c). $\phi'(t)$ decreases with $t > n_1$ for some $n_1 > 0$.

Here, we define

$$\sigma(n) = \begin{cases} \frac{n(r_1 - r_0)}{n_1} + r_0, & \text{if } n \leq n_1 \\ \theta(n+1)^{\omega}, & \text{otherwise} \end{cases} \tag{16}$$

where

$$\omega = \frac{\log \frac{r_1}{r_2}}{\log \frac{n_1+1}{n_2+1}},$$

and

$$\theta = r_1(n_1+1)^{-\omega},$$

where $1 > r_0 > r_1 > r_2 > 0, n_2 > n_1 > 0$, and $n_2 r_2 > n_1 r_1$, n_1 is the estimation of an ideal number of users or items in a community and r_1 is the value of σ at n_1, i.e., $r_1 = \sigma(n_1)$; n_2 is the threshold that R_0 is defined a large matrix, and r_2 is the value of σ at n_2, i.e., $r_2 = \sigma(n_2)$; r_0 is the value of σ at 0, i.e., $r_0 = \sigma(0)$. All these parameters are chosen independent of the active user–item pair.

This procedure is summarized in Algorithm 1.

Algorithm 1 Homotopy-based community search algorithm

Input: R_0, r_{i_0}, r_{u_0};
Output: (u_0, i_0)'s community A and its relatedness;

```
1: Initialization: $A \longleftarrow R_0, \tilde{q} \longleftarrow q_0, \tilde{p} \longleftarrow p_0, \rho \in (0,1)$;
2: Set the adaptive size of community: $q \longleftarrow q_0 \cdot \sigma(q_0), p \longleftarrow p_0 \cdot \sigma(p_0)$;
3: while $(\tilde{p} > p \vee \tilde{q} > q)$ do
4:    Select related items:
5:        $\tilde{q} \longleftarrow \max\{\rho \cdot \tilde{q},\ q\}$;
6:        $\tilde{I} \longleftarrow \mathrm{GH}(A, r_{i_0}, q)$;
7:        Update $A$ and $r_{u_0}$ via Eq. (14);
8:    Select related users:
9:        $\tilde{p} \longleftarrow \max\{\rho \cdot \tilde{p},\ p\}$;
10:       $\tilde{U} \longleftarrow \mathrm{GH}(A^T, r_{u_0}, p)$;
11:       Update $A$ and $r_{i_0}$ via Eq. (15);
12: end while
13: Calculate the relatedness value $\Gamma$ via Problem (12).
```

Remark (1) At line 5 and 9, the size of the target community should not be less than threshold q and p, respectively. Hence, $\tilde{q}$ takes $\max\{\rho \cdot \tilde{q},\ q\}$. Similarly, $\tilde{p}$ takes $\max\{\rho \cdot \tilde{p},\ p\}$. Parameter $\rho \in (0, 1)$ represents the fraction of users or items remaining in the active sets at each iteration.
(2) Parameter λ in Problem (13) is obtained when the GH algorithm terminates.
(3) We obtain an estimation of $R(u_0, i_0)$ when a good community is found by Algorithm 1. In fact, at line 6, the GH algorithm solves not only $\tilde{I}$ but also optimal solution β^* when Algorithm 1 terminates. Then, $\langle r_{u_0}, \beta^* \rangle$ is an estimation for $R(u_0, i_0)$.

As the HCS algorithm iteratively calls the subroutine algorithm GH, its computational complexity is determined by the number of its iterations and the complexity of GH. The number of iterations in HCS is estimated by

$$ite_{HCS} = \max\left\{\left\lceil \frac{\ln(q/q_0)}{\ln \rho} \right\rceil, \left\lceil \frac{\ln(p/p_0)}{\ln \rho} \right\rceil\right\},$$

where $\lceil\cdot\rceil$ is the ceiling function, and p, q, ρ are parameters in the algorithms and are pre-chosen for each active user–item pair. Hence, ite_{HCS} is upper-bounded by a constant.

5 Generalized homotopy algorithm

In this section, we present the GH algorithm to seek the highly related users and items by solving Problem (10) and Problem (11). The homotopy algorithm [12,30] and its variants [11] have been proposed to solve unconstrained ℓ_1-RLS problem

$$\min_x \frac{1}{2}\|Ax - r\|^2 + \lambda\|x\|_1 \tag{17}$$

by exploiting the piecewise linear property of the regularization path. These path-following methods compute the entire solution path in Problem (17), and their solutions are often extremely sparse [10]. The sparse solution selects the columns in A highly related to vector r. Here, we extend the homotopy algorithm to solve a general constrained ℓ_1-RLS problem

$$\begin{array}{ll} \min_x & \frac{1}{2}\|Ax - r\|_2^2 + \lambda\|x\|_1 \\ \text{s.t.} & \|x\|_\infty \le \kappa, \end{array} \tag{18}$$

where $x \in \mathbb{R}^l$, l is the column number of A, $\lambda > 0$ and $\kappa > 0$ are parameters. The classical homotopy algorithm solves the unconstrained ℓ_1-RLS problem, without box constants. When box constraints are added, complementary slackness condition and Lagrange multipliers (Eq.(21)) are required to characterize the optimality conditions. As a result, it is not straightforward to define an iteration format for the solution. Particularly, the constraints introduce a new index set J_3. We develop a formula for updating three index sets, J_1, J_2 and J_3 and design a new step size formula. Compared with other common solvers, GH algorithm generates a solution with specified number of nonzeros. These characteristics make GH embed HCS well.

Optimality conditions It is well known [36] that x is an optimal solution of convex programming (18) if and only if there exists $u \in \mathbb{R}$ such that the following Karush–Kuhn–Tucker conditions hold

$$\left\{\begin{array}{ll} 0 \in \partial_x L(x, u), & (19) \\ \|x\|_\infty \le \kappa, & (20) \\ u \ge 0, \quad u(\|x\|_\infty - \kappa) = 0, & (21) \end{array}\right.$$

where $\partial_x L(x, u)$ is subdifferential of Lagrange function

$$L(x, u) = \tfrac{1}{2}\|Ax - r\|^2 + \lambda\|x\|_1 + u(\|x\|_\infty - \kappa),$$

and u is a Lagrange multiplier.

Let

$$\begin{aligned} &J = \{i \mid x_i \ne 0\}, \\ &J_1 = \{i \mid |x_i| = \|x\|_\infty\}, \quad J_2 = J \backslash J_1, \quad J_3 = \{1, \ldots, l\} \backslash J. \end{aligned} \tag{22}$$

If $x \ne 0$, then one has $J = J_1 \cup J_2$, $\{1, \ldots, l\} = J_1 \cup J_2 \cup J_3$.

Note that Eq. (19) is equivalent to

$$0 \in A^T(Ax - r) + \lambda \cdot \partial\|x\|_1 + u \cdot \partial\|x\|_\infty, \tag{23}$$

where

$$\partial\|x\|_\infty = \operatorname{conv}\{\operatorname{sgn}(x_i)e_i \mid |x_i| = \|x\|_\infty\},$$

where "conv" denotes the convex hull of a set, e_i is the unit vector with 1 at the ith position, and

$$\partial \|x\|_1 = \left\{ u \in \mathbb{R}^l \;\middle|\; \begin{array}{l} u_i = \operatorname{sgn}(x_i), \text{ if } x_i \neq 0 \\ u_i \in [-1, 1], \text{ if } x_i = 0 \end{array} \right\}.$$

We use the following notations in the rest of the section. For an index set S, let A_S denote the submatrix consisting of the columns in A whose indices belong to S, and let $v(S)$ denote the vector consisting of the elements in a vector v whose indices also belong to S. Then Eq. (19) can be transformed to

$$\begin{cases} -A_{J_1}^T(Ax - r) \in \lambda \cdot \operatorname{sgn}(x(J_1)) \\ \qquad\qquad +u \cdot \operatorname{conv}_{i \in J_1}\{\operatorname{sgn}(x_i) \cdot e_i(J_1)\}, & (24) \\ -A_{J_2}^T(Ax - r) = \lambda \cdot \operatorname{sgn}(x(J_2)), & (25) \\ |A_{J_3}^T(Ax - r)| \leq \lambda. & (26) \end{cases}$$

GH algorithm The GH algorithm computes the solution to Problem (18) by iteratively updating x and λ until the criterion is met. At each iteration, λ decreases with a certain amount, and a new x is found to satisfy KKT conditions. The detailed implementation of the algorithm is described as follows.

