Knowledge-Based Systems 27 (2012) 407–423

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# A careful assessment of recommendation algorithms related to dimension reduction techniques

Chun-Xia Yin *, Qin-Ke Peng

*School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China*

ARTICLE INFO

*Article history:*
Received 7 May 2011
Received in revised form 26 October 2011
Accepted 26 November 2011
Available online 4 December 2011

*Keywords:*
Information overload
Personalized recommendation
Collaborative filtering
Dimension reduction
Non-negativity constraints

ABSTRACT

Due to lack of detailed tests under the uniform test framework for existing personalized recommendation algorithms, the best performances claimed in literatures are hard to credit. For this reason, this paper presents a comparative evaluation of eight collaborative filtering (CF) algorithms, mainly focusing on recommendation algorithms related to dimension reduction techniques, on two common popular datasets by using three quality metrics. The eight algorithms are the *k*-nearest neighbor (KNN) algorithm, three native dimension-reducing algorithms respectively based on singular value decomposition (SVD), non-negative matrix factorization (NMF) and weighted non-negative matrix factorization, and four hybrid algorithms respectively crossing principal component analysis: PCA and KNN, SVD and KNN, NMF and KNN, and PCA and a recursive rectangular clustering. There are some interesting findings in our experiments. First, dimension-reducing techniques can help dig out more valuable information from the rating data than the nearest-neighbor technique. Second, in comparison with four hybrid algorithms, three native algorithms only based on dimension-reducing techniques are able to better satisfy users' actual needs. Third, dimension-reducing techniques with non-negativity constraints are more effective than not with non-negativity ones. Fourth, the decision on the optimum algorithm among eight algorithms is insensitive to the sparsity of dataset. Fifth, proper selections for appropriate values of parameters of algorithms are very often problem dependent. Sixth, native dimension-reducing algorithms can defeat these algorithms related to KNN in the processing time. These findings not only show that it is necessary to evaluate in detail these algorithms under the uniform test framework but also play an important role in the right navigation for further innovation research on the technology of the personalized recommendation.



## 1. Introduction

When gaining in convenience from the network, nowadays people are simultaneously suffering from information overload, that is, they face too much data and sources to find out those most relevant for him [1]. Search engine, which can only present the mainstream results to the target user, seems helpless when individual demands are distinct from the mainstream. To dig out these latent objects of interest to the target user, an explosion of studies on personalized recommendation came into being. It poses challenge to three parties: researchers, users and website entrepreneurs. Researchers should endeavor to find effective methods digging out these latent items of interest to the target user from vast amounts of resources. Users should have the patience to provide their personal information (such as profiles, ratings, friends and tags) used for helping researchers find out better methods. Website entrepreneurs should offer user friendly interfaces attracting and facilitating users to leave their personal information. To meet the personalized needs of users and to increase the economic interests of websites, three parties should give full cooperation to promote the development of PRT. Thus far, PRT has been applied to a wide range of fields such as book, movie, music, joke, food, travel, etc.

The recommendation algorithm is the soul of the recommendation system. There are currently four major branches of recommendation algorithms: content-based recommendation [2,3], collaborative filtering (CF) recommendation [4–8], recommendation based on network structure and graph theory (NSGT) [1,9–11] and hybrid recommendation [12–14]. Content-based algorithms require the structured information and therefore they are incompetent at unstructured items such as video, music, photograph, idea, etc. Both CF and NSGT algorithms effectively overcome content-based limitations because they mainly depend on the footprints of net users (e.g. rating information, transaction information and visit information) to excavate user interest. CF is the earliest and most successful technology applied to the recommendation system. NSGT, an extension of CF, analyzes and solves problems with the help of complex networks and graph theory

* Corresponding author.
*E-mail addresses:* dingdong1104@163.com, xia.1204@stu.xjtu.edu.cn (C.-X. Yin).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.11.022

methods. The hybrid algorithms blending two or multiple branches of recommendation algorithms attempt to improve the recommendation by overcoming the limitations of native branches of algorithms.

There are roughly three types of CF algorithms: memory-based [4,5,15,16], model-based [6,7,17–21] and hybrid [18,22,23]. The memory-based algorithms use the weighted average of opinions by users to produce the predictions. The *k*-nearest neighbor (KNN) algorithm is a typical example of this mechanism. The model-based algorithms recommend objects by using preference models learned from the opinions by users. Dimension-reducing models [6,17–20] are typical examples of this mechanism. The hybrid CF algorithms blending native CF approaches attempt to improve the predictions by overcoming the limitations of native CF approaches.

Currently there are three main problems about existing recommendation algorithms. First, almost every literature owns its own probe sets. However, the better algorithm on one data set may be worse on another data set with different ratings density, ratings scale and other properties. Second, evaluation metrics are not consistent, that is, different algorithms are often evaluated respectively by using different quality metrics. Much early evaluation work focused specifically on the "accuracy" of recommendation algorithms. However, the ranking of recommendation items sometimes have a larger effect on user satisfaction and performance. Obviously, the bottom-line measure of recommender system success should be user satisfaction. Third, parameters are not universally tested in detail. Though many new algorithms often appear to do better than the older algorithms they are compared to, when each algorithm is tuned to its optimum, they all produce similar measures of quality. As a result, the best performances claimed in literatures are hard to credit.

Therefore, it is very necessary to evaluate in detail existing recommendation algorithms under the uniform test framework, that is, the algorithm should be tested on the uniform common popular datasets by using the uniform quality metrics. Almost each original algorithm can be improved by adapting its original formulas or adding some additional information, so there exist a significant volume of variations of original recommendation algorithms. For example, a few similarity metrics proposed in [4,5,15,16] are the variations of original memory-based methods. For this reason, the representative and well-accepted algorithms should be selected to build a fair comparison framework. Ref. [16] evaluated three variants of memory-based CF algorithms to provide recommendations of articles on CiteULike. However, its representative and coverage for CF algorithms is not sufficient. This paper concentrates upon evaluating in detail eight CF algorithms, the representatives of three types of CF ones, on two popular common rating datasets by using three quality metrics. The eight algorithms contain one *k*-nearest neighbor (KNN) algorithm, three dimension-reducing algorithms respectively based on singular value decomposition (SVD), non-negative matrix factorization (NMF) and weighted non-negative matrix factorization (WNMF), and four hybrid algorithms respectively crossing principal component analysis: PCA and KNN, SVD and KNN, NMF and KNN, and PCA and a recursive rectangular clustering.

This paper presents six specific contributions towards evaluation of eight recommender algorithms.

(1) We introduce a fundamental framework of a rating recommendation system.
(2) We discuss in detail each of eight algorithms about the idea and parameters influencing its effectiveness.
(3) We review the properties of datasets selected for evaluation.
(4) We conceptually analyze three widely used evaluation metrics.
(5) We evaluate in detail each of eight algorithms by testing the representative values of each parameter and then report the results.
(6) We report some interesting findings by analyzing overall comparisons among eight algorithms.

The rest of the paper is structured as follows: Section 2 gives a simple description of a rating recommendation system. Section 3 elaborates eight algorithms with examples. Section 4 presents the evaluation criteria. Section 5 describes two benchmark data sets. Section 6 provides our experiment planning. Section 7 presents the experimental results for each algorithm and discusses the appropriate values of parameters (appropriate configurations) of each algorithm. Section 8 compares eight algorithms with appropriate configurations and discusses some interesting findings. Finally, Section 9 presents the most relevant conclusions and future work.

## 2. Simple description of a rating recommendation system

A rating recommendation system consists of $m$ users and $n$ objects, and each user has rated some objects. Denoting the user-set as $U = \{u_1, u_2, \ldots, u_m\}$ and object-set as $O = \{o_1, o_2, \ldots, o_n\}$, the recommendation system can be fully described by a rating matrix $A = \{a_{ij}\} \in R^{m,n}$, where $a_{ij}$ denotes the raw rating of $o_j$ by $u_i$ within a numerical scale. Each prediction is a numerical value, $p_{ij}$, expressing the predicted rating of $o_j$ for $u_i$. For a given user, a recommendation algorithm predicts the ratings of all the objects he/she has not rated before and then generates a ranking of these objects. Fig. 1 gives a simple example about the raw ratings of five users for eight objects. For convenience, we use "?" as the symbol of an unknown rating. In Fig. 1, each rating is an integer value in $[-5,5]$, so the minimum value $a_{\min}$, the maximum value $a_{\max}$ and the middle value are $-5$, 5 and 0, respectively.

Here, we introduce in advance three symbols, $A^{(i)}$, $A^j$ and max $Q$, to be used in this paper. The first two ones, respectively, stand for the $i$th row and $j$th column vectors of matrix $A$. The last one represents the maximum value in vector $Q$.

## 3. Algorithms

Next, we will elaborate eight algorithms according to three types of CF algorithms.

### 3.1. KNN algorithm

The KNN algorithm uses a weighted aggregate of the ratings of an unknown object by $k$ nearest users of the active user to generate the predicted rating of the object for the active user. Due to certain differences of interpretation of the rating scale for users, the predicted rating in the algorithm is generally computed as follows:

| | $o_1$ | $o_2$ | $o_3$ | $o_4$ | $o_5$ | $o_6$ | $o_7$ | $o_8$ |
|---|---|---|---|---|---|---|---|---|
| $u_1$ | -2 | ? | 5 | 1 | ? | 3 | ? | ? |
| $u_2$ | -1 | 3 | 2 | ? | 4 | ? | 5 | ? |
| $u_3$ | ? | ? | -4 | 2 | 4 | -2 | ? | 5 |
| $u_4$ | -2 | 2 | 4 | 2 | 5 | ? | ? | ? |
| $u_5$ | -3 | ? | -5 | 1 | 3 | 1 | ? | 4 |

**Fig. 1.** The raw ratings of five users for eight objects.

$$p_{ij} = \overline{u_i} + \frac{\sum_{u_l \in N_i} s_{il}(a_{lj} - \overline{u_l})}{\sum_{u_l \in N_i} s_{il}}, \quad (1)$$

where $\overline{u_l}$ is the average rating of $u_l$, $s_{il}$ represents the similarity between $u_i$ and $u_l$, and $N_i$ denotes the set of $k$ users most similar with $u_i$ among all the users having rating $o_j$.

The Pearson correlation coefficient, the most common similarity metric, measures the degree to which a linear relationship exists between two users. The correlation coefficient between $u_i$ and $u_l$, $c_{il}$, is computed as follows:

$$c_{il} = \frac{\sum_{o_j \in J}(a_{ij} - \overline{u_i})(a_{lj} - \overline{u_l})}{\sqrt{\sum_{o_j \in J}(a_{ij} - \overline{u_i})^2 \times \sum_{o_j \in J}(a_{lj} - \overline{u_l})^2}}, \quad (2)$$

where $J$ is the set of objects which have been rated by both $u_i$ and $u_l$. Obviously, when the correlation coefficient between two users is larger, the two users may be more similar. Here we directly use the correlation coefficient as the similarity, namely $s_{il} = c_{il}$. After computing the similarities between users, the predicted rating is calculated according to Eq. (1).

The KNN algorithm based on Pearson correlation coefficient is abbreviated as $p$KNN. Obviously, it is critical the appropriate decision on the value of $k$ for $p$KNN.

**Example 1.** Given the raw ratings in Fig. 1 and an active user $u_2$, if the value of $k$ is set to 2, $p$KNN predicts the rating of $o_6$ for $u_2$, $p_{26}$, as the following steps.

*The first step:* Find all these users having rated $o_6$. They are $u_1$, $u_3$ and $u_5$.
*The second step:* Compute the similarity between $u_2$ and each of $u_1$, $u_3$ and $u_5$ according to formula 2. Here, $\overline{u_1} = 1.75$, $\overline{u_2} = 2.6, \overline{u_3} = 1, \overline{u_5} = 0.17, c_{21} = s_{21} = 0.64, c_{23} = s_{23} = 0.71$ and $c_{25} = s_{25} = 0.71$.
*The third step:* Select $k = 2$ users with maximum similarities as the neighbors of $u_2$. Obviously, $u_3$ and $u_5$ are the neighbors of $u_2$, that is, $N_2 = \{u_3, u_5\}$.
*The fourth step:* Compute $p_{26}$ according to formula 1. Here, $p_{26} \approx -3$.