Initially, we set

$$\lambda^0 = \|A^T r\|_\infty, \tag{27}$$

such that $x^0 = 0$ is an optimal solution to the constrained ℓ_1-RLS problem at λ^0. At the kth iteration, we update x by

$$x^{k+1} = x^k + \gamma^k d^k, \tag{28}$$

where superscript k is used to represent variables at the iteration k, d^k is the direction along which x is updated, and γ^k is the step size. In the following, we compute d^k by (25) and γ^k by (20) and (24)–(26).

a. Compute direction d^k.

Denote by

$$c^k = -A^T(Ax^k - r), \tag{29}$$

the residual correlation [10,12] at iteration k. Then it follows by Eq. (24) and (25) that $\operatorname{sgn}(c^k(J_1^k)) = \operatorname{sgn}(x^k(J_1^k))$ and $\operatorname{sgn}(c^k(J_2^k)) = \operatorname{sgn}(x^k(J_2^k))$. We seek direction d^k such that 1) it belongs to the linear space generated by $\{a_i\}_{i \in J_2^k}$; 2) all magnitudes of residual correlations c^k on J_2^k decline equally. That is, $d^k(J_1^k) = 0$, $d^k(J_3^k) = 0$ and for any $\gamma > 0$,

$$-A_{J_2^k}^T[A(x^k + \gamma d^k) - r] = (\lambda^k - \gamma) \cdot \operatorname{sgn}(c^k(J_2^k)).$$

Thus,

$$A_{J_2^k}^T A_{J_2^k} d^k(J_2^k) = \operatorname{sgn}(c^k(J_2^k)). \tag{30}$$

b. Compute step size γ^k.

As scalar γ increases from 0 along direction d^k, the solution $x^k + \gamma d^k$ may violate the optimal conditions. There exist four cases.

First, an element of vector $-A^T_{J^k_3}(A(x^k+\gamma d^k)-r)$ may increase in magnitude beyond λ^k, violating (26). This case occurs when γ equals

$$\min\left\{\frac{\lambda^k-c^k(i)}{1-a_i^T v^k},\ \frac{\lambda^k+c^k(i)}{1+a_i^T v^k}\ \middle|\ i\in J^k_3\right\}\cap\mathbb{R}_+, \tag{31}$$

where $v^k=Ad^k=A_{J^k_2}d^k(J^k_2)$ and $\mathbb{R}_+$ denotes the set of positive real values. Denote γ^k_+ the solution of (31) and i^k_+ the index minimizing Problem (31).

The second case is that a coordinate of $x^k(J^k_2)+\gamma d^k(J^k_2)$ crosses zero. It occurs when γ equals

$$\min\{-x^k(i)/d^k(i)\mid i\in J^k_2, d^k(i)\neq 0\}\cap\mathbb{R}_+. \tag{32}$$

Denote γ^k_- the solution of (32) and i^k_- the index minimizing Problem (32).

The third case is that a coordinate of $x^k(J^k_2)+\gamma d^k(J^k_2)$ reaches constraint $|x^k_i+\gamma d^k(i)|\le\kappa$. It occurs when γ equals

$$\min\left\{\tfrac{\kappa-x^k(i)}{d^k(i)},\ \tfrac{-\kappa-x^k(i)}{d^k(i)}\ \middle|\ i\in J^k_2,\qquad d^k(i)x^k(i)\ge 0,\ d^k(i)\neq 0\right\}\cap\mathbb{R}_+. \tag{33}$$

Denote γ^k_{--} the solution of (33) and i^k_{--} the index minimizing Problem (33).

The fourth case is that the magnitude of a coordinate of $-A^T_{J^k_1}(A(x^k+\gamma d^k)-r)$ decreases to $\lambda^k-\gamma$, violating Eq. (24). It occurs when γ equals

$$\begin{aligned}\min\ &\left\{\left\{\frac{\lambda^k-c^k(i)}{1-a_i^T v^k}\ \middle|\ c^k(i)>0,\ a_i^T v^k>1,\ i\in J^k_1\right\}\right.\\ &\left.\cup\left\{\frac{\lambda^k+c^k(i)}{1+a_i^T v^k},\ \middle|\ c^k(i)<0,\ a_i^T v^k<-1\ i\in J^k_1\right\}\right\}\cap\mathbb{R}_+.\end{aligned} \tag{34}$$

Denote γ^k_{++} the solution of (34) and i^k_{++} the index minimizing Problem (34).

Moreover, the algorithm terminates when λ^k reaches the threshold λ. Hence, we have $\gamma^k\le\lambda^k-\lambda$. In summary, γ^k is determined by

$$\gamma^k=\min\{\gamma^k_+,\gamma^k_-,\gamma^k_{--},\gamma^k_{++},\lambda^k-\lambda\}, \tag{35}$$

and active index set J^k and index sets J^k_1, J^k_2 and J^k_3 are updated by

$$\begin{cases} J^k_1\xrightarrow{i^k_{++}}J^k_2, & \text{if } \gamma^k=\gamma^k_{++},\\ J^k_1\xleftarrow{i^k_{--}}J^k_2, & \text{if } \gamma^k=\gamma^k_{--},\\ J^k_2\xrightarrow{i^k_-}J^k_3,\ J^k=J^k\backslash\{i^k_-\} & \text{if } \gamma^k=\gamma^k_-,\\ J^k_2\xleftarrow{i^k_+}J^k_3,\ J^k=[J^k\ i^k_+] & \text{if } \gamma^k=\gamma^k_+,\end{cases} \tag{36}$$

where $[J^k\ i^k_+]$ represents an index set obtained by appending index i^k_+ at the end of J^k. The transition between index sets J^k_j, $j=1,2,3$ is shown in Fig. 4.

The updated index sets are J^{k+1}, J^{k+1}_1, J^{k+1}_2 and J^{k+1}_3, respectively. Then, we calculate λ^{k+1} by

$$\lambda^{k+1}=\max\{|c^{k+1}(i)|\mid i\in J^{k+1}_2\cup J^{k+1}_3\}, \tag{37}$$

Fig. 4 Transition between index sets in GH algorithm

$$\begin{tikzcd} J_1^k \arrow[r, harpoon, shift left, "i_{++}^k"] & J_2^k \arrow[l, harpoon, shift left, "i_{--}^k"] \arrow[r, harpoon, shift left, "i_{-}^k"] & J_3^k \arrow[l, harpoon, shift left, "i_{+}^k"] \end{tikzcd}$$

Stopping criteria As the HCS algorithm aims to find a certain number of related users and items, the iteration of the GH method may stop when a threshold is reached. Let l_0 be the number of selected columns in A. The criteria when GH is called by HCS algorithm are

Criterion. For a given l_0 $(0 < l_0 \leq l)$, $|J^k| = l_0$.

With this criterion, parameter λ in Eq. (35) is set as 0.

The GH algorithm for constrained ℓ_1-RLS problem (18) is summarized in Algorithm 2.

Algorithm 2 Generalized homotopy algorithm

```
Input: $A, r$;
Output: solution $x$ and active index set $J$

1: Initialization. Set $k = 0$ and $x^0 = 0$. Initialize $J_i^0$, $i = 1, 2, 3$, by Eq. (22), and $\lambda^0$ by (27).
2: while (stopping criterion is not satisfied) do
3:   Update direction $d^k$ by solving Eq. (30).
4:   Determine step size $\gamma^k$ by solving Eq. (35).
5:   Calculate $x^{k+1}$ by (28), update $J_1^k$, $J_2^k$, $J_3^k$ and $J^k$ by (36), and calculate $\lambda^{k+1}$ by (37).
6:   Set $k := k + 1$.
7: end while
8: Set $J = J_1^k \cup J_2^k$; $x = x^k$.
```

The computational complexity of GH at each iteration is $O(dl)$, where d and l are the dimensions of matrix A. Given that GH algorithm terminates within k steps, then the computational complexity of GH is $O(kdl)$. Note that the HCS algorithm starts with a p_0-by-q_0 rating matrix and calls GH twice at each iteration. Furthermore, the number of nonzeros, k, in the optimal solution given by GH is bounded by $\min\{p_0, q_0\}$, and ite_{HCS} is bounded by a constant. Hence, the total complexity of HCS is $O(\min\{p_0^2 q_0, p_0 q_0^2\})$.

In computation of relatedness, HCS and GH work on the relatively small-sized submatrix R_0, no matter what size the original user–item matrix R is. Moreover, as user–item pairs are independent, the computation of relatedness can be implemented in parallel. Hence, the relatedness computation model can be scaled to large-scale systems.

6 Applications of relatedness metric in CF systems

In this section, we investigate the application of the relatedness metric in CF systems.

Different from existing combination methods, we propose the data-oriented combination (DOC) method in which a particular CF approach is selected for each user–item pair according to its relatedness value. We categorize user–item pairs into three groups and call them white, gray and black sheep, originally named in [25]. In this paper, a user–item pair is defined as a white sheep if its relatedness value is larger than a threshold, a black sheep if its relatedness value is smaller than a threshold, and a gray sheep if in between. Most CF methods do not perform well due to the lack of local information on black sheep, while they can generate accurate predictions on white sheep due to sufficient local information.

The contribution of each CF method to the prediction of the DOC method is computed by minimizing the sum of differences between observations and predictions over a set of training data, called weight-training set. Let $\Omega = \{(u, i) | u \in U, i \in I\}$ denote the index set of all user–item pairs and $\Omega_0 \subset \Omega$ be the index set of weight-training set. Given K CF methods $P_k, k = 1, \ldots, K$, the weight w_k from P_k is computed by solving the unconstrained ℓ_1-RLS problem

$$\min_{w} \sum_{(u,i)\in\Omega_0} \left(R(u, i) - \sum_{k=1}^{K} \hat{R}_k(u, i) w_k \right)^2 + \mu \|w\|_1, \tag{38}$$

where $\hat{R}_k(u, i)$ is the predicted rating for (u, i) given by P_k, $w = (w_1, \ldots, w_K)$, and $\mu > 0$. We add constraint $\|w\|_1$ in (38) to prevent that a single CF method dominates the final prediction.

For comparison with the DOC method, we propose a basic combination (BC) method by linearly combining K CF methods with w_k. The prediction of the BC method for user–item pair (u, i) is

$$\hat{R}(u, i) = \sum_{k=1}^{K} \hat{R}_k(u, i) w_k. \tag{39}$$

For the white sheep in a weight-training set, we train combination weights in (38) and calculate ratings $\hat{R}(u, i)$ using (39) based on the training weights. Similarly, we calculate ratings $\hat{R}(u, i)$ on gray and black sheep, respectively. This prediction method is defined as BC-sep.

DOC is defined as the best prediction chosen from K benchmark CF methods, BC and BC-sep method on white, gray and black sheep, i.e.,

$$\hat{R}(u, i) = \begin{cases} \hat{R}_h(u, i), & \text{if white sheep;} \\ \hat{R}_g(u, i), & \text{if gray sheep;} \\ \hat{R}_b(u, i), & \text{if black sheep;} \end{cases} \tag{40}$$

where h, g and b denote the most accurate method for white, gray and black sheep, respectively. The accuracy of a CF method on a certain group of sheep is calculated and compared on an independent set of user–item pairs (excluding the weight-training set).

7 Experimental studies

In this section, we validate the effectiveness of the metric relatedness and the efficiency of the relatedness-based CF algorithm by comparing their performance with other benchmark CF methods. The experiments were conducted on a PC with Intel Core i7-4700 CPU, 3.40 GHz $\times$ 4, and 8 Gb RAM.

7.1 Experiment setup

The experiment study is conducted on the MovieLens [35], EachMovie [8] and Yahoo! Music [45] datasets. MovieLens dataset contains 100,000 ratings given by 943 users on 1682 movies, and EachMovie dataset contains 2,811,983 ratings given by 72,916 users on 1628 movies. Yahoo! Music is a larger dataset which contains 115,579,440 ratings given by 1,948,882 users on 98,213 artists. In the EachMovie dataset, 136,614 ratings of 2155 users rated at least ten movies are randomly chosen for training and test. In the Yahoo! Music dataset, 10,203,139 ratings given by 100,000 users on 17,569 artists in which each user rates at least 20 artists are randomly selected for training and test. The rating system of MovieLens uses a 1–5 scale, and EachMovie uses 0–5 scale, and Yahoo! Music uses 0–100 scale.

To distinct the zero ratings and missing ratings on EachMovie and Yahoo! Music datasets, the ratings are transformed to 1–6 and 1–101 star scale on EachMovie and Yahoo! Music datasets. We split the total ratings into a training set and a test set on all the three datasets. The test set is formed by randomly selecting ten ratings from each user on MovieLens and EachMovie, and one rating from each user on Yahoo! Music, and the training set is composed of the remaining ratings. The training set is split into the weight-training and model-training sets according to 10–90 % ratio on MovieLens and EachMovie, and 2–98 % ratio on Yahoo! Music dataset. The model-training set is used for training CF algorithms to compute predictions for active user–item pairs and the weight-training set for their weights in the DOC method. Also, the efficiency of the relatedness is validated by showing the relationship between the relatedness value and performance of CF methods on the weight-training set. The test set is split into test-1 and test-2 set according to 50–50 % ratio. The efficiency of the DOC method is examined on the test set.

Metrics and parameter settings Mean absolute error (MAE) [5,34] is a widely used metric to evaluate the performance of CF algorithms. It is defined as

$$\text{MAE} := \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} |R(u,i) - \hat{R}(u,i)|.$$

The criterion for chosen parameters is that they maximize the value of sep for a given set of user–item pairs Ω. The metric sep is defined as

$$sep = (\text{MAE}(\Omega_2) - \text{MAE}(\Omega_1))/(\text{r\_right} - \text{r\_left}), \tag{41}$$

where r_right and r_left indicate the maximum and minimum rating in the rating system, respectively, $\Omega_1 \subseteq \Omega$, $\Omega_2 \subseteq \Omega$, MAE($\Omega_2$) and MAE($\Omega_1$) represent MAEs on Ω_2 and Ω_1, respectively. Note that Ω is divided into Ω_1 and Ω_2 with equal number of user–item pairs, and the relatedness value of any user–item pair in Ω_1 is larger than that of any user–item pair in Ω_2. The value of sep defines the interdifference of relatedness over Ω_1 and Ω_2. A large value of sep indicates a good set of parameters for the relatedness metric.

We have done a series of parameter tests over MovieLens, EachMovie and Yahoo! Music, and the results are summarized in Table 2, where $c > 0$ is a parameter for tuning κ_α and κ_β. We define

$$\kappa_j = c \cdot \max\left\{\frac{2}{l}, \frac{1}{3\sqrt{l}}\right\}, \quad j = \alpha, \beta$$

where l represents the number of columns of A in Eq. (11) when $j = \beta$, and the number of columns of A^T in Eq. (10) when $j = \alpha$. Also, columns of A and A^T, vector r_{i_0}, and r_{u_0} are normalized in the implementation.

Relatedness results over three datasets by using different parameter settings are summarized in Table 2. To reduce the computation, we randomly selected 5, 5 and 1 % user–item pairs from the weight-training set of MovieLens, EachMovie and Yahoo! Music, respectively. The experiment of each parameter setting runs 10 times. The mean values and the standard deviations of sep metric are listed in the column sep-I, sep-II and sep-III corresponding to MovieLens, EachMovie and Yahoo! Music, respectively. For instance, "0.047 ± 0.009" represents the mean sep value of the 10 runs is 0.047 and the standard deviation is 0.009.