### 3.2. Native dimension-reducing algorithms

We know that the raw rating data are usually very sparse. The $p$KNN-based similarity between two users is zero when the two users have not rated any common object even if they have rated many objects with similar properties. This is obviously unreasonable in the real world. Therefore, dimension reduction techniques such as SVD [24,25], NMF [26], WNMF [27], etc. are brought into the recommendation system to solve the above problem by reflecting users' tastes on a small number of features extracted from a large number of interrelated objects. Before applying any factorization technique to a matrix, the matrix with any unknown entry must be filled to become a dense matrix. However, fillings, not true ratings from users, inevitably bring "noise". Besides, the rating from a user may be influenced by some factors such as his stomachache and hunger. Due to the "noise" brought by users and fillings, only the most essential features need to be retained. The number of retained features must be large enough to retain the correct information as much as possible and small enough to remove the "noise" as much as possible. For the recommendation purpose, dimension reduction techniques can be used alone or crossed with other methods. An object generally owns multi-features with different proportions. A user may care about multi-features with different levels of interest. The two points are the basis of explaining the ratings predicted by recommendation algorithms related to dimension reduction techniques.

#### 3.2.1. SVD-based prediction algorithm

Given a dense rectangular matrix $Z_{m\times p}$ with rank $r$, SVD can decompose $Z_{m\times p}$ into the product of three factors:

$$Z = U \sum V^{\mathrm{T}}, \quad (3)$$

where $U$ is a $m \times r$ matrix whose columns are the orthogonal eigenvectors of $ZZ^{\mathrm{T}}$, $V$ is a $p \times r$ matrix whose columns are the orthogonal eigenvectors of $Z^{\mathrm{T}}Z$, and $\sum$ is a $r \times r$ matrix whose entries are all zero except for $\sum_{ii}$. $\sum_{ii} = \sqrt{\lambda_i}$, where $\lambda_i$ is the $i$th largest eigenvalue of $ZZ^{\mathrm{T}}$.

Based on the idea that the effect of small eigenvalues (and their eigenvectors) on a matrix–vector product is small [25], the low-rank approximation $Z_d$ to $Z$, whose rank is at most $d$ and which minimizes $\|Z - Z_d\|_F$, can be obtained by

$$Z_d = U_d \sum_d V_d^T = U_d \sum_d^{1/2} \left( V_d \sum_d^{1/2} \right)^T, \quad (4)$$

where $U_d$ is a $m \times d$ matrix whose columns are the first $d$ columns of $U$, $V_d$ is a $p \times d$ matrix whose columns are the first $d$ columns of $V$, and $\sum_d$ is a $d \times d$ matrix which consists of the first $d$ rows and the first $d$ columns of $\sum$.

The $i$th row of $U_d \sum_d^{1/2}$ can denote the preference model of $u_i$ so that $\left(U_d \sum_d^{1/2}\right)_{il}$ can represent the preference of $u_i$ for feature $l$. The $j$th row of $V_d \sum_d^{1/2}$ can denote the proportion model of $o_j$ so that $\left(V_d \sum_d^{1/2}\right)_{jl}$ can represent the proportion of feature $l$ on $o_j$. Obviously, $(Z_d)_{ij} = \left(U_d \sum_d^{1/2}\right)^{(i)} \cdot \left(\left(V_d \sum_d^{1/2}\right)^{(j)}\right)^T$. Therefore, $(Z_d)_{ij}$ can be used for the predicted rating of $o_j$ for $u_i$.

The SVD-based prediction algorithm (or SVDP for short) is described as follows:

(1) Convert the raw rating matrix $A_{m\times p}$ into a dense matrix $B_{m\times p}$ by a suitable matrix filling method.
(2) Normalize $B_{m\times p}$ as the matrix $Z_{m\times p}$ by $Z$-scores, namely $Z_{ij} = \frac{B_{ij} - \overline{B^j}}{\sigma_j}$ where $\overline{B^j}$ and $\sigma_j$, respectively, represent the mean value and standard deviation of the ratings in $B^j$, that is, $\overline{B^j} = \frac{1}{m}\sum_{i=1}^m B_{ij}$ and $\sigma_j^2 = \frac{1}{m-1}\sum_{i=1}^m (B_{ij} - \overline{B^j})^2$.
(3) Apply SVD to $Z$.
(4) Acquire the appropriate low-rank approximation $Z_d$ to $Z$.
(5) Compute $p_{ij}$ according to $p_{ij} = \overline{B^j} + \sigma_j (Z_d)_{ij}$.

Different users have different rating habits (e.g. some users prefer high ratings, while some users prefer low ratings). To weaken the impact from these differences, $B_{m\times p}$ needs to be normalized as $Z_{m\times p}$ for SVDP.

To effectively mine the latent information in rating data, these appropriate decisions on the matrix filling method and the value of $d$are critical to the effectiveness of the algorithm.

**Example 2.** Given the raw ratings in Fig. 1 and an active user $u_2$, if the matrix filling method is designated as the middle value-based filling method, which fills all unknown entries in the matrix with the middle value in the rating range, and the value of $d$ is set to 2, SVDP will predict $p_{26}$ as the following steps.

*The first step:*

$$A_{5\times 8}=\begin{pmatrix} -2 & ? & 5 & 1 & ? & 3 & ? & ? \\ -1 & 3 & 2 & ? & 4 & ? & 5 & ? \\ ? & ? & -4 & 2 & 4 & -2 & ? & 5 \\ -2 & 2 & 4 & 2 & 5 & ? & ? & ? \\ -3 & ? & -5 & 1 & 3 & 1 & ? & 4 \end{pmatrix} \xrightarrow{\text{filling}}$$

$$B_{5\times 8}=\begin{pmatrix} -2 & 0 & 5 & 1 & 0 & 3 & 0 & 0 \\ -1 & 3 & 2 & 0 & 4 & 0 & 5 & 0 \\ 0 & 0 & -4 & 2 & 4 & -2 & 0 & 5 \\ -2 & 2 & 4 & 2 & 5 & 0 & 0 & 0 \\ -3 & 0 & -5 & 1 & 3 & 1 & 0 & 4 \end{pmatrix}$$

*The second step:*

$$\overline{B}^1=-1.6,\ \overline{B}^2=1.0,\ \overline{B}^3=0.4,\ \overline{B}^4=1.2,\ \overline{B}^5=3.2,$$
$$\overline{B}^6=0.4,\ \overline{B}^7=1.0,\ \overline{B}^8=1.8.$$
$$\sigma_1=1.14,\ \sigma_2=1.41,\ \sigma_3=4.62,\ \sigma_4=0.84,$$
$$\sigma_5=1.92,\ \sigma_6=1.82,\ \sigma_7=2.24,\ \sigma_8=2.49.$$

$$\underbrace{\begin{matrix}\overline{B}^1,\overline{B}^2,\overline{B}^3,\overline{B}^4,\overline{B}^5,\overline{B}^6,\overline{B}^7,\overline{B}^8\\ \sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5,\sigma_6,\sigma_7,\sigma_8\end{matrix}}_{B_{5\times 8}} \xrightarrow{Z\text{-scores}} Z_{5\times 8}$$

$$=\begin{pmatrix} -0.351 & -0.707 & 0.997 & -0.239 & -1.664 & 1.431 & -0.447 & -0.723 \\ 0.526 & 1.414 & 0.347 & -1.434 & 0.416 & -0.220 & 1.789 & -0.723 \\ 1.403 & -0.707 & -0.953 & 0.956 & 0.416 & -1.321 & -0.447 & 1.285 \\ -0.351 & 0.707 & 0.780 & 0.956 & 0.936 & -0.220 & -0.447 & -0.723 \\ -1.228 & -0.707 & -1.170 & -0.239 & -0.104 & 0.330 & -0.447 & 0.884 \end{pmatrix}$$

*The third step:*

$$Z_{5\times 8}\overset{\text{SVD}}{=}\underbrace{\begin{pmatrix} 0.371 & -0.657 & -0.075 & -0.475 & -0.447 \\ 0.531 & 0.609 & 0.378 & -0.067 & -0.447 \\ -0.713 & 0.250 & -0.016 & -0.479 & -0.447 \\ 0.072 & 0.138 & -0.780 & 0.409 & -0.447 \\ -0.260 & -0.340 & 0.492 & 0.612 & -0.447 \end{pmatrix}}_{U}$$
$$\times\underbrace{\begin{pmatrix} 3.616 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 3.285 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 2.211 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.802 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}}_{\sum}$$
$$\times\underbrace{\begin{pmatrix} -0.154 & 0.387 & -0.058 & -0.796 & -0.090 \\ 0.339 & 0.453 & -0.136 & 0.242 & -0.501 \\ 0.441 & -0.054 & -0.503 & -0.242 & 0.606 \\ -0.387 & -0.080 & -0.634 & -0.002 & -0.288 \\ -0.165 & 0.492 & -0.229 & 0.489 & 0.416 \\ 0.347 & -0.471 & 0.075 & 0.044 & -0.063 \\ 0.328 & 0.415 & 0.382 & -0.083 & 0.099 \\ -0.511 & -0.014 & 0.344 & 0.012 & 0.322 \end{pmatrix}^T}_{V}.$$

*The fourth step:*

$$Z_d\overset{d=2}{=}\underbrace{\begin{pmatrix} 0.371 & -0.657 \\ 0.531 & 0.609 \\ -0.713 & 0.250 \\ 0.072 & 0.138 \\ -0.260 & -0.340 \end{pmatrix}}_{U_d}\underbrace{\begin{pmatrix} 3.616 & 0.000 \\ 0.000 & 3.285 \end{pmatrix}}_{\sum_d}\underbrace{\begin{pmatrix} -0.154 & 0.387 \\ 0.339 & 0.453 \\ 0.441 & -0.054 \\ -0.387 & -0.080 \\ -0.165 & 0.492 \\ 0.347 & -0.471 \\ 0.328 & 0.415 \\ -0.511 & -0.014 \end{pmatrix}^T}_{V_d}$$
$$=\begin{pmatrix} -1.041 & -0.522 & 0.706 & -0.346 & -1.283 & 1.481 & -0.455 & -0.656 \\ 0.479 & 1.557 & 0.738 & -0.904 & 0.666 & -0.277 & 1.459 & -1.009 \\ 0.715 & -0.503 & -1.180 & 0.932 & 0.830 & -1.280 & -0.505 & 1.307 \\ 0.136 & 0.294 & 0.090 & -0.137 & 0.180 & -0.124 & 0.273 & -0.139 \\ -0.288 & -0.825 & -0.355 & 0.454 & -0.394 & 0.200 & -0.772 & 0.496 \end{pmatrix}.$$

*The fifth step:*

$$p_{26}=\overline{B^6}+\sigma_6^*(Z_d)_{26}\quad\approx 0.$$

*3.2.2. NMF-based prediction algorithm*

NMF learns factors by non-negativity constraints. This means two points. First, NMF only can be applied to a nonnegative matrix. Second, factors learned by NMF are nonnegative.

Given a dense nonnegative matrix $Z_{m\times p}$, NMF aims at finding out two nonnegative factors,$G_{m\times d}$ and $H_{d\times p}$, to complete the low-rank approximation $Z_d$ to $Z$, whose rank is at most $d$ and which satisfies $Z\approx Z_d=GH$.

The $i$th row of $G$ can denote the preference model of $u_i$ so that $G_{il}$ can denote the preference of $u_i$ for feature $l$. The $j$th column of $H$ can denote the proportion model of $o_j$ so that $H_{lj}$ can denote the proportion of feature $l$ on $o_j$. Obviously $(Z_d)_{ij}=G^{(i)}\cdot H^j$. Therefore, $(Z_d)_{ij}$ can be used for the predicted rating of $o_j$ for $u_i$.

The quality of $Z_d$ is often measured by the Euclidean distance between $Z_d$ and $Z$. Therefore NMF for $Z$ is a nonlinear optimization problem as follows:

$$\arg\min_{G,H}\|Z-Z_d\|_F=\arg\min_{G,H}\sqrt{\sum_{ij}\left(Z_{ij}-(GH)_{ij}\right)^2}\quad\text{subject to } Z,G,H\geqslant 0. \tag{5}$$

The most common NMF methods solving the optimization problem are multiplicative update (MU) and alternating least squares (ALS).

MU for solving the optimization problem is described as follows:

Initialize $G$ and $H$;
for *iter* = 1 to $f$ // $f$ denotes the iteration number.
$H=H\cdot{}^*(G^TZ)\cdot/G^TGH$; //update rule
$G=G\cdot{}^*(ZH^T)\cdot/GHH^T$; //update rule
end.
where $\cdot^*$ and $\cdot/$, respectively, denote element-wise multiplication and element-wise division.
ALS for solving the optimization problem is described as follows:
Initialize $G$;
for *iter* = 1 to $f$ // $f$ denotes the iteration number.
Solve for $H$ in matrix equation $G^TGH=G^TZ$; //update rule
Solve for $G$ in matrix equation $HH^TG^T=HZ^T$; //update rule
end.