Table 2 Evaluation of relatedness with different parameters

id	n_1	n_2	r_0	r_1	r_2	c	sep-I	time-I	sep-II	time-II	sep-III	time-III
1	10	60	1	0.8	0.6	1	0.047 ± 0.009	38.6 ± 4.8	0.069 ± 0.005	51.2 ± 1.0	0.084 ± 0.006	140.4 ± 2.4
2	20	60	1	0.8	0.6	1	0.048 ± 0.010	39.0 ± 5.1	0.071 ± 0.006	39.6 ± 0.6	0.086 ± 0.007	138.0 ± 4.5
3	40	60	1	0.8	0.6	1	0.047 ± 0.009	48.9 ± 4.4	0.074 ± 0.008	50.6 ± 1.4	0.088 ± 0.005	137.6 ± 3.2
4	10	120	1	0.8	0.6	1	0.047 ± 0.009	38.2 ± 5.0	0.070 ± 0.005	40.0 ± 0.9	0.084 ± 0.007	141.8 ± 4.8
5	20	120	1	0.8	0.6	1	0.047 ± 0.010	45.2 ± 4.9	0.071 ± 0.005	52.9 ± 0.8	0.086 ± 0.008	143.0 ± 4.5
6	40	120	1	0.8	0.6	1	0.047 ± 0.010	38.4 ± 4.9	0.070 ± 0.006	42.1 ± 1.1	0.086 ± 0.006	145.8 ± 3.8
7	20	60	0.8	0.6	0.4	1	0.048 ± 0.007	49.5 ± 5.4	0.071 ± 0.007	48.8 ± 0.8	0.086 ± 0.004	132.9 ± 2.6
8	20	60	0.7	0.6	0.35	1	0.049 ± 0.006	42.7 ± 4.5	0.069 ± 0.005	39.0 ± 0.7	0.087 ± 0.006	138.7 ± 2.2
9	20	60	0.4	0.35	0.3	1	0.044 ± 0.007	51.3 ± 4.7	0.069 ± 0.006	51.8 ± 1.1	0.085 ± 0.006	140.7 ± 3.1
10	40	150	0.8	0.6	0.4	1	0.047 ± 0.010	39.4 ± 4.6	0.072 ± 0.007	37.6 ± 0.8	0.086 ± 0.008	133.3 ± 2.8
11	60	150	0.8	0.6	0.4	1	0.047 ± 0.009	39.6 ± 5.1	0.072 ± 0.005	50.0 ± 1.1	0.086 ± 0.007	132.9 ± 3.8
12	80	150	0.8	0.6	0.4	1	0.050 ± 0.009	47.1 ± 4.8	0.069 ± 0.005	40.3 ± 0.6	0.087 ± 0.006	142.3 ± 3.1
13	100	150	0.8	0.6	0.4	1	0.049 ± 0.009	40.6 ± 5.3	0.072 ± 0.004	49.1 ± 3.6	0.089 ± 0.006	138.8 ± 4.2
14	40	200	0.8	0.6	0.4	1	0.047 ± 0.010	46.6 ± 4.9	0.072 ± 0.007	38.0 ± 0.5	0.083 ± 0.005	136.4 ± 2.8
15	60	200	0.8	0.6	0.4	1	0.048 ± 0.009	38.7 ± 5.0	0.073 ± 0.005	50.8 ± 0.5	0.084 ± 0.005	137.8 ± 3.3
16	80	200	0.8	0.6	0.4	1	0.048 ± 0.009	45.7 ± 4.4	0.071 ± 0.006	40.4 ± 0.9	0.085 ± 0.007	138.7 ± 4.8
17	100	200	0.8	0.6	0.4	1	0.048 ± 0.009	38.6 ± 5.8	0.071 ± 0.006	52.5 ± 1.2	0.085 ± 0.006	141.8 ± 2.7
18	60	160	0.8	0.6	0.4	1	0.048 ± 0.009	45.1 ± 5.0	0.072 ± 0.005	38.6 ± 0.9	0.085 ± 0.006	134.9 ± 3.0
19	60	160	0.7	0.6	0.35	1	0.047 ± 0.009	41.4 ± 5.5	0.071 ± 0.006	50.4 ± 0.7	0.087 ± 0.006	134.8 ± 3.8
20	60	160	0.4	0.35	0.3	1	0.046 ± 0.012	48.5 ± 5.0	0.070 ± 0.010	36.8 ± 0.9	0.087 ± 0.006	133.8 ± 2.8
21	20	60	1	0.8	0.6	0.2	0.011 ± 0.005	43.2 ± 4.2	-0.001 ± 0.005	38.3 ± 1.0	-0.038 ± 0.010	203.5 ± 7.0
22	20	60	1	0.8	0.6	10	0.038 ± 0.008	29.1 ± 5.2	0.060 ± 0.007	34.9 ± 0.9	0.073 ± 0.007	72.3 ± 1.5
23	60	160	0.8	0.6	0.4	0.2	0.010 ± 0.006	43.1 ± 5.2	-0.003 ± 0.005	49.9 ± 0.9	-0.029 ± 0.010	206.3 ± 6.1
24	60	160	0.8	0.6	0.4	10	0.039 ± 0.008	29.6 ± 5.0	0.058 ± 0.008	23.1 ± 0.5	0.071 ± 0.006	72.3 ± 1.3

sep-I, *sep*-II and *sep*-III denote separation values on MovieLens, EachMovie and Yahoo! Music, respectively
time-I, time-II and time-III denote elapsed time (s) on MovieLens, EachMovie and Yahoo! Music, respectively

Table 3 Parameters of the BellKor algorithm

		λ_6	λ_7	λ_8	γ_1	γ_2	γ_3	f	ite
I	BellKor01	0.005	0.005	0.02	0.007	0.05	0.01	10	5
	BellKor07	0.005	0.005	0.02	0.007	0.05	0.07	10	5
	BellKor007	0.005	0.005	0.05	0.007	0.05	0.007	7	3
II	BellKor01	0.005	0.003	0.003	0.007	0.01	4.0E-6[a]	8	4
	BellKor07	0.005	0.005	0.02	0.007	0.007	0.07	10	5
	BellKor007	0.005	0.005	0.05	0.007	0.007	0.007	7	3
III	BellKor01	0.005	0.045	0.003	0.002	0.002	0.0005	10	5
	BellKor07	0.005	0.045	0.003	0.002	0.002	0.0002	10	5
	BellKor007	0.005	0.045	0.003	0.002	0.002	$5.0E$-5	10	5

I: MovieLens dataset; II: EachMovie dataset; III: Yahoo! Music dataset;
ite: the iteration number;
[a] On EachMovie set, BellKor01 sets γ_3 as $4.0E$-6, but still follows the name "BellKor01"

The mean values and standard deviations of the elapsed time are listed in the column $time$-I, $time$-II and $time$-III. The conclusion is that the results on three datasets vary in a very small range under setting with $c = 1.0$ and other parameters satisfy $1 > r_0 > r_1 > r_2 > 0$, $n_2 > n_1 > 0$, and $n_2 r_2 > n_1 r_1$. As we can see, under $c = 1.0$, the mean sep values on various parameter settings differ in a range of 0.046–0.050 on MovieLens. Inasmuch as the range of the standard deviation is 0.006–0.012, there is little change in the sep values with different parameter settings under $c = 1.0$. Similarly, the mean values of sep change little with different parameter settings on EachMovie and Yahoo! Music. Particularly, the mean values vary in the range of 0.069–0.074 on Eachmovie and 0.083–0.089 on Yahoo! Music.

Table 2 also lists different sizes of c values: large $c = 10.0$, medium $c = 1.0$, and small $c = 0.2$. When $c = 10.0$, the values of κ_α and κ_β in model (13) are large. In this case, constraints $\|\alpha\|_\infty \leq \kappa_\alpha$ and $\|\beta\|_\infty \leq \kappa_\beta$ are less likely active. On the other hand, when $c = 0.2$, many users or items with low correlation are included into the community, resulting in low sep values. An ideal case occurs when $c = 1.0$. In this case, the constraints are relatively tight, and the sep values are higher than $c = 10.0$ and $c = 0.2$ on all the three data sets, indicating the effect of the constraints $\|\alpha\|_\infty \leq \kappa_\alpha$ and $\|\beta\|_\infty \leq \kappa_\beta$ of the model (13) for searching the community.

The DOC method and benchmark methods We consider ten benchmark CF methods in the experiments. The first three methods are defined by choosing different parameters in the Netflix Prize winner model BellKor [22]. The selected parameters are listed in Table 3, where the three methods are named as BellKor01, BellKor07 and BellKor007 according to $\gamma_3 = 0.01$, $\gamma_3 = 0.07$ and $\gamma_3 = 0.007$.

The next two are nNbr-user [16] and nNbr-item methods [37]. These two methods predict the rating of the active user–item pair based on a set of similar users or items. The third group of CF methods calculate the prediction based on the SVD technique [22]. By implementing the SVD algorithm, a by-product named baseline prediction [22] is obtained.

We also include two basic methods, User-Mean and Item-Mean prediction, which use the average rating of users and items as the prediction, respectively. The last one named "Rel-based" is obtained by implementation of the HCS (see the note of the HCS algorithm at the end of Sect. 4).

Table 4 Regression coefficients

	I	II	III		I	II	III
SVD	0.391	0.476	0.1321	nNbr-item	0.179	0.215	0.1014
BellKor01	0.120	−0.177	0.2028	Baseline	0.244	−0.178	0.0473
BellKor07	−0.092	0.156	0.1516	Rel-based	0.228	0.189	0.1879
BellKor007	−0.086	0.259	0.3642	User-Mean	−0.074	−0.105	−0.2352
nNbr-user	0.122	0.135	0.0935	Item-Mean	−0.025	0.017	−0.0524

I: MovieLens dataset; II: EachMovie dataset; III: Yahoo! Music dataset

The DOC method is constructed based on linearly combining the 10 benchmark CF methods. The weights of these methods in the DOC method are determined by regression coefficients in Eq. (38) over the weight-training set, where μ is a small positive scalar. Particularly, we choose $\mu = 1.0 \times 10^{-4}$. The weights are listed in Table 4.

7.2 Performance evaluation

7.2.1 Effectiveness of relatedness

Relationship between relatedness and MAE As the relatedness is defined to reflect the prediction potential of a user–item pair, data with high quality should have large relatedness values and those with low quality have small ones. The validation of the effectiveness of relatedness was conducted on the weight-training set, in which the relatedness of each user–item pair is computed. The whole range of relatedness was divided into 23 small intervals, and each user–item pair is assigned to the corresponding interval according to its relatedness value. Since few user–item pairs have relatedness values <2.0, we grouped all of them into interval $(0, 2.0]$. Also, we grouped the pairs with relatedness >4.5 into the interval $[4.5, \infty)$. The other 21 intervals were equally divided from 2.0 to 4.5. The distribution of the number of user–item pairs in each relatedness interval on MovieLens dataset is shown in Fig. 5a. We count the number of user–item pairs in the same way as EachMovie. Its distribution is shown in Fig. 5c.

We computed the MAEs of five benchmark CF methods on each interval, and the results are shown in Fig. 5b (on MovieLens) and Fig. 5d (on EachMovie). The five chosen CF methods all have MAEs <0.75 and 0.94 on MovieLens and EachMovie, resp., and hence, their predictions are acceptable. As we can see, in general, the MAE decreases as the relatedness value increases, showing that user–item pairs with large relatedness value have better prediction performance. The decreasing pattern is quite steady when $\Gamma \geq 2.8$ on both of the two datasets. When $\Gamma < 2.8$, there are fewer user–item pairs in each interval and, thus, the average MAE may be largely influenced by a single abnormal sample.

We call the user–item pairs with the largest 30 % relatedness values white sheep, the lowest 30 % black sheep, and the rest of them gray sheep. Figure 6 shows the prediction performance of nNbr-user on three groups of user–item pairs over the weight-training set. The MAEs on white sheep in the two datasets are 17.6 and 26.4 % lower than those on gray sheep, respectively, and they are 31.3 and 41.5 % lower than those on black sheep.

The MAE was also computed by using the nNbr-user method on each user–item pair, and the statistics of MAE on white, gray and black sheep are given in Fig. 7. As we can see, on MovieLens dataset (Fig. 7a), 50.0 % of white sheep lie in small MAE range (0, 0.5],

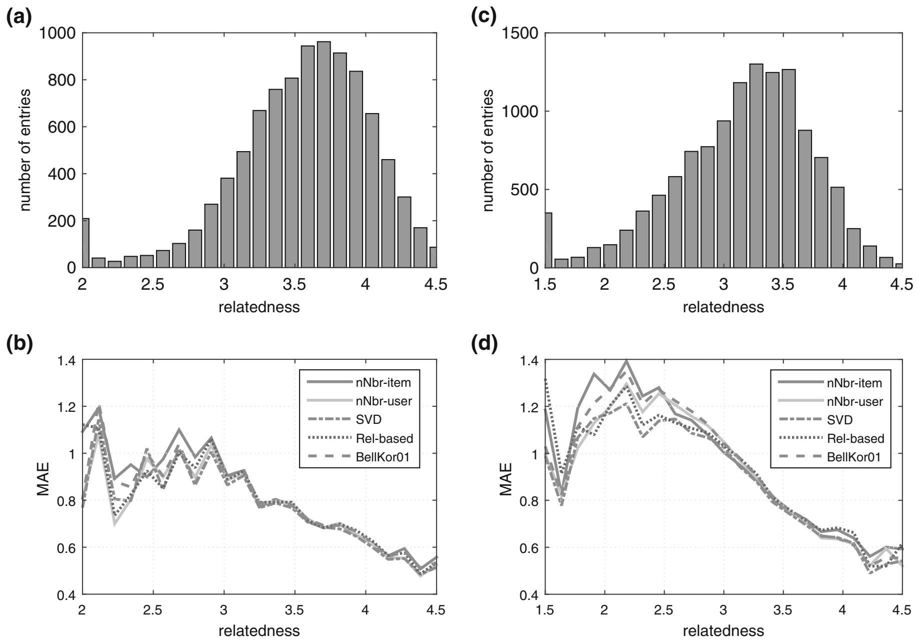
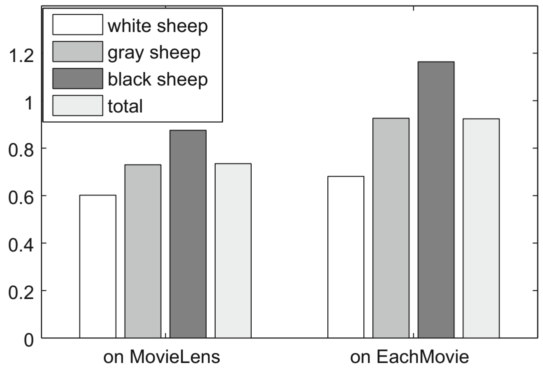


Fig. 5 Relationship between relatedness and MAE. **a** Distribution of the number of user–item pairs on MovieLens; **b** MAEs of five benchmark CF methods on MovieLens; **c** Distribution of the number of user–item pairs on EachMovie; **d** MAEs of five benchmark CF methods on EachMovie

Fig. 6 MAEs of nNbr-user on *white*, *gray* and *black sheep*



while 42.3 % of gray sheep and 35.2 % of black sheep lie in this range. On the other hand, only 4.9 % of white sheep lie in large MAE range (1.5,4.0], while 10.1 % of gray sheep and 17.2 % of black sheep lie in this range. Similar pattern can be observed on EachMovie dataset, where only 3.8 % of white sheep lie in large MAE range (2.0,5.0], while 9.6 % of gray sheep and 17.2 % of black sheep lie in this range. Hence, it is more likely to make accurate prediction on white sheep than on black sheep. We have similar statistics for other methods.

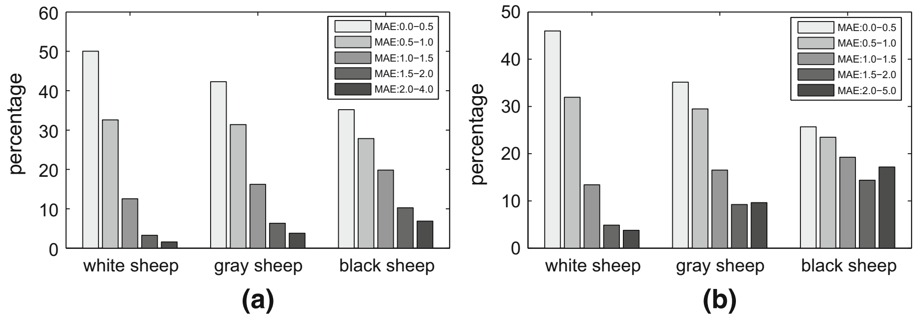


Fig. 7 Distribution of MAEs on *white*, *gray* and *black sheep*

Table 5 Comparison of relatedness with other metrics according to *sep*

		Relatedness	nNbr-user	nNbr-item	Corr-user	Corr-item	Reg-error
sep	MovieLens	0.0552	0.0270	0.0227	0.0433	0.0200	0.0334
	EachMovie	0.0632	0.0209	0.0372	0.0255	0.0229	0.0380
	Yahoo! Music	0.1371	0.0259	0.0371	0.0517	−0.0004	0.0279
Elapsed time	MovieLens	141.5	24.4	99.6	57.4	57.2	33.7
	EachMovie	230.8	60.5	194.8	72.4	72.5	49.3
	Yahoo! Music	2129.0	18974.5	2214.6	6937.7	7175.4	7937.3

Comparison of relatedness with other metrics We compared relatedness with five baseline metrics. First, for user–item pair (u_0, i_0), we use the mean Pearson similarities between user u_0 and k most similar users who have rated item i_0 as a measure for prediction capability of (u_0, i_0). This measure is denoted as "nNbr-user". Second, we use the mean Pearson similarities between item i_0 and k most similar items who have been rated by user u_0 as a measure for (u_0, i_0). This measure is denoted as "nNbr-item." The parameter k is set to 50 for both "nNbr-user" and "nNbr-item." The "significance weighting" technique described in [16] is incorporated in nNbr-user and nNbr-item. We use this technique to assign the similarity weights on ratings of nearest neighbors for calculating prediction.