The simple random initialization (SRI) and the nonnegative double singular value decomposition (NNDSVD) [28] are two most common initialization methods. SRI randomly initializes each element of $G$ (and $H$). NNDSVD is based on two SVD processes, one approximating the data matrix, the other approximating positive sections of the resulting partial SVD factors utilizing an algebraic property of unit rank matrices [28]. If the initial matrices, $G_0$ and

$H_0$, are strictly positive, then $G$ and $H$ remain positive throughout the iterations [29]. Therefore, to remain these matrices positive throughout the iterations, each element of $G$ (and $H$) needs to be initialized with a positive random value not more than 1. If each element of $Z$ is also in the interval $(0,1]$, it takes much less time to perform matrix factorization than if some elements of $Z$ are not in $(0,1]$ [30]. Hence, if some elements of $Z$ are not in $(0,1]$, $Z$ needs to be normalized by the column normalization (CN), that is, $Z_{ij}$ needs to be normalized as $Z_{ij}/\max Z^j$.

Obviously, before applying NMF to a matrix with some negative elements, the matrix must be preprocessed to become nonnegative. We resort to the matrix positive processing method of adding $a_{\max}+1$ to each element of the matrix.

The NMF-based prediction algorithm (or NMFP for short) is described as follows:

(1) Convert $A_{m\times p}$ into a dense positive matrix $Z_{m\times p}$ by the following two steps. First, $A_{m\times p}$ is filled into a dense matrix $B_{m\times p}$ by a suitable matrix filling method. Second, $B_{m\times p}$ is transformed into the positive matrix $Z_{m\times p}$ by the matrix positive processing method.
(2) Normalize $Z_{m\times p}$ into the matrix $\zeta_{m\times p}$ by CN, namely, $\zeta_{ij}=Z_{ij}/\max Z^j$.
(3) Apply the appropriate NMF method, with the appropriate initialization method and the appropriate iteration number, to $\zeta_{m\times p}$ and then obtaining $\zeta_d=G_{m\times d}H_{d\times p}$.
(4) Compute $p_{ij}$ by the following two steps. First, $q_{ij}=(\zeta_d)_{ij}{}^*\max Z^j$ if CN has been applied to $Z$ and $q_{ij}=(\zeta_d)_{ij}$ otherwise. Second, $p_{ij}=q_{ij}-a_{\max}-1$ if the matrix's positive processing has been applied to $B$ and $p_{ij}=q_{ij}$ otherwise.

Not only will different NMF methods produce different NMF factors, the same NMF method, when running with slightly different parameters, can produce different NMF factors [31]. It is evident that these appropriate decisions on the matrix filling method, NMF method, initialization method and values of $d$ and $f$ are critical to the effectiveness of the algorithm.

**Example 3.** Given the raw ratings in Fig. 1 and an active user $u_4$, if the matrix filling method, NMF method, initialization method and values of $d$ and $f$ are designated as the middle value-based method, MU, NNDSVD, 2 and 10, respectively, NMFP will predict the rating of $o_7$ for $u_4$, $p_{47}$, as the following steps.

*The first step:*

$$A_{5\times 8}\overset{\text{filling}}{\Rightarrow}B_{5\times 8}=\begin{pmatrix}-2&0&5&1&0&3&0&0\\-1&3&2&0&4&0&5&0\\0&0&-4&2&4&-2&0&5\\-2&2&4&2&5&0&0&0\\-3&0&-5&1&3&1&0&4\end{pmatrix}\overset{\text{positive}}{\Rightarrow}$$

$$Z_{5\times 8}=\begin{pmatrix}4&6&11&7&6&9&6&6\\5&9&8&6&10&6&11&6\\6&6&2&8&10&4&6&11\\4&8&10&8&11&6&6&6\\3&6&1&7&9&7&6&10\end{pmatrix}.$$

*The second step:*

$$Z_{5\times 8}\overset{\text{CN}}{\Rightarrow}\zeta_{5\times 8}=\begin{pmatrix}0.667&0.667&1.000&0.875&0.545&1.000&0.545&0.545\\0.833&1.000&0.727&0.750&0.909&0.667&1.000&0.545\\1.000&0.667&0.182&1.000&0.909&0.444&0.545&1.000\\0.667&0.889&0.909&1.000&1.000&0.667&0.545&0.545\\0.500&0.667&0.091&0.875&0.818&0.778&0.545&0.909\end{pmatrix}$$

*The third step:*

$$\zeta\overset{\substack{\text{NNDSVD}\\ \Rightarrow\\ d=2}}{\Rightarrow}\begin{cases}G_0=\begin{pmatrix}0.941&0.590\\1.044&0.236\\0.963&0.000\\1.024&0.331\\0.867&0.000\end{pmatrix}\\ H_0=\begin{pmatrix}0.761&0.808&0.613&0.925&0.866&0.728&0.662&0.721\\0.000&0.076&0.685&0.000&0.000&0.186&0.049&0.000\end{pmatrix}\end{cases}$$

$$\begin{matrix}\zeta\\ G_0\\ H_0\end{matrix}\overset{\substack{\text{MU}\\ \Rightarrow\\ f=10}}{\Rightarrow}\zeta_d=\underset{G}{\begin{pmatrix}0.793&0.917\\0.994&0.490\\1.030&0.000\\0.936&0.662\\0.919&0.000\end{pmatrix}}\times\underset{H}{\begin{pmatrix}0.787&0.789&0.231&0.955&0.899&0.594&0.660&0.760\\0.000&0.103&0.955&0.000&0.000&0.367&0.055&0.000\end{pmatrix}}$$

$$=\begin{pmatrix}0.624&0.721&1.059&0.757&0.712&0.808&0.573&0.602\\0.783&0.835&0.697&0.949&0.893&0.771&0.683&0.755\\0.811&0.813&0.238&0.983&0.925&0.612&0.679&0.782\\0.737&0.807&0.848&0.894&0.841&0.799&0.654&0.711\\0.723&0.725&0.212&0.877&0.825&0.546&0.606&0.698\end{pmatrix}$$

*The fouth step:*

$$p_{47}=(\zeta_d)_{47}{}^*\max Z^7-a_{\max}-1\approx 1.$$

### 3.2.3. WNMF-based prediction algorithm

Because the raw rating matrix is usually very sparse, it looks attractive predicting unknown ratings by only minimizing the errors between known ratings and estimated ratings for known ones. WNMF, a variant of NMF, caters to the need. Given a dense nonnegative matrix $Z_{m\times p}$, WNMF for the matrix is a weight optimization problem as follows:

$$\|W.^*(Z-GH)\|_F=\sqrt{\sum_{ij}(W_{ij}*(Z_{ij}-(GH)_{ij}))^2},\quad \text{subject to } Z,G,H\geqslant 0, \tag{6}$$

where $W$ denotes a weight matrix, and $W_{ij}$ is equal to one if the corresponding raw rating exists and zero otherwise. It is straightforward to modify update rules of NMF to handle missing entries in the matrix [27]. We focus on the problem whether WNMF is always superior to NMF under the similar update method. For simplicity,

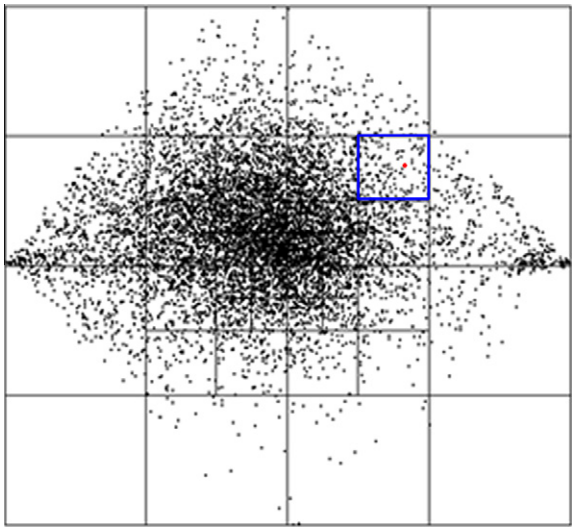

**Fig. 2.** A simple explanation of Eigentaste.

we only adapted update rules of MU. The multiplicative update rules for solving the above weight optimization problem are:

$$H = H \cdot {}^*((G^{\mathrm{T}}(W \cdot {}^*Z)) \cdot /(G^{\mathrm{T}}(W \cdot {}^*(GH)))),$$
$$G = G \cdot {}^*(((W \cdot {}^*Z)H^{\mathrm{T}}) \cdot /((W \cdot {}^*(GH))H^{\mathrm{T}})).$$

The WNMF-based prediction algorithm (or WNMFP for short) is the same as NMFP except for the adapted update rules. For clarity, we repeat no more those details.

**Example 4.** Given the raw ratings in Fig. 1 and an active user $u_1$, if the matrix filling method, NMF method, initialization method and values of $d$ and $f$ are designated as the middle value-based method, MU, SRI, 2 and 10, respectively, WNMFP will predict the rating of $o_5$ for $u_1$, $p_{15}$, as follows:

*The first two steps* are identical to the first two steps in Example 3.
*The third step:*

$$\zeta \overset{\text{SRI}}{\underset{d=2}{\Rightarrow}} \begin{cases} G_0 = \begin{pmatrix} 0.130 & 0.100 \\ 0.245 & 0.388 \\ 0.358 & 0.392 \\ 0.061 & 0.274 \\ 0.395 & 0.253 \end{pmatrix} \\ H_0 = \begin{pmatrix} 0.967 & 0.421 & 0.376 & 0.880 & 0.315 \\ 0.897 & 0.305 & 0.529 & 0.489 & 0.306 \end{pmatrix} \end{cases}$$

$$\begin{matrix} \zeta \\ G_0 \\ H_0 \end{matrix} \overset{\text{MU}}{\underset{f=10}{\Rightarrow}} \zeta_d = \begin{pmatrix} 0.708 & 0.264 \\ 0.640 & 0.622 \\ 0.769 & 0.192 \\ 0.539 & 0.791 \\ 0.660 & 0.242 \end{pmatrix}_G \times \begin{pmatrix} 1.005 & 0.850 & 0.467 & 1.154 & 0.764 & 0.990 & 0.635 & 0.965 \\ 0.171 & 0.520 & 0.687 & 0.313 & 0.768 & 0.112 & 0.514 & 0.165 \end{pmatrix}_H$$

$$= \begin{pmatrix} 0.789 & 0.728 & 0.706 & 0.808 & 0.687 & 0.856 & 0.627 & 0.683 \\ 0.795 & 0.841 & 0.649 & 0.960 & 0.900 & 0.771 & 0.690 & 0.752 \\ 0.727 & 0.779 & 0.587 & 0.892 & 0.842 & 0.696 & 0.637 & 0.694 \\ 0.720 & 0.856 & 0.532 & 0.997 & 0.995 & 0.618 & 0.675 & 0.736 \\ 0.646 & 0.710 & 0.512 & 0.816 & 0.782 & 0.604 & 0.575 & 0.627 \end{pmatrix}$$

*The fourth step:*

$$p_{15} = (\zeta_d)_{15}{}^* \max Z^5 - a_{\max} - 1 \approx 2.$$

### 3.3. Hybrid algorithms

#### 3.3.1. Hybrid algorithm based on PCA and KNN

PCA [32] is a standard tool which can reduce the dimensionality of a data set consisting of a large number of interrelated variables. This reduction is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables. Here, the hybrid algorithm based on PCA and KNN makes use of differences in the contribution rates of features mined by PCA to build the similarity model of users.

Given the raw rating matrix $A_{m\times p}$, the matrix is transformed into a normalized dense matrix $Z_{m\times p}$ by a suitable matrix filling method and $Z$-scores. The correlations between $p$ objects are computed as follows:

$$C = \frac{1}{m-1} Z^{\mathrm{T}} Z, \tag{7}$$

where $C_{il}$ denotes the degree of correlation between $u_i$ and $u_l$.

The descending eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_q$ and their corresponding eigenvectors $T_1, T_2, \ldots, T_q$ are got by applying eigen-analysis to $C$. Now the $j$th extracted feature $f_j$ can be obtained by

$$f_j = T_j' X, \tag{8}$$

where $X$ is a column vector consisting of $p$ object-variables. The variance of $f_j$ is $\lambda_j$. Usually the number of retained features depends on whether the contribution rate of accumulative total of variance of these features is no less than 80%, that is,

$$d = \min\left\{ r | c_r = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_r}{\lambda_1 + \lambda_2 + \cdots + \lambda_q} \geqslant 0.8 \right\}, \tag{9}$$

where $c_r$ denotes the contribution rate of accumulative total of variance of the first $r$ features.