The third and fourth baseline metrics are mean similarities calculated based on the user factors and the item factors trained by SVD. As stated in [13], user–user similarity can be computed using a vector similarity metric between the two users' interest factors. We calculate the user similarities by dot product of two user factors. For user–item pair (u_0, i_0), we choose k largest similar users as a neighborhood for the user u_0. The mean value of the similarities between user u_0 and the users in its neighborhood is calculated as a baseline metric, which is denoted as "corr-user." Similarly, the mean value of the similarities between item i_0 and the items in its neighborhood is calculated as another baseline metric. It is denoted as "corr-item." We choose the setting of parameter k as $k = 20$ for all the three datasets based on our numerical results. The last baseline metric is the shrinkage method, proposed in [3]. We named the metric "reg-error" due to the regression error used for reliability. The *sep* values and the elapsed time (in s) are calculated on test set and listed in Table 5.

As we can see, the relatedness outperforms other baseline metrics in terms of *sep* on all the three datasets. The relatedness metric works well because the prediction for an active user–item pair is made on a group of highly related users rating on a set of highly related items, and unreliable neighbors have been excluded. Particularly, on the large-scale Yahoo! Music dataset, the *sep* value of relatedness metric is significantly higher than other metrics. On the other hand, in the nNbr-user metric, the similarities of the neighbors of a user are calculated on the known ratings of all items. Although the significance weighting skill is incorporated, ratings on lots of the irrelevant items affect the accuracy of this metric, and it is easily seen on the large-scale dataset. Similar cases occur for the nNbr-item metric. The SVD-based corr-user and corr-item metrics are well known for catching the global rating pattern but poor performance at catching local rating information [Y. Koren 2008]. The "reg-error" baseline metric fits the ratings of an item by linearly combining the ratings of the items in its neighbors on each interested user–item pair. As it does not exclude ratings of unreliable users, this metric yields relatively low *sep* values, especially on large-scale Yahoo! Music dataset.

It is also shown in Table 5 that the computation time for relatedness metric is comparable to other baseline metrics. Particularly, on the large-scale Yahoo! Music dataset, the computation time for relatedness metric is the fastest. The elapsed time for relatedness is 2129.0 s, 10–30 % of four chosen baseline metrics. We also analyzed the main computational load of each baseline metric. As the major computation of the nNbr-user and nNbr-item metrics lies on calculating the pairwise similarities between items or users for determining the active user/item's neighborhood, their complexities are $O(m^2)$ and $O(n^2)$, respectively, where m denotes the total number of users and n denotes the total number of items. The major work of the corr-user and corr-item metrics is the training of the SVD model. On Yahoo! Music dataset, which contains $m = 100,000$ users, $n = 17,569$ items, and $f = 10$ attributes for each user and item, it takes the metric to train up to $(m+n)*f \approx 1.2 \times 10^6$ model parameters. Based on the nearest neighbor method, the reg-error metric also calculates pairwise similarities between items to determine the neighbors of each active item. For each interested user–item pair, this metric needs to construct a k-by-k coefficient matrix for solving a quadratic programming, which takes an order of $O(k^2|U|)$ calculation. Here, k denotes the neighborhood size and $|U|$ denotes the number of known ratings on the active item. The relatedness metric only computes a $|U|$-by-$|I|$ matrix ($|I|$ denotes the number of items given by the active user) at each active user–item pair, instead of the whole user–item matrix, which is a small-sized problem. More importantly, the relatedness metric can be easily scaled to the large-scale problem as the size of $|U|$-by-$|I|$ matrix does not increase much along with the size of the problem. This advantage has been shown in Yahoo! Music dataset (Table 5), where HCS is faster than all other baseline metrics.

7.2.2 Evaluation of the DOC method

On MAEs The DOC computes predictions according to formula (40), where the BC method is trained using weight-training set, and predictions of the 10 benchmark CF methods, BC and BC-sep method on three groups of sheep are compared on test-1 set. Test-2 set is used to evaluate the accuracy of DOC method.

We list DOC performance on both test-1 and test-2 to show that there are no overfitting and underfitting problems. Tables 6 and 7 show the performance of the DOC method and benchmark CF methods on MovieLens, while Tables 8 and 9 show the performance on EachMovie, and Tables 10, 11 on Yahoo! Music. All the benchmark methods reach lowest MAEs on white sheep and highest MAEs on black sheep, indicating the effectiveness of the relatedness metric. Among the 10 benchmark methods, the SVD CF method gives the best result on the test-1 and test-2 on both MovieLens and EachMovie, and the BellKor method (BellKor07) achieves the best performance on Yahoo! Music.

Table 6 MAE on various sheep (test-1, MovieLens)

	White	Gray	Black	Total MAE
SVD	0.6115	0.7113	0.8730	0.7299
BellKor01	0.6279	0.7203	0.8900	0.7435
BellKor07	0.6261	0.7207	0.8990	0.7458
BellKor007	0.6349	0.7370	0.9072	0.7574
nNbr-user	0.6223	0.7232	0.9093	0.7487
nNbr-item	0.6214	0.7315	0.9521	0.7647
Baseline	0.7229	0.7864	0.9477	0.8157
Rel-based	0.6378	0.7376	0.9147	0.7608
User-Mean	0.7172	0.7846	1.0187	0.8346
Item-Mean	0.6991	0.7927	1.0203	0.8329
BC-sep	0.6087	0.7107	0.8754	0.7295
BC	0.6048	0.7059	0.8789	0.7275
DOC	0.6048	0.7059	0.8730	0.7257

Table 7 MAE on various sheep (test-2, MovieLens)

	White	Gray	Black	Total MAE
SVD	0.6085	0.7481	0.8649	0.7413
BellKor01	0.6118	0.7588	0.8804	0.7512
BellKor07	0.6075	0.7612	0.8874	0.7530
BellKor007	0.6191	0.7823	0.8945	0.7670
nNbr-user	0.6237	0.7544	0.8971	0.7580
nNbr-item	0.6357	0.7516	0.9506	0.7765
Baseline	0.6935	0.8387	0.9293	0.8223
Rel-based	0.6286	0.7671	0.9188	0.7711
User-Mean	0.7059	0.8126	1.0000	0.8368
Item-Mean	0.6765	0.8431	0.9996	0.8401
BC-sep	0.6022	0.7447	0.8626	0.7373
BC	0.5999	0.7407	0.8703	0.7374
DOC	0.5999	0.7407	0.8649	0.7357

For the MovieLens dataset, the BC method outperforms SVD CF, except for the black sheep. On test-1, the MAE is reduced from 0.6115 to 0.6048 on white sheep, from 0.7113 to 0.7059 on gray sheep and from 0.7299 to 0.7275 on the whole set. Similarly, on test-2, the MAE is reduced from 0.6085 to 0.5999 on white sheep, from 0.7481 to 0.7407 on gray sheep and from 0.7413 to 0.7374 on the whole dataset. BC performs not as good as SVD CF on the black sheep due to the influence of other predictors. DOC overcomes the drawback of BC by using the predictions of SVD CF on black sheep, and it reduces total MAE to 0.7257 and 0.7357, respectively, on test-1 and test-2.

On the EachMovie dataset, BC outperforms SVD CF on all types of data. In test-1, the MAE is reduced from 0.7097 to 0.6936 on white sheep, from 0.9605 to 0.9417 on gray sheep, from 1.0374 to 1.0318 on black sheep and from 0.9084 to 0.8943 on the whole set. Similarly, in test-2, the MAE is reduced from 0.7027 to 0.6915 on white sheep, from 0.9520 to 0.9430 on gray sheep, from 1.0312 to 1.0284 on black sheep and from 0.9010 to 0.8932 on the whole set.

Table 8 MAE on various sheep (test-1, EachMovie)

	White	Gray	Black	Total MAE
SVD	0.7097	0.9605	1.0374	0.9084
BellKor01	0.8239	1.0899	1.3218	1.0797
BellKor07	0.8476	1.1072	1.3591	1.1049
BellKor007	0.7459	1.0205	1.1629	0.9809
nNbr-user	0.7248	0.9832	1.0888	0.9374
nNbr-item	0.7383	0.9795	1.1151	0.9478
Baseline	0.8353	1.1016	1.1524	1.0369
Rel-based	0.7319	0.9756	1.0742	0.9321
User-Mean	0.8943	1.1327	1.2957	1.1101
Item-Mean	0.8262	1.1190	1.2725	1.0772
BC-sep	0.6951	0.9446	1.0232	0.8934
BC	0.6936	0.9417	1.0318	0.8943
DOC	0.6936	0.9417	1.0232	0.8917

Table 9 MAE on various sheep (test-2, EachMovie)

	White	Gray	Black	Total MAE
SVD	0.7027	0.9520	1.0312	0.9010
BellKor01	0.8455	1.0893	1.3413	1.0918
BellKor07	0.8747	1.1068	1.3786	1.1187
BellKor007	0.7572	1.0271	1.1662	0.9879
nNbr-user	0.7140	0.9935	1.0721	0.9332
nNbr-item	0.7434	0.9799	1.1144	0.9493
Baseline	0.8403	1.1138	1.1441	1.0409
Rel-based	0.7297	0.9739	1.0657	0.9282
User-Mean	0.9031	1.1204	1.2951	1.1076
Item-Mean	0.8523	1.1192	1.2599	1.0814
BC-sep	0.6948	0.9437	1.0201	0.8919
BC	0.6915	0.9430	1.0284	0.8932
DOC	0.6915	0.9430	1.0201	0.8907

BC performs better than BC-sep on all except black sheep in EachMovie. The DOC method overcomes this drawback of the BC method in EachMovie by using the predictions of BC-sep on black sheep, and it reduces total MAE to 0.8917 and 0.8907, respectively, on test-1 set and test-2 set. As shown in Table 8, the MAEs given by the DOC, BC and SVD are 0.8917, 0.8943 and 0.9084 on test-1, respectively. Compared with SVD, the DOC method improves MAE by $0.9084 - 0.8917 = 0.0167$. As the MAE of DOC is 0.8943-0.8917 = 0.0026 better than that of BC, the relatedness metric contributes 0.0026/0.0167 = 15.6 %. Similarly, the relatedness metric contributes (0.9010-0.8907)/(0.8932-0.8907) = 24.3 % for the MAE improvement on test-2.

On the Yahoo! Music dataset, BC outperforms BellKor07 on all types of data. In test-1, the MAE is reduced from 7.7819 to 7.4412 on white sheep, from 17.6257 to 17.1003 on gray sheep, from 22.6327 to 21.9036 on black sheep and from 16.1747 to 15.6436 on the whole set. Similarly, in test-2, the MAE is reduced from 7.9063 to 7.6326 on white sheep,

Table 10 MAE on various sheep (test-1, Yahoo! Music)

	White	Gray	Black	Total MAE
SVD	7.3500	18.1184	24.1716	16.7038
BellKor01	7.9825	17.5948	22.5032	16.1836
BellKor07	7.7819	17.6257	22.6327	16.1747
BellKor007	8.4590	17.7861	22.2994	16.3420
nNbr-user	11.5280	18.2908	28.8614	19.4331
nNbr-item	6.1684	18.4196	26.9126	17.2921
Baseline	37.1581	36.0425	34.7549	35.9909
Rel-based	5.0635	17.8035	26.0633	16.4594
User-Mean	5.8702	20.6250	31.6829	19.5159
Item-Mean	29.0360	28.8027	36.0761	31.0547
BC-sep	7.4412	17.1003	21.9036	15.6436
BC	5.8486	17.3401	21.9295	15.2695
DOC	5.0635	17.1003	21.9036	14.9303

Table 11 MAE on various sheep (test-2, Yahoo! Music)

	White	Gray	Black	Total MAE
SVD	7.4754	18.0497	24.5131	16.8164
BellKor01	8.1118	17.6177	22.8100	16.3237
BellKor07	7.9063	17.5843	23.0364	16.3165
BellKor007	8.6666	17.8376	22.7389	16.5567
nNbr-user	11.6371	18.1556	28.8871	19.4195
nNbr-item	6.3348	18.2718	26.7914	17.2466
Baseline	37.1961	35.8864	34.9376	35.9946
Rel-based	5.1844	17.6720	26.3422	16.5268
User-Mean	6.0334	20.4014	31.6559	19.4673
Item-Mean	29.2102	28.7549	36.4450	31.1985
BC-sep	7.6326	17.0964	22.3259	15.8261
BC	6.0050	17.3838	22.3698	15.4660
DOC	5.1844	17.0964	22.3259	15.0917

from 17.5843 to 17.0964 on gray sheep, from 23.0364 to 22.3259 on black sheep and from 16.3165 to 15.8261 on the whole set.

The performance of BC is not better than the Rel-based method on white sheep in Yahoo! Music. The DOC method improves the performance further by using the predictions of the Rel-based method on white sheep, and it reduces total MAE to 14.9303 and 15.0917, respectively, on test-1 set and test-2 set. As shown in Table 10, the MAEs given by the DOC, BC and BellKor07 are 14.9303, 15.2695 and 16.1747 on test-1, respectively. Compared with BellKor07, the DOC method improves MAE by 16.1747-14.9303 = 1.2444. As the MAE of DOC is 15.2695-14.9303 = 0.3392 better than that of BC (basic linear combination), the relatedness metric contributes approximately 0.3392/1.2444 = 27.3 %. Similarly, the relatedness metric contributes (15.4660-15.0917)/(16.3165-15.0917) = 30.6 % for the MAE improvement on test-2.

8 Related work

CF methods based on local and global information CF methods are usually categorized into memory-based and model-based methods [5]. Memory-based CF methods, such as nNbr-user [34] and nNbr-item [37], exploit local rating information (neighbors' ratings) to predict the active user's preference. In contrast, model-based CF methods first develop a model based on historical data and then use the model to predict new preferences for active users. With global rating information, model-based methods usually perform better than memory-based methods on sparse dataset. Recently, some model-based CF methods [22] applied the SVD technique to exploit the global rating information for recommendation by characterizing both items and users in the same latent factor space. As a variant of the SVD method, the matrix factorization method decomposes the user–item rating matrix into two matrices consisting of user feature vectors and item feature vectors. This method is apt to incorporate additional information, and it improves prediction accuracy on black sheep by integrating local and global rating information [23,41].

Integration of multiple CF methods A number of CF systems combine multiple methods to overcome disadvantages of a particular one. For example, the bagging method [6] uses the average of multiple predictions generated by sampling a training set several times. The adaptive boosting [14], a variant of the committee method, repeatedly runs a given learning algorithm, trains multiple classifiers in sequence and then combines these classifiers into a composite one for prediction. The decision tree framework [7] selects one from a group of models for each active user–item pair. The linear regression method [4] linearly combines predictions generated by multiple CF models and employs ridge regression to obtain the final prediction. In particular, the BigChaos [42] has shown its success on improving prediction accuracy for the Netflix Grand Prize by combining 101 CF predictors.

Confidence measure The confidence metric was proposed in [21] to indicate the support for a predicted rating by estimating the likelihood of a predicted rating being correct. The confidence on the prediction has also been studied under a probabilistic framework in [18]. Although the probability distribution over the likelihood of predicted ratings can be leveraged to analyze the algorithmic confidence, the probabilistic method requires a priori probability distribution. In addition, fairness of a measure beyond statistical framework was presented in [24], where bias and controversy were discussed.

9 Conclusion and future work

This paper introduced a novel metric, called relatedness, to reflect how well a user preference on an item can be correctly predicted. According to the relatedness value, we categorized user–item pairs into three groups: white, gray and black sheep, corresponding to "good," "middle" and "bad" user–item pairs, respectively. The relatedness played critical roles in the design of the DOC CF method, in which a particular CF method was chosen for a user–item pair. We have performed a comprehensive evaluation on effectiveness of the relatedness and the performance of the DOC method over MovieLens, EachMovie and Yahoo! Music datasets. We have showed that the DOC method improved the prediction performance in terms of MAE metric. The computation of relatedness was formulated as a max–max optimization problem. We have presented the HCS method to efficiently find a good community for computing the relatedness of a user–item pair.