Given $T_{p\times d} = (T_1, T_2, \ldots, T_d)$, the new representations for m users, $Y_{m\times d}$, is produced by projecting $Z_{m\times p}$ onto the new $d$-dimensional subspace spanned by the first $d$ features, namely,

$$Y_{m\times d} = Z_{m\times p} T_{p\times d}. \tag{10}$$

Obviously, $Y^{(i)}$ can be seen as the preference model of $u_i$.

The contribution rate of the $l$th feature is

$$\alpha_l = \frac{\lambda_l}{\sum_{l=1}^{q} \lambda_l}. \tag{11}$$

A larger contribution rate inevitably leads to a greater impact on the similarity of two users because the larger the contribution rate of a feature, the stronger its ability in explaining the original variables. Based on this thinking, the distance and similarity formulas are designed as Eqs. (12) and (13) respectively.

$$\xi_{il} = \left( \sum_{f=1}^{d} \alpha_f (Y_{if} - Y_{lf})^2 \right)^{\frac{1}{2}}, \tag{12}$$

where $\xi_{il}$ denotes the distance between $u_i$ and $u_l$.

$$s_{il} = 1 - \frac{\xi_{il}}{\xi_{\max} + 1}, \tag{13}$$

where $\xi_{\max} = \max\{\xi_{il}\}$. When $\xi_{il} = 0$, $s_{il} = 1$ and when $\xi_{il} = \xi_{\max}$, $s_{il}$ approaches zero. Therefore, $s_{il}$ lies in the real range $(0, 1]$, where greater values correspond to higher similarities.

After the similarities between users are acquired, each predicted rating is calculated according to Eq. (1).

We refer to the above recommendation algorithm closely related to the contribution rates as the hybrid algorithm based on PCA and $\alpha$KNN (or PCA-$\alpha$KNN for short).

Obviously, the algorithm is only sensitive to the matrix filling method and the value of $k$.

**Example 5.** Given the raw ratings in Fig. 1 and an active user $u_2$, if the matrix filling method and value of $k$ are designated as the middle value-based method and 2 respectively, NMFP will predict $p_{26}$ as follows:

*The first two steps* are identical to the first two steps in Example 2.
*The third step:*

$$Z_{5\times 8} \overset{\text{Eq.(7)}}{\Rightarrow} C_{5\times 5} = \begin{pmatrix} 1.000 & 0.155 & -0.086 & 0.157 & 0.296 & -0.700 & 0.294 & 0.211 \\ 0.155 & 1.000 & 0.460 & -0.423 & 0.551 & -0.195 & 0.791 & -0.639 \\ -0.086 & 0.460 & 1.000 & -0.155 & -0.265 & 0.513 & 0.194 & -0.949 \\ 0.157 & -0.423 & -0.155 & 1.000 & 0.280 & -0.395 & -0.802 & 0.384 \\ 0.296 & 0.551 & -0.265 & 0.280 & 1.000 & -0.816 & 0.232 & 0.167 \end{pmatrix}$$

*The fourth step:*

$$C_{5\times5} \overset{\text{eigen-analysis}}{\Rightarrow} \begin{cases} \boxed{\begin{array}{l}\text{Descending eigenvalues}: \lambda_1 = 3.269, \lambda_2 = 2.697, \lambda_3 = 1.223,\\ \lambda_4 = 0.811, \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = 0.0.\\ \text{Accumulative contribution rates}: \ c_1 = 0.409, c_2 = 0.746,\\ c_3 = 0.899, c_4 = c_5 = c_6 = c_7 = c_8 = 1.0.\\ \text{Therefore}, d = 3.\\ \text{Contribution rates of the first three features}: \ \alpha_1 = 0.409,\\ \alpha_2 = 0.337, \alpha_3 = 0.153.\end{array}} \\ T_{8\times3} = \begin{pmatrix} 0.154 & 0.387 & 0.058 \\ -0.339 & 0.453 & 0.136 \\ -0.441 & -0.054 & 0.503 \\ 0.387 & -0.080 & 0.634 \\ 0.165 & 0.492 & 0.229 \\ -0.347 & -0.471 & -0.075 \\ -0.328 & 0.415 & -0.382 \\ 0.511 & -0.014 & -0.344 \end{pmatrix} \end{cases}$$

*The fifth step:*

$$Z_{5\times8}, T_{8\times3} \overset{\text{Eq.(10)}}{\Rightarrow} Y_{5\times3} = \begin{pmatrix} -1.340 & -2.158 & 0.165 \\ -1.919 & 2.001 & -0.836 \\ 2.578 & 0.821 & 0.034 \\ -0.259 & 0.454 & 1.725 \\ 0.941 & -1.118 & -1.089 \end{pmatrix}$$

*The sixth step:*

$$\underset{Y_{5\times3}}{\alpha_1, \alpha_2, \alpha_3} \overset{\text{Eq.(12)}}{\Rightarrow} \xi_{5\times5} = \begin{pmatrix} 0.000 & 2.474 & 3.044 & 1.775 & 1.653 \\ 2.474 & 0.000 & 2.974 & 1.713 & 2.575 \\ 3.044 & 2.974 & 0.000 & 1.942 & 1.598 \\ 1.775 & 1.713 & 1.942 & 0.000 & 1.622 \\ 1.653 & 2.575 & 1.598 & 1.622 & 0.000 \end{pmatrix}$$

*The seventh step:*

$$\xi_{5\times5} \overset{Eq.(13)}{\Rightarrow} S_{5\times5} = \begin{pmatrix} 1.000 & 0.388 & 0.247 & 0.561 & 0.591 \\ 0.388 & 1.000 & 0.265 & 0.576 & 0.363 \\ 0.247 & 0.265 & 1.000 & 0.520 & 0.605 \\ 0.561 & 0.576 & 0.520 & 1.000 & 0.599 \\ 0.591 & 0.363 & 0.605 & 0.599 & 1.000 \end{pmatrix}$$

*The eighth step:*

$$\boxed{\underset{S_{5\times5}}{A_{5\times8} \Rightarrow \text{Users having rated } O_6, \text{ namely, } u_1, u_3 \text{ and } u_5.}} \Rightarrow N_2 = \{u_1, u_5\}$$

$$A_{5\times8} \Rightarrow \bar{u}_1 = 1.75, \quad \bar{u}_2 = 2.6, \quad \bar{u}_3 = 0.17$$

$$\boxed{\begin{array}{l}\bar{u}_1, \bar{u}_2, \bar{u}_4\\ N_2\\ A_{5\times8}\\ S_{5\times5}\end{array}} \overset{Eq.(1)}{\Rightarrow} p_{26} \approx 4$$

### 3.3.2. Hybrid algorithm based on SVD and KNN

The hybrid algorithm based on SVD and KNN is described as follows:

(1) Run the first three steps of SVDP to acquire $U_d\sum_d^{1/2}$.
(2) Take advantage of the preference model of each user from $U_d\sum_d^{1/2}$ to compute similarities between users as follows:

$$sim(i,l) = \frac{\left(U_d\Sigma_d^{1/2}\right)^{(i)} \cdot \left(U_d\Sigma_d^{1/2}\right)^{(l)}}{\left\|\left(U_d\Sigma_d^{1/2}\right)^{(i)}\right\|_2 * \left\|\left(U_d\Sigma_d^{1/2}\right)^{(l)}\right\|_2}.$$

(3) Compute $p_{ij}$ according to Eq. (1).

Because of the cosine similarity between two users' preference models, the algorithm is called the hybrid algorithm based on SVD and *c*KNN (or SVD-*c*KNN for short).

Obviously, these appropriate decisions on the matrix filling method and values of $d$ and $k$ are critical to the effectiveness of SVD-*c*KNN.

**Example 6.** Given the raw ratings in Fig. 1 and an active user $u_2$, if the matrix filling method is designated as the middle value-based method and the values of both $d$ and $k$ are set to 2, SVD-*c*KNN will predict $p_{26}$ as follows:

*The first step:*
After the first three steps in Example 2, $U$ and $\sum$, respectively, are

$$U = \begin{pmatrix} 0.371 & -0.657 & -0.075 & -0.475 & -0.447 \\ 0.531 & 0.609 & 0.378 & -0.067 & -0.447 \\ -0.713 & 0.250 & -0.016 & -0.479 & -0.447 \\ 0.072 & 0.138 & -0.780 & 0.409 & -0.447 \\ -0.260 & -0.340 & 0.492 & 0.612 & -0.447 \end{pmatrix} \text{ and}$$

$$\sum = \begin{pmatrix} 3.616 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 3.285 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 2.211 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.802 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}.$$

Then

$$U_d\sum_d^{1/2} \overset{d=2}{=} \underset{U_d}{\begin{pmatrix} 0.371 & -0.657 \\ 0.531 & 0.609 \\ -0.713 & 0.250 \\ 0.072 & 0.138 \\ -0.260 & -0.340 \end{pmatrix}} \underset{\sum_d^{1/2}}{\begin{pmatrix} 1.902 & 0.000 \\ 0.000 & 1.812 \end{pmatrix}}$$

$$= \begin{pmatrix} 0.705 & -1.191 \\ 1.009 & 1.104 \\ -1.355 & 0.453 \\ 0.136 & 0.251 \\ -0.495 & -0.617 \end{pmatrix}.$$

*The second step:*

$$U_d\sum_d^{1/2} \Rightarrow S_{5\times5} = \begin{pmatrix} 1.000 & -0.291 & -0.756 & -0.513 & 0.352 \\ -0.291 & 1.000 & -0.406 & 0.971 & -0.998 \\ -0.756 & -0.406 & 1.000 & -0.175 & 0.346 \\ -0.513 & 0.971 & -0.175 & 1.000 & -0.984 \\ 0.352 & -0.998 & 0.346 & -0.984 & 1.000 \end{pmatrix}$$

*The third step:*

$$\boxed{\underset{S_{5\times5}}{A_{5\times8} \Rightarrow \text{Users having rated } O_6, \text{ namely, } u_1, u_3 \text{ and } u_5.}} \Rightarrow N_2 = \{u_1, u_3\}$$

$$A_{5\times 8} \Rightarrow \bar{u}_1 = 1.75, \bar{u}_2 = 2.6, \bar{u}_5 = 1.0$$

$$\boxed{\begin{array}{l}\bar{u}_1, \bar{u}_2, \bar{u}_4 \\ N_2 \\ A_{5\times 8} \\ S_{5\times 5}\end{array}} \overset{Eq.(1)}{\Rightarrow} p_{26} \approx 1$$

*3.3.3. Hybrid algorithm based on NMF and KNN*

The hybrid algorithm based on NMF and KNN is described as follows:

(1) Run the first three steps of NMFP to acquire $G_{m\times d}$.
(2) Take advantage of the preference model of each user from $G_{m\times d}$ to compute similarities between users as follows:

$$sim(i,l) = \frac{(G_{m\times d})^{(i)} \cdot (G_{m\times d})^{(l)}}{\|(G_{m\times d})^{(i)}\|_2 * \|(G_{m\times d})^{(l)}\|_2}.$$

(3) Compute $p_{ij}$ according to Eq. (1).

Because of the cosine similarity between two users' preference models, the algorithm is called the hybrid algorithm based on NMF and *c*KNN (or NMF-*c*KNN for short).

Obviously, NMF-*c*KNN is sensitive to the matrix filling method, NMF method, initialization method and values of $d$, $f$ and $k$.

**Example 7.** Given the raw ratings in Fig. 1 and an active user $u_2$, if the matrix filling method, NMF method, initialization method and values of $d$, $f$ and $k$ are designated as the middle value-based method, MU, NNDSVD, 2, 10 and 2, respectively, SVD-*c*KNN will predict $p_{26}$ as follows:

*The first step:*
After the first three steps in Example 3, $G_{m\times d}$ is

$$G_{5\times 2} = \begin{pmatrix} 0.793 & 0.917 \\ 0.994 & 0.490 \\ 1.030 & 0.000 \\ 0.936 & 0.662 \\ 0.919 & 0.000 \end{pmatrix}.$$

**Table 1**
The evaluated methods (or values) of matrix filling method, NMF method, initialization method,$k$, $d$ and $f$.

| | |
|---|---|
| Matrix filling method | User average-based, object average-based, middle value-based |
| NMF method | MU, ALS |
| Initialization method | SRI, NNDSVD |
| ***k*** | *MovieLens*: 1, 2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600,700, 800, 900, 942<br>*Jester*: 1, 2, 10,20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600,700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500 |
| ***d*** | *MovieLens*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 943<br>*Jester*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 128 |
| ***f*** | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 30, 40, 50, 60, 70, 80, 90,100 |

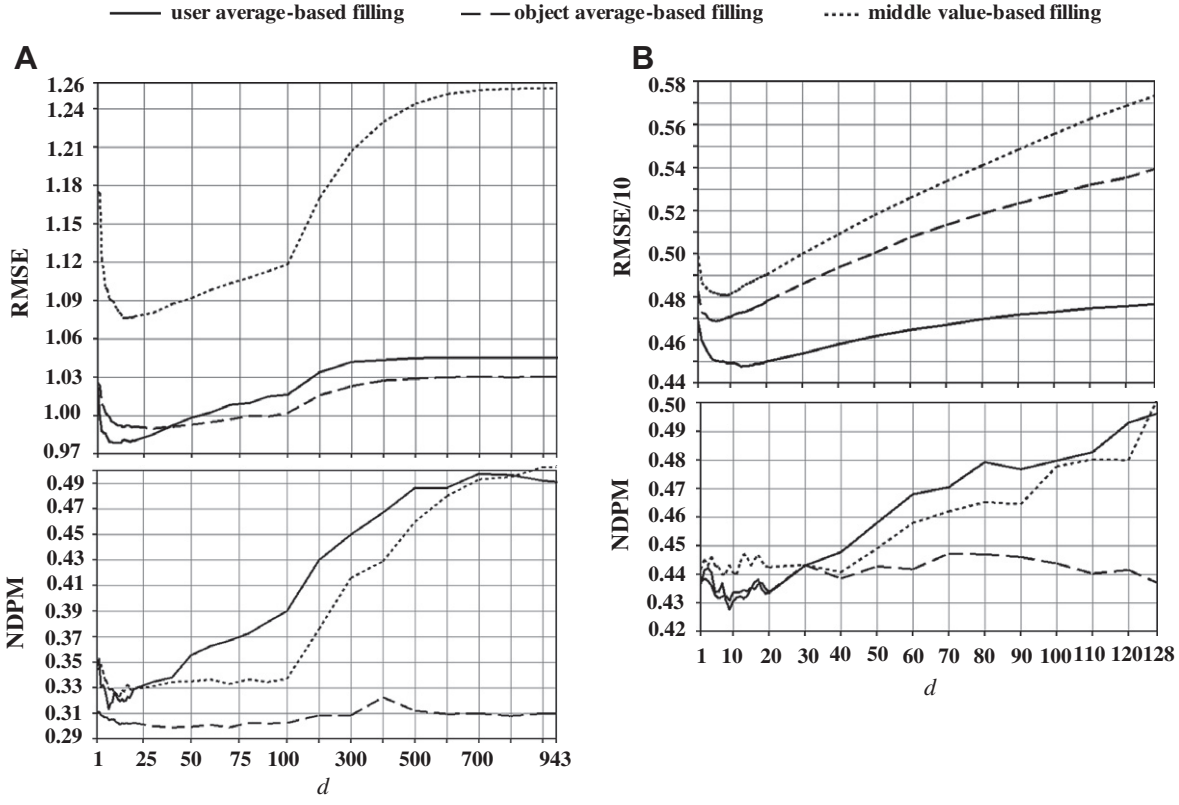


**Fig. 3.** The RMSE and NDPM vs. $d$ for SVDP on *MovieLens* (A) and *Jester* (B).

**Table 2**
Experimental results for NMFP.

| | Matrix filling method | NMF method | Initialization method | RMSE | NDPM |
|---|---|---|---|---|---|
| *MovieLens* | user average-based | MU | SRI | (1,2,1.0247) | (1,1,0.3434) |
| | | ALS | SRI | (9,14,0.978) | (10,15,0.3152) |
| | | MU | NNDSVD | (7,100,0.9943) | (8,10,0.3215) |
| | | ALS | NNDSVD | (9,6,0.978) | (11,4,0.3136) |
| | Object average-based | MU | SRI | (1,2,1.0069) | (2,80,0.3072) |
| | | ALS | SRI | (13,30,0.9744) | (15,70,0.291) |
| | | MU | NNDSVD | (11,100,0.9896) | (16,80,0.2961) |
| | | ALS | NNDSVD | ***(14,5,0.9739)*** | ***(16,3,0.2906)*** |
| | Middle value-based | MU | SRI | (600,100,1.1293) | (1,1,0.3444) |
| | | ALS | SRI | (14,30,1.0503) | (14,19,0.3163) |
| | | MU | NNDSVD | (17,100,1.0745) | (15,2,0.3144) |
| | | ALS | NNDSVD | (12,2,1.0501) | (15,2,0.3174) |
| | Matrix filling method | NMF method | Initialization method | RMSE/10 | NDPM |
| *Jester* | User average-based | MU | SRI | (100,100,0.4646) | (1,1,0.4346) |
| | | ALS | SRI | ***(20,6,0.4508)*** | (10,13,0.433) |
| | | MU | NNDSVD | (30,100,0.4586) | ***(10,3,0.425)*** |
| | | ALS | NNDSVD | (20,19,0.4511) | (10,7,0.4332) |
| | Object average-based | MU | SRI | (2,100,0.4791) | (2,90,0.4347) |
| | | ALS | SRI | (10,40,0.4705) | (40,80,0.4282) |
| | | MU | NNDSVD | (20,100,0.4755) | (30,5,0.4313) |
| | | ALS | NNDSVD | (10,19,0.4705) | (30,30,0.4288) |
| | Middle value-based | MU | SRI | (7,100,0.4909) | (1,2,0.4335) |
| | | ALS | SRI | (10,50,0.4823) | (2,18,0.4327) |
| | | MU | NNDSVD | (20,100,0.4888) | (10,1,0.4315) |
| | | ALS | NNDSVD | (10,2,0.4823) | (2,10,0.4327) |

**Table 3**
Experimental results for MU-based WNMFP.

| | Matrix filling method | Initialization method | RMSE | NDPM |
|---|---|---|---|---|
| *MovieLens* | User average-based | SRI | (600,40,0.9411) | (400,10,0.3057) |
| | | NNDSVD | (1,6,0.9428) | (1,5,0.3109) |
| | Object average-based | SRI | (1,4,0.9429) | ***(600,5,0.3053)*** |
| | | NNDSVD | (1,2,0.9429) | (1,2,0.3086) |
| | Middle value-based | SRI | ***(2,20,0.937)*** | (500,9,0.3056) |
| | | NNDSVD | (1,6,0.9428) | (1,5,0.3109) |
| | Matrix filling method | Initialization method | RMSE/10 | NDPM |
| *Jester* | User average-based | SRI | (1,5,0.4516) | (1,3,0.4343) |
| | | NNDSVD | (20,1,0.4464) | (6,60,0.4276) |
| | Object average-based | SRI | (2,100,0.4477) | (1,3,0.4343) |
| | | NNDSVD | (20,2,0.445) | ***(20,9,0.4262)*** |
| | Middle value-based | SRI | (2,100,0.451) | (1,3,0.4343) |
| | | NNDSVD | ***(9,2,0.4442)*** | (2,1,0.4318) |

*The second step:*

$$G_{5\times 2} \Rightarrow S_{5\times 5} = \begin{pmatrix} 1.000 & 0.921 & 0.654 & 0.971 & 0.654 \\ 0.921 & 1.000 & 0.897 & 0.988 & 0.897 \\ 0.654 & 0.897 & 1.000 & 0.816 & 1.000 \\ 0.971 & 0.988 & 0.816 & 1.000 & 0.816 \\ 0.654 & 0.897 & 1.000 & 0.816 & 1.000 \end{pmatrix}$$

*The third step:*

$$\boxed{\begin{array}{l} A_{5\times 8} \Rightarrow \text{Users having rated } O_6, \text{ namely, } u_1, u_3 \text{ and } u_5. \\ \qquad\qquad\qquad\qquad\qquad\qquad S_{5\times 5} \end{array}} \Rightarrow N_2 = \{u_1, u_3\}$$

$$A_{5\times 8} \Rightarrow \bar{u}_1 = 1.75, \bar{u}_2 = 2.6, \bar{u}_3 = 1.0$$

$$\boxed{\begin{array}{l} \bar{u}_1, \bar{u}_2, \bar{u}_4 \\ \qquad N_2 \\ \qquad A_{5\times 8} \\ \qquad S_{5\times 5} \end{array}} \overset{Eq.(1)}{\Rightarrow} p_{26} \approx 2$$

*3.3.4. Hybrid algorithm based on PCA and a recursive rectangular clustering*

The hybrid algorithm of crossing PCA and a recursive rectangular clustering, Eigentaste [22], has six main steps:

(1) Normalize the dense matrix from the gauge set as the dense normalized matrix by *Z*-scores.
(2) Mine the first two unrelated features (PCs) by applying PCA to the dense normalized matrix.
(3) Project users onto 2-dimensional subspace spanned by the first two PCs.
(4) Cluster users by the recursive rectangular clustering.

**Table 4**
Summary of retained features about different filling methods for PCA-αKNN.

| | Matrix filling method | $p$ | $d$ | $c_d$ |
|---|---|---|---|---|
| *MovieLens* | User average-based | 943 | 1 | 0.8154 |
| | Object average-based | 943 | 246 | 0.8010 |
| | Middle value-based | 943 | 232 | 0.8008 |
| *Jester* | User average-based | 128 | 13 | 0.8001 |
| | Object average-based | 128 | 80 | 0.8047 |
| | Middle value-based | 128 | 76 | 0.8045 |

(5) Compute the mean rating for each non-gauge object of each cluster, based on the number of users who have rated that object in the cluster.
(6) Yield a lookup table of recommendations for each cluster by sorting the non-gauge objects in order of decreasing mean ratings.

**Example 8.** Given the *Jester* data introduced in Section 5, after the first four steps of Eigentaste are executed, the result on *Jester* is shown in Fig. 2, where each point is a user and each grid represents a cluster. The predicted rating of an unknown object for the active user marked by the red point is the mean of ratings of the object by all the users in the blue grid.

## 4. Evaluation criteria

Some users tend to be concerned about whether the predicted ratings are what he wishes and some users tend to focus on whether a ranked list of recommended objects agrees with his own wish. Correspondingly, we make use of three metrics, namely RMSE, Recall and NDPM, to evaluate a recommendation algorithm.

Accuracy metrics are used to evaluate the accuracy of a recommender algorithm by comparing the numerical prediction values against user raw ratings. The root mean square error (RMSE) [33] is the most common accuracy metric.

RMSE is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\Omega|}\sum_{(u_i,o_j)\in\Omega} |a_{ij} - p_{ij}|^2}, \tag{14}$$

where $\Omega = \{(u_i,o_j)|u_i \text{ had rated } o_j \text{ in the probe set}\}$.

A lower value of RMSE corresponds to a higher accuracy of the recommendation system.

The Recall [35] is defined as the ratio of the number of predictable items in the probe set to the total number of items in the probe set, that is,

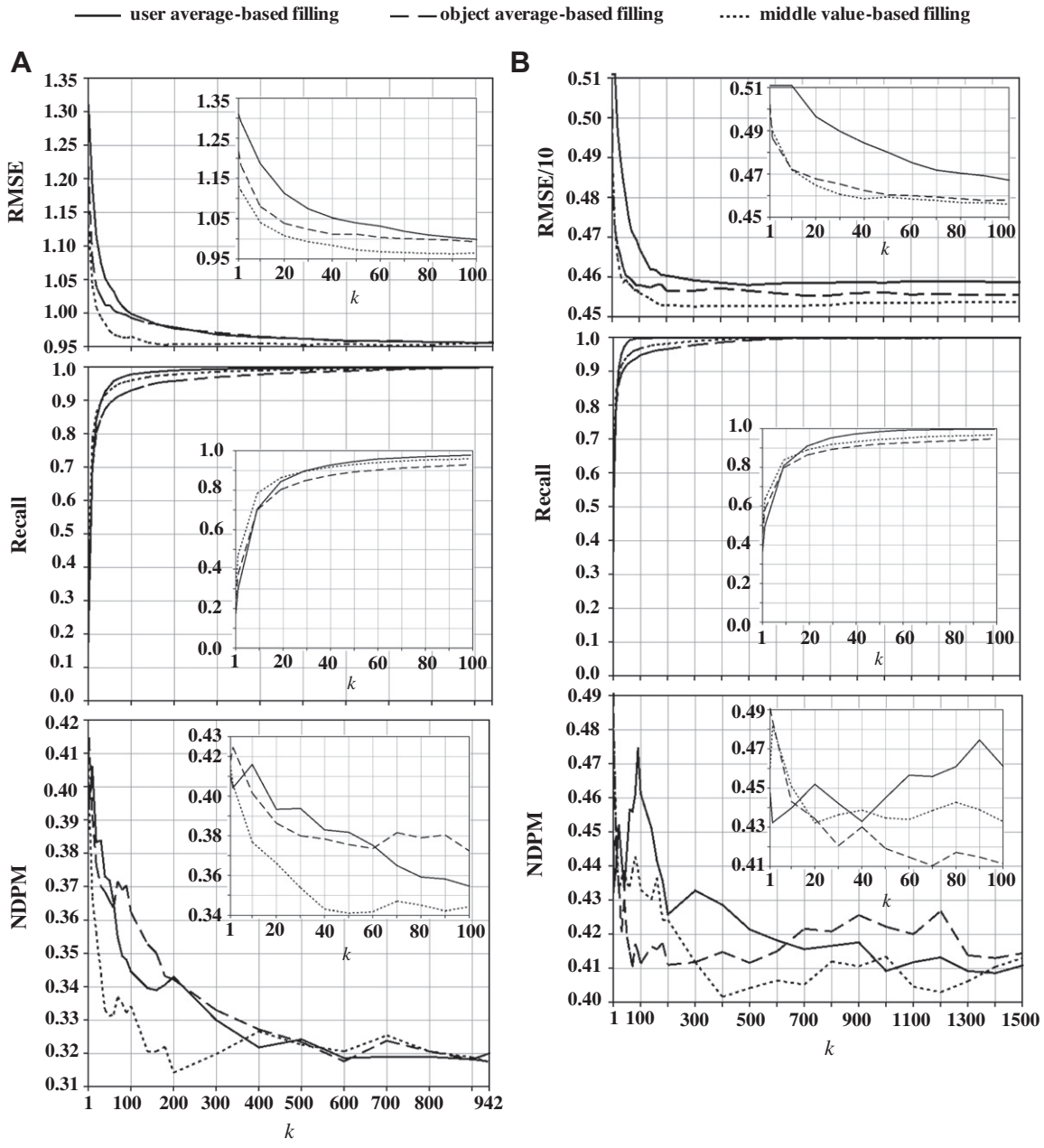


**Fig. 4.** The RMSE, Recall and NDPM vs.$k$ for PCA-$\alpha$KNN on *MovieLens* (A) and *Jester* (B). The insets show the enlarged drawings when $k = [1 \cdots 100]$.

$$\text{Recall} = p/t, \tag{15}$$

where $p$and $t$, respectively, denote the number of predictable items in the probe set and the total number of items in the probe set.

A larger value of Recall corresponds to a better performance.

The normalized distance-based performance measure (NDPM) [34] is a popular ranking metric. It only uses the information about the relative order of objects to examine the agreement or disagreement between the system and user rankings. There exists a preference relation between two objects in a ranking if the ranking puts one higher than the other and there exists an indifference relation otherwise. The system ranking $r^s$ and the user ranking $r^u$ contradict on $\{o_h, o_j\}$ if one ranking puts $o_h$ higher and the other puts $o_j$ higher. They are $r^u$-compatible on $\{o_h, o_j\}$ if $r^u$ puts $o_h$ or $o_j$ higher and $r^s$ has the two objects tied. NDPM for a user is defined as:

$$\text{NDPM}(r^s, r^u) = \frac{2C^- + C^u}{2C}, \tag{16}$$

where $C^-$ and $C^u$ represent, respectively, the numbers of contradictory pairs and $r^u$-compatible pairs in $r^s$ and $r^u$, and $C$ represents the number of object pairs with preference relations in $r^u$. The NDPM of the whole system with $m$ users is

$$\text{NDPM} = \frac{1}{m}\sum_{i=1}^{m}\frac{2C_i^- + C_i^u}{2C_i}. \tag{17}$$

A lower value of NDPM corresponds to a better ranking of the recommendation system.

## 5. Experimental data

We use two different types of data sets, *MovieLens* [36] and *Jester* [37], to evaluate algorithms in this paper.

The *movieLens* data contains 100,000 ratings (integer values ranging from 1 to 5) for 1682 movies by 943 users. Each user has rated at least 20 movies. The user-movie ratings need to be randomly divided into two parts: the training set contains 90% of ratings from each user, and the remaining 10% constitutes the probe.

The *Jester* data contains over 1.7 million continuous ratings (real values ranging from −10.00 to +10.00) of 150 jokes from 63,974 users. There exists a gauge set only containing 8 jokes, each of which has been rated by almost all users and is called a gauge joke. The remaining 142 are called non-gauge jokes, among which, 22 jokes are never displayed or rated. We randomly selected 9000 users for our experiments, each of who has rated all gauge jokes and one or more non-gauge jokes. The user-joke ratings need to be randomly divided into two parts: the training set contains all the ratings for gauge jokes and 90% of ratings for non-gauge jokes from each user, and the remaining 10% constitutes the probe.

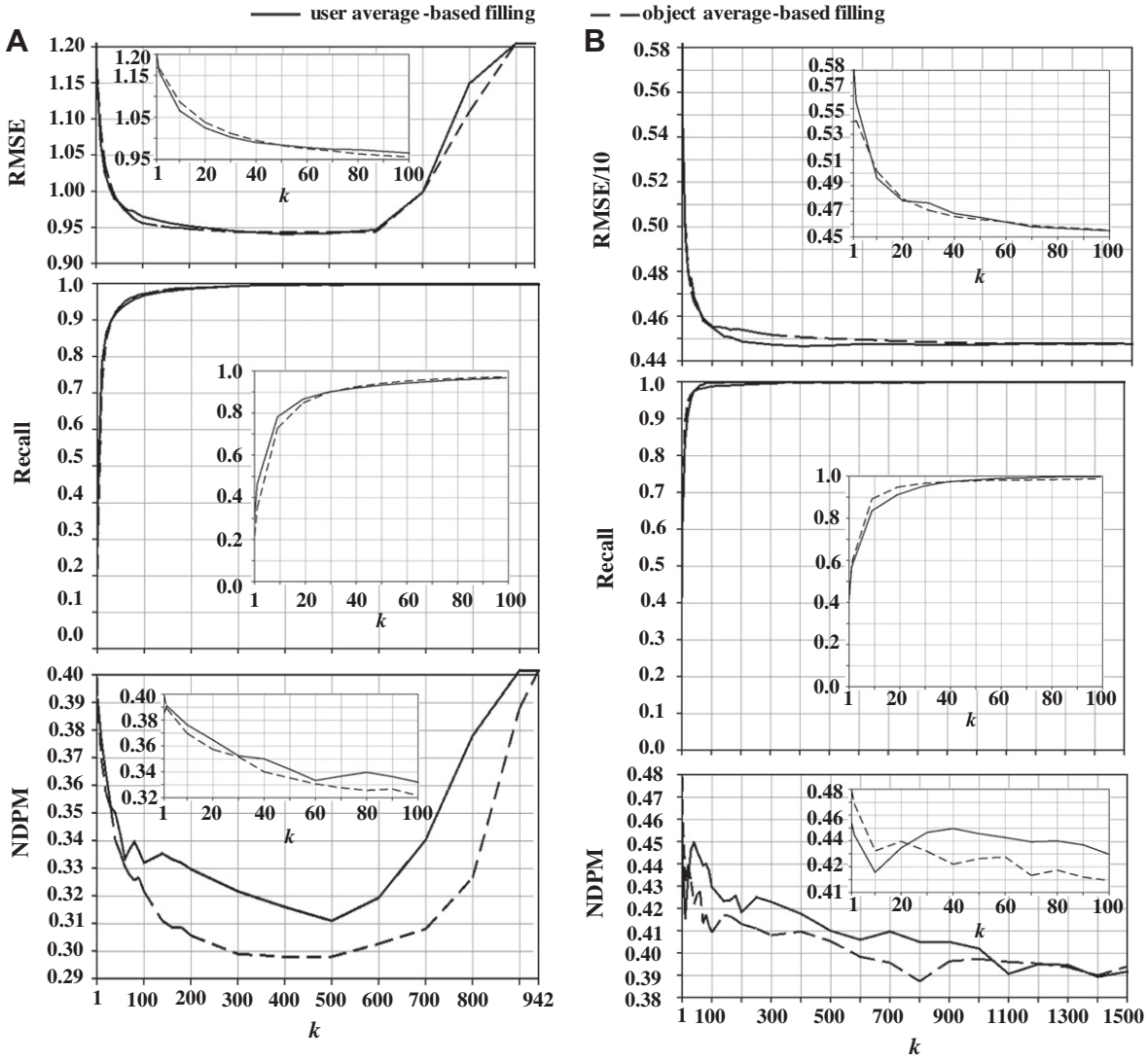


**Fig. 5.** The RMSE, Recall and NDPM vs. $k$ for SVD-cKNN on *MovieLens* (A) and *Jester* (B). The insets show the enlarged drawings when $k = [1 \cdots 100]$.

**Table 5**
the best values of $d$ for each metric when using a specified matrix filling method.

| | Matrix filling method | RMSE | NDPM |
|---|---|---|---|
| *MovieLens* | User average-based | $d = 11$ | $d = 12$ |
| | Object average-based | $d = 700$ | $d = 70$ |
| *Jester* | User average-based | $d = 20$ | $d = 10$ |
| | Object average-based | $d = 6$ | $d = 6$ |

The sparsity of the rating data refers to the ratio of the number of known ratings to the total number of known and unknown ratings. The sparsities of *MovieLens* and*Jester*, respectively, are 0.05765 and 0.2621. Evidently *MovieLens* is much sparser than *Jester*.

## 6. Experiment planning

To find out the appropriate configurations of the matrix filling method, NMF method, initialization method and values of $k$, $d$ and $f$ for algorithms, Table 1 lists these representative methods (or values) evaluated in our experiments. Because of the total number of users in the *MovieLens* data, the largest value of $k$ on *MovieLens* is 942.

The user average-based filling method uses the average rating of a user to fill all unknown entries belonging to the user. The object average-based filling method uses the average rating of an object to fill all unknown entries belonging to the object. The middle values of *MovieLens*and *Jester* are 3 and 0, respectively.

In Section 7 we provide the experimental results for each algorithm. Based on these results, the appropriate configurations for each algorithm can be found. Section 8 exhibits the comparisons among eight algorithms with appropriate configurations and presents some interesting findings.

## 7. Results for each algorithm

### 7.1. pKNN

We know in Section 3.1 that the effectiveness of *p*KNN is only sensitive to the value of $k$. To save space, the numerical results for *p*KNN are directly exhibited in Figs. 6 and 7.

### 7.2. SVDP

Fig. 3 shows the experimental results for SVDP. We can see that the middle value-based filling method is always inferior to the other two ones in uncovering users' tastes. The optimal values of RMSE on *MovieLens* and *Jester* are both acquired by the user average-based filling method. The optimal values of NDPM on *MovieLens* and *Jester* are both acquired by the object average-based filling method. Furthermore, each optimal value occurs at a certain small value of $d$. The Recall of SVDP is always 1.0.

### 7.3. NMFP

If the experimental results for NMFP are displayed in a figure, it will be chaotic. Therefore, we put these results in Table 2 where each cell $(d,f,v)$ denotes that, under the specified condition, the best value $v$ is obtained at rank $d$ and the number of iterations $f$. For example, under the condition of the user average-based filling method, ALS and SRI, the best value of NDPM on *MovieLens*, 0.3152, is obtained at $d = 10$ and $f = 15$. In this table, the optimum cell of each quality metric is highlighted in bold italic type.

The optimum values of RMSE and NDPM on *MovieLens*, respectively, are 0.9739 and 0.2906. These values are acquired respectively by the appropriate configuration of the object average-based matrix filling method, ALS, NNDSVD, $d = 14$ and $f = 5$, and the appropriate configuration of the object average-based matrix filling method, ALS, NNDSVD, $d = 16$ and $f = 3$. The optimum values of RMSE and NDPM on *Jester*, respectively, are 4.508 and 0.425. These values are acquired respectively by the appropriate configuration of the user average-based filling method, ALS, SRI, $d = 20$ and $f = 6$, and the appropriate configuration of the user average-based filling method, MU, NNDSVD, $d = 10$ and $f = 3$.

Obviously, the Recall of NMFP is always 1.0.

### 7.4. WNMFP

As we mentioned earlier, we focus on the problem whether WNMFP is always superior to NMFP. For simplicity, we only evaluated MU for WNMFP. To avoid chaos, experimental results for MU-based WNMFP are exhibited in Table 3 where each cell denotes the same meaning as that of NMFP and the optimum cell of each metric is highlighted in bold italic type.

The optimum values of RMSE and NDPM on *MovieLens*, respectively, are 0.937 and 0.3053. These values are acquired respectively by the appropriate configuration of the middle value-based filling method, SRI, $d = 2$ and $f = 20$, and the appropriate configuration of the object average-based filling method, SRI, $d = 600$ and $f = 5$. The optimum values of RMSE and NDPM on *Jester*, respectively, are 4.442 and 0.4262. These values are acquired respectively by the appropriate configuration of the middle value-based filling method, NNDSVD, $d = 9$ and $f = 2$, and the appropriate configuration of the object average-based filling method, NNDSVD, $d = 20$ and $f = 9$.

We can see from the comparison between Tables 2 and 3 that WNMFP is not always superior to NMFP.

Obviously, the recall of WNMFP is always 1.0.

### 7.5. PCA-αKNN

Table 4 shows the summary of retained features about different matrix filling methods for PCA-αKNN.

Fig. 4 exhibits the experimental results for PCA-αKNN. Whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 942]$, the optimal values of RMSE, Recall and NDPM on *MovieLens* are acquired respectively by the

**Table 6**
Experimental results for NMF-*c*KNN based on MU and NNDSVD.

| | Matrix filling method | RMSE (L) | RMSE (S) | NDPM (L) | NDPM (S) |
|---|---|---|---|---|---|
| *MovieLens* | User average-based | (7,400,0.9416) | (7,100,0.9604) | (9,700,0.3028) | (9,100,0.3231) |
| | Object average-based | (6,700,0.9457) | (6,100,0.9875) | (5,800,0.3027) | (5,100,0.3388) |
| | Middle value-based | ***(8,500,0.9387)*** | ***(8,100,0.9575)*** | ***(5,300,0.3022)*** | ***(5,100,0.3221)*** |
| *Jester* | Matrix filling method | RMSE (L)/10 | RMSE (S)/10 | NDPM (L) | NDPM (S) |
| | User average-based | ***(20,500,0.4481)*** | ***(20,100,0.4548)*** | (30,500,0.3916) | ***(30,100, 0.4142)*** |
| | Object average-based | (10,700,0.4493) | (10,100,0.4611) | (5,700,0.3939) | (5,100,0.4248) |
| | Middle value-based | (5,1000,0.4489) | (5,100,0.4604) | ***(5,500,0.3897)*** | (5,90,0.4314) |

middle value-based, user average-based, middle value-based filling methods. Whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 1500]$, the optimal values of RMSE and Recall on *Jester* are acquired respectively by the middle value-based and user average-based filling methods. When $k = [1 \cdots 100]$, the optimal value of NDPM on *Jester* is acquired by the object average-based filling method. When $k = [1 \cdots 1500]$, it is acquired by the middle value-based filling method.

### 7.6. SVD-cKNN

We learn in Section 7.2 that the middle value-based filling method for SVDP is always inferior to the other two ones in uncovering users' tastes. Therefore, only the user average-based and object average-based filling methods for SVD-*c*KNN need to be evaluated.

To avoid chaos, Fig. 5 only exhibited these results at the best values of $d$ listed in Table 5. For example, the best value of $d$ for NDPM by the user average-based matrix filling method on *Jester* is 10.

We can see from Fig. 5 that the optimal values of RMSE, Recall and NDPM on *MovieLens* can all be acquired by the object average-based filling method whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 942]$. The optimal values of RMSE and Recall on *Jester* can both be got by the user average-based filling method whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 1500]$. The optimal value of NDPM on *Jester* is acquired by the object average-based filling method whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 1500]$.

### 7.7. NMF-cKNN

We obviously felt in the experiments that the calculating speeds of ALS and NNDSVD are superior to those of MU and SRI respectively. Our focus is on whether NMF-*c*KNN is certainly better than NMFP under the same other conditions. Considering the speed, we evaluated NMF-*c*KNN based on NNDSVD and ALS. We found that the numbers of iterations at all the optimum values for NMFP are less than 20, so, for simplicity, we always set the number of iterations for NMF-*c*KNN to 20. Now NMF-*c*KNN only concerns two parameters, $d$ and $k$.

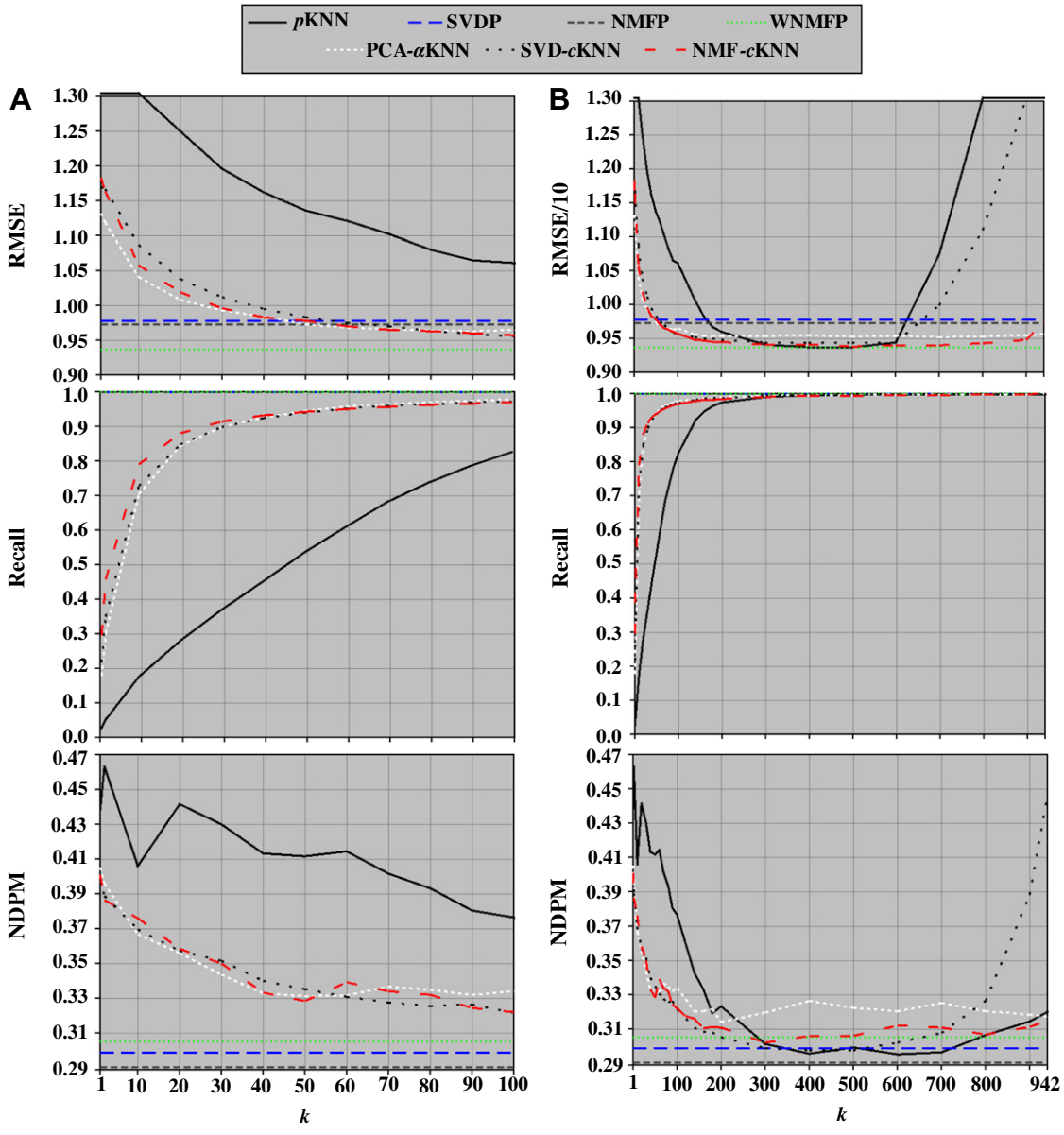


**Fig. 6.** The RMSE, Recall and NDPM vs. $k$ for eight different algorithms with appropriate configurations on *MovieLens* when $k = [1 \cdots 100]$ (A) and $k = [1 \cdots 942]$ (B).

To avoid chaos, experimental results for NMF-*c*KNN are exhibited in Table 6 where each cell $(d,k,v)$ denotes that, under the specified condition, the best value $v$ is obtained at $d$ and $k$. The optimum cell of each metric in this table is highlighted in bold italic type. "L" means $k = [1 \cdots 942]$ on *MovieLens* and $k = [1 \cdots 1500]$ on *Jester*, and "S" means $k = [1 \cdots 100]$.

The optimum values of RMSE (L), RMSE (S), NDPM (L) and NDPM (S) on *MovieLens*, respectively, are 0.9387, 0.9575, 0.3022 and 0.3221, which are all acquired by the middle value-based filling method. The optimum values of RMSE (L), RMSE (S) and NDPM (S) on *Jester*, respectively, are 4.481, 4.548 and 0.4142, which are all acquired by the user average-based filling method. The optimum value of NDPM (L) on *Jester*, 0.3897, is got by the middle value-based filling method. The optimum value of Recall on *MovieLens* is acquired by the middle value-based filling method whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 942]$, and that on *Jester* is got by the user average-based filling method whenever $k = [1 \cdots 100]$ or $k = [1 \cdots 1500]$.

We can see from the comparison between Tables 2 and 6 that NMF-*c*KNN is not always better than NMFP. What's more, the amount of calculation of NMF-*c*KNN is much larger than NMFP.

### 7.8. Eigentaste

Because the *MovieLens* data don't contain the gauge set which is necessary for Eigentaste, we only evaluated Eigentaste on the *Jester* data. The RMSE and NDPM of Eigentaste on *Jester* are 5.1442 and 0.4501 respectively. The Recall of Eigentaste is also always 1.0.

## 8. Overall comparisons among eight algorithms

Figs. 6 and 7 exhibit the comparisons among eight algorithms with appropriate configurations respectively on *MovieLens* and *Jester*.

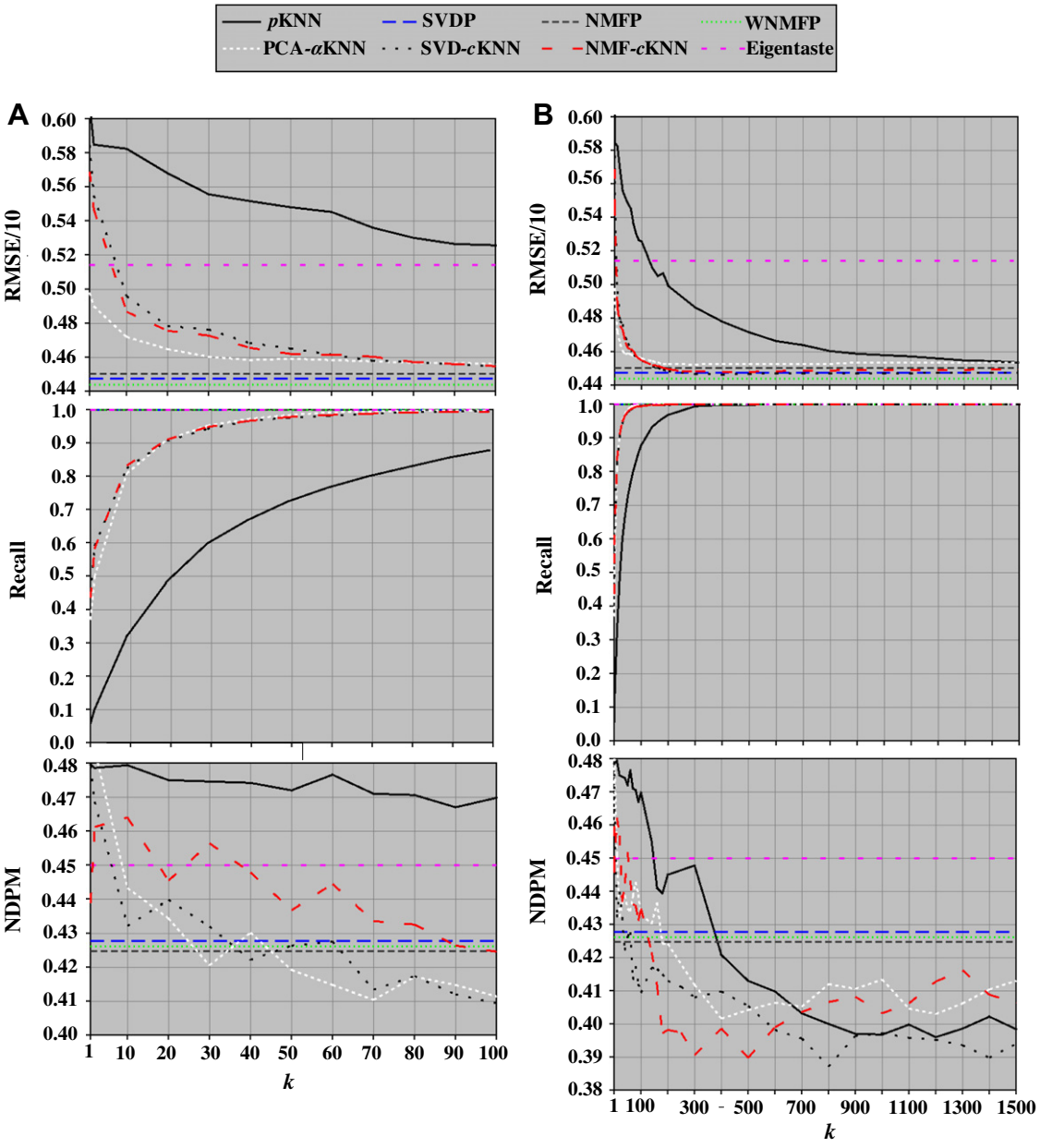


**Fig. 7.** (Online color) The RMSE, Recall and NDPM vs. $k$ for eight different algorithms with appropriate configurations on *Jester* when $k = [1 \cdots 100]$ (A) and $k = [1 \cdots 1500]$ (B).

Obviously, a larger value of $k$ means that the algorithm related to KNN needs a higher computational complexity. From the angle of the online recommendation speed, small values of $k$ can better serve the actual needs of users than its large values. It is well known that a hybrid algorithm needs more computation than its corresponding native algorithm.

We can see a few interesting points from Figs. 6 and 7.

When $k = [1 \cdots 100]$, $p$KNN is always the worst. Neither was $p$KNN best at larger values of $k$. This means that dimension-reducing techniques can help dig out more valuable information from the rating data than the native nearest-neighbor technique.

Three native dimension-reducing algorithms: SVDP, NMFP and WNMFP defeat four hybrid ones: PCA-$\alpha$KNN, SVD-$c$KNN, NMF-$c$KNN and Eigentaste at small values of $k$. This means that they are able to better satisfy users' actual needs, namely faster and better recommendation.

As $k$ grows larger, four algorithms related to KNN can approach (in RMSE, Recall and NDPM on *MovieLens* and in RMSE and Recall *Jester*) or exceed (in NDPM *Jester*) native dimension-reducing methods. This also means that the native dimension-reducing algorithms are in general more effective.

WNMFP is superior to the others in RMSE and Recall on both *MovieLens* and *Jester*. NMFP defeats the others in NDPM at all values of $k$ on *MovieLens* and at small values of $k$ on *Jester*. SVD-$c$KNN surpasses the others in NDPM at large values of $k$ on *Jester*. This unfolds two aspects. First, dimension-reducing techniques with non-negativity constraints are more effective than those not with non-negativity ones. Second, the decision on the optimum algorithm among eight algorithms is almost insensitive to the sparsity of dataset.

The processing time is an important factor. Table 7 exhibits an average processing time table covering the eight collaborative filtering algorithms with the optimum configurations when $k = [1 \cdots 100]$. The running environment is the dual core processor: 2.11 GHz and memory: 4G.

Clearly, as $k$ grows larger, the computational complexity of algorithms related to KNN will become higher and that of native dimension-reducing algorithms is unchanged. Therefore, Table 7 can represent the processing efficiency of eight algorithms. Obviously, three native dimension-reducing algorithms defeat these algorithms related to KNN in the processing time.

Considering the practical application, only when $k = [1 \cdots 100]$, the appropriate configurations and their corresponding values for eight algorithms are summarized in Table 8 where, through the comparison among the eight algorithms, the optimum cell of each metric is highlighted in bold italic type.

**Table 7**
An average processing time table (time in seconds) covering the eight collaborative filtering algorithms with the optimum configurations.

| | $p$ KNN | SVDP | NMFP | WNMFP | Eigentaste | PCA-$\alpha$ KNN | SVD-$c$ KNN | NMF-$c$ KNN |
|---|---|---|---|---|---|---|---|---|
| *MovieLens* | 165 | 24 | 27 | 29 | | 193 | 191 | 212 |
| *Jester* | 171 | 30 | 34 | 37 | 27 | 202 | 205 | 228 |

**Table 8**
The summarization of optimum configurations and their corresponding values for eight algorithms.

| | | $p$ KNN | SVDP | NMFP | WNMFP | Eigentaste | PCA-$\alpha$ KNN | SVD-$c$ KNN | NMF-$c$ KNN |
|---|---|---|---|---|---|---|---|---|---|
| *MovieLens* | RMSE | $K = 100$ | User average-based | Object average-based | ***Middle value-based*** | | Middle value-based | Object average-based | Middle value-based |
| | | 1.0615 | $d = 9$ | NNDSVD | ***SRI*** | | $d = 232$ | $d = 700$ | NNDSVD |
| | | | | ALS | ***MU*** | | | | ALS |
| | | | 0.979 | $d = 14/f = 5$ | ***$d = 2/f = 20$*** | | $k = 90$ | $k = 100$ | $d = 8/f = 20$ |
| | | | | 0.9739 | ***0.937*** | | 0.9633 | 0.9563 | $k = 100$ |
| | | | | | | | | | 0.9575 |
| | Recall | $k = 100$ | ***1.0*** | ***1.0*** | ***1.0*** | | User value-based | Object value-based | Middle value-based |
| | | 0.8276 | | | | | $k = 100$ | $k = 100$ | $k = 100$ |
| | | | | | | | 0.9762 | 0.9741 | 0.9704 |
| | NDPM | $k = 100$ | Object average-based | ***Object average-based*** | Object average-based | | Middle value-based | Object average-based | Middle value-based |
| | | 0.3766 | $d = 40$ | ***NNDSVD*** | SRI | | $d = 232$ | $d = 70$ | NNDSVD |
| | | | | ***ALS*** | MU | | | | ALS |
| | | | 0.2991 | ***$d = 16/f = 3$*** | $d = 600/f = 5$ | | $k = 50$ | $k = 100$ | $d = 5/f = 20$ |
| | | | | ***0.2906*** | 0.3053 | | 0.3314 | 0.3218 | $k = 100$ |
| | | | | | | | | | 0.3221 |
| *Jester* | RMSE/10 | $K = 100$ | User average-based | User average-based | ***Middle value-based*** | 0.5144 | Middle value-based | User average-based | User average-based |
| | | 0.5257 | $d = 13$ | SRI | ***NNDSVD*** | | $d = 76$ | $d = 20$ | NNDSVD |
| | | | | ALS | ***MU*** | | | | ALS |
| | | | 0.448 | $d = 20/f = 6$ | ***$d = 9/f = 2$*** | | $k = 100$ | $k = 100$ | $d = 20/f = 20$ |
| | | | | 0.4508 | ***0.4442*** | | 0.4561 | 0.455 | $k = 100$ |
| | | | | | | | | | 0.4548 |
| | Recall | $k = 100$ | ***1.0*** | ***1.0*** | ***1.0*** | ***1.0*** | User average-based | User average-based | User average-based |
| | | 0.8784 | | | | | $k = 100$ | $k = 100$ | $k = 100$ |
| | | | | | | | 0.9886 | 0.9887 | 0.9885 |
| | NDPM | $K = 90$ | User average-based | User average-based | Object average-based | 0.4501 | Object average-based | ***Object average-based*** | Object average-based |
| | | 0.4672 | $d = 9$ | NNDSVD | NNDSVD | | $d = 80$ | ***$d = 6$*** | NNDSVD |
| | | | | MU | MU | | | | ALS |
| | | | 0.4281 | $d = 10/f = 3$ | $d = 20/f = 9$ | | $k = 70$ | ***$k = 100$*** | $d = 5/f = 20$ |
| | | | | 0.425 | 0.4262 | | 0.4105 | ***0.4095*** | $k = 100$ |
| | | | | | | | | | 0.4248 |

The optimum values of RMSE and NDPM on *MovieLens* are 0.937 and 0.2906, respectively. They are acquired respectively by WNMFP with the optimum configuration of the middle value-based filling method, MU, SRI, $d = 2$ and $f = 100$, and NMFP with the optimum configuration of the object average-based filling method, ALS, NNDSVD, $d = 16$ and $f = 3$.

The optimum values of RMSE and NDPM on *Jester* are 4.442 and 0.4095 respectively. They are acquired respectively by WNMFP with the optimum configuration of the middle value-based filling method, MU, NNDSVD, $d = 9$ and $f = 2$, and SVD-*c*KNN with the optimum configuration of the object average-based filling method, $d = 6$ and $k = 100$.

Obviously, three native dimension-reducing algorithms (and Eigentaste) are steady winners in Recall.

It is clear that all the optimum values are not produced by the same matrix filling method, the same NMF method (ALS or MU), the same initialization method (SRI or NNDSVD) and the same values of $d$, $f$ and $k$. This means that the actual application demand decides proper selections on the matrix filling method, NMF method, initialization method and values of $d$, $f$ and $k$, that is, these proper selections are very often problem dependent.

It is noteworthy that the interests and tastes of a web user may change gradually over time. Therefore, the recommendation system should periodically update users' preference models.

## 9. Conclusion and future work

In this paper, we evaluated in detail eight representative collaborative filtering algorithms: *p*KNN, SVDP, NMFP, WNMFP, PCA-$\alpha$KNN, SVD-*c*KNN, NMF-*c*KNN and Eigentaste on two data sets: *Jester* and *MovieLens* by using three quality metrics: RMSE, Recall and NDPM.

Our work is very valuable. It not only provides a fair framework for the performance comparison between a new recommendation algorithm and these representative and well-accepted ones but also plays an important role in the right navigation for further innovation research on the technology of the personalization recommendation. We can summarize the results of our study in six main interesting findings. First, dimension-reducing techniques can dig out more latent valuable information than the nearest-neighbor technique. Second, in comparison with these four hybrid algorithms, namely PCA-$\alpha$KNN, SVD-*c*KNN, NMF-*c*KNN and Eigentaste, these three native algorithms only based on dimension-reducing techniques, namely SVDP, NMFP and WNMFP, are able to better satisfy users' actual needs in both speed and accuracy of recommendation. Third, dimension-reducing techniques with non-negativity constraints are more effective than those not with non-negativity ones. Fourth, the decision on the optimum algorithm among the eight ones is insensitive to the sparsity of dataset. Fifth, proper selections on the matrix filling method, NMF method, initialization method and values of $d$, $f$ and $k$ for eight algorithms are very often problem dependent. Sixth, native dimension-reducing algorithms can defeat these algorithms related to KNN in the processing time.

We have experienced deeply the importance of recommendation speed during the experiments. Although some high performance computing techniques such as computational grid, clustering technology, cloud computing, etc. have emerged, the widespread computing environment nowadays is still in great need of high-speed recommendation algorithms. Nowadays, neither accuracy nor speed is dispensable.

The effectiveness of a recommendation algorithm rests with the effective use of personal information of net users. Nowadays some web sites, such as *last.fm* and *delicious*, have taken the lead in guiding users with similar tastes to build trust relationships, that is, user on such a website can select some other users with similar tastes as his trusted friends. Trust information obviously contains reliable friend relations between users and therefore has potential to help in providing better recommendations. Based on the above thinking, next we will study how to effectively exploit trust information for better prediction.

## Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant No. 60774086, and two National High Technology Research and Development Programs of China ( two 863 Programs) under Grant Nos. 2007AA01Z475 and 2007AA01Z464, respectively.

## References

[1] T. Zhou, J. Ren, M. Medo, Y.C. Zhang, Bipartite network projection and personal recommendation, Physical Review E 76 (4) (2007) 046115.
[2] M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendation, Communications of the ACM 40 (3) (1997) 66–72.
[3] M.J. Pazzani, D. Billsus, Learning and revising user profiles: the identification of interesting Web sites, Machine Learning 27 (3) (1997) 313–331.
[4] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowledge-Based Systems 23 (6) (2010) 520–528.
[5] J.M. Yang, K.F. Li, D.F. Zhang, Recommendation based on rational inferences in collaborative filtering, Knowledge-Based Systems 22 (1) (2009) 105–114.
[6] F. Wang, P. Li, Efficient nonnegative matrix factorization with random projections, in: SDM, 2010, pp. 281–292.
[7] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions on Information Systems 22 (1) (2004) 89–115.
[8] D. Wei, T. Zhou, G. Cimini, P. Wu, W.-P. Liu, Y.-C. Zhang, Effective mechanism for social recommendation of news, Physica A 390 (2011) 2117–2126.
[9] M.-S. Shang, L. Lü, Y.-C. Zhang, T. Zhou, Empirical analysis of web-based user-object bipartite networks, Europhysics Letters 90 (2010) 48006.
[10] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Personalized recommendation via integrated diffusion