To calculate relatedness metrics, explicit ratings are required. Currently, it could not employ implicit rating feedbacks or side information. Moreover, the relatedness metric needs a certain quantity of known ratings and is not suitable for cold-start problem.

For future work, we will extend the relatedness concept to very sparse datasets. As few ratings are available for a community in this case, the proposed relatedness metric may not be sufficient to measure prediction capability of the active user–item pair. Data sparsity remains a challenge for CF methods. Indirect and additional information [31,32] has been used to alleviate the sparsity problem. In SLIM [28,29], side information was considered to improve recommendation performance. Also, due to lack of substantial evidence on which items the customer dislike in some datasets, [19] and BPR [33] used implicit ratings for recommendation. We will investigate a new metric based on relatedness by exploring implicit rating feedbacks. Note that the numeric value of implicit feedback somehow indicates a certain level of confidence on a user–item pair. We attempt to characterize the relatedness of implicit feedbacks based on this observation.

References

1. Adomavicius G, Kamireddy S, Kwon, Y (2007) Towards more confident recommendations: improving recommender systems using filtering approach based on rating variance. In: Proceedings of the 17th workshop on information technology and systems
2. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowl Data Eng IEEE Trans 17(6):734–749
3. Bell R, Koren Y, Volinsky C (2007) Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, pp 95–104
4. Bell RM, Koren Y, Volinsky C (2008) The bellkor 2008 solution to the netflix prize. Statistics Research Department at AT&T Research
5. Breese JS, Heckerman D, Kadie C, et al (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th conference on Uncertainty in Artificial Intelligence, pp 43–52
6. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
7. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth, Boston
8. DEC Systems Research Center (1997) EachMovie 1997. http://www.research.digital.com/SRC/eachmovie/
9. Chen SS, Donoho DL, Saunders MA (1999) Atomic decomposition by basis pursuit. SIAM J Sci Comput 20(1):33–61
10. Donoho DL, Tsaig Y (2008) Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. Inf Theory IEEE Trans 54(11):4789–4812
11. Donoho DL, Tsaig Y, Drori I, Starck JL (2012) Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. Inf Theory IEEE Trans 58(2):1094–1121
12. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. The Ann Stat 32(2):407–499
13. Ekstrand MD, Riedl JT, Konstan JA (2010) Collaborative filtering recommender systems. Found Trends Human–Comput Interact 4(2):81–173
14. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Thirteenth international conference on machine learning, pp 148–156
15. Gunawardana A, Shani G (2009) A survey of accuracy evaluation metrics of recommendation tasks. J Mach Learn Res 10:2935–2962
16. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, Berkeley, California, pp 230–237
17. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst (TOIS) 22(1):5–53
18. Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Trans Inf Syst (TOIS) 22(1):89–115

19. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Eighth IEEE international conference on data mining 2008. IEEE, pp 263–272
20. Jin R, Si L, Zhai C (2006) A study of mixture models for collaborative filtering. Inf Retri 9:357–382
21. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J (1997) GroupLens: applying collaborative filtering to Usenet news. Commun ACM 40(3):77–87
22. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434
23. Koren Y (2009) The bellkor solution to the netflix grand prize, Netflix prize documentation. http://www.netflixprize.com/
24. Lauw HW, Lim E-P, Wang K (2006) Bias and controversy: beyond the statistical deviation. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 625–630
25. McCrae J, Piatek A, Langley A (2004) Collaborative filtering. http://www.imperialviolet.org/
26. McLaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp 329–336
27. McNee SM, Lam SK, Guetzlaff C, Konstan JA, Riedl J (2003) Confidence displays and training in recommender systems. In: Proceedings of INTERACT'03 IFIP TC13 international conference on human–computer interaction, pp 176–183
28. Ning Xia, Karypis George (2011) Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th international conference on data mining (ICDM). IEEE, pp 497–506
29. Ning Xia, Karypis George (2012) Sparse linear methods with side information for top-n recommendations. In: Proceedings of the sixth ACM conference on recommender systems. ACM, pp 155–162
30. Osborne MR, Presnell B, Turlach BA (2000) A new approach to variable selection in least squares problems. IMA J Numer Anal 20(3):389–403
31. Papagelis M, Plexousakis D, Kutsuras T (2005). Alleviating the sparsity problem of collaborative filtering using trust inferences. In: iTrust. Springer, pp 224–239
32. Piccart B, Struyf J, Blockeel H (2010) Alleviating the sparsity problem in collaborative filtering by using an adapted distance and a graph-based method. In: Proceedings of the 2010 SIAM international conference on data mining. SIAM, pp 189–198
33. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 452–461
34. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on computer supported cooperative work, pp 175–186
35. Riedl J, Konstan J, Terveen L (2006) MovieLens. http://www.grouplens.org/node/73
36. Rockafellar RT (1970) Convex analysis. Princeton University Press, New Jersey
37. Sarwar B, Karypis G, Konstan J, Reidl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web. ACM, Hong Kong, pp 285–295
38. Sarwar B, Karypis G, Konstan J, Riedl J (2000) Analysis of recommendation algorithms for e-commerce. In: Proceedings of the 2nd ACM conference on Electronic commerce, pp 158–167
39. 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, pp 345–354
40. Symeonidis P, Nanopoulos A, Papadopoulos AN, Manolopoulos Y (2008) Nearest-biclusters collaborative filtering based on constant and coherent values. Inf Retr 11(1):51–75
41. Takács G, Pilászy I, Németh B, Tikk D (2009) Scalable collaborative filtering approaches for large recommender systems. J Mach Learn Res 10:623–656
42. Töscher A, Jahrer M, Bell RM (2009) The BigChaos solution to the Netflix grand prize. Netflix prize documentation. http://www.netflixprize.com/
43. Wang J, De Vries AP, Reinders MJT (2006) Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 501–508
44. Wu FQ, He L, Xia WW, Ren L (2008) A collaborative filtering algorithm based on users' partial similarity. In: 10th international conference on control, automation, robotics and vision, 2008. ICARCV 2008, pp 1072–1077
45. YahooMusic (2008) Yahoo! Music user ratings of musical artists, version 1.0. http://research.yahoo.com

46. Yang J, Wang H, Wang W, Yu P (2003) Enhanced biclustering on expression data. In: Proceedings third IEEE symposium on bioinformatics and bioengineering (BIBE "03), pp 321–327

Xijun Liang received the B.S. degree in School of Mathematical Science from China University of Petroleum, Dongying, in 2003, and the Ph.D. degree in mathematics from Dalian University of Technology in 2013. He is now a lecturer in the College of Science, China University of Petroleum, Qingdao, People's Republic of China. His research interests are in the area of large-scale optimization and machine learning.

Zhonghang Xia received the B.S. degree in applied mathematics from Dalian University of Technology and the Ph.D. degree in computer science from the University of Texas at Dallas. He is now a professor in the Department of Computer Science, Western Kentucky University, Bowling Green, KY. His research interests are in the area of bioinformatics, data mining and distributed systems.

Liping Pang received the B.S. degree and the M.S. degree in applied mathematics from Jilin University of Technology in 1990 and 1993, respectively, and the Ph.D. degree in mathematics from Dalian University of Technology in 2004. She is now a professor in the School of Mathematical Sciences, Dalian University of Technology, Dalian, People's Republic of China. Her research interests are in the area of nonsmooth optimization, stochastic programming and their applications.

Liwei Zhang is now a professor in School of Mathematical Sciences, Dalian University of Technology, People's Republic of China. He received his Ph.D from Dalian University of Technology in 1998. His research interests include conic optimization, stochastic programming and equilibrium optimization.

Hongwei Zhang received the B.S. degree and the M.S. degree in Department of Mathematics from Jilin University in 1982 and 1989, respectively, and the Ph.D. degree in mathematics from Dalian University of Technology in 2004. He is now a professor in the School of Mathematical Sciences, Dalian University of Technology, Dalian, People's Republic of China. His research interests are in the area of numerical optimization, nonsmooth optimization and numerical methods for differential equations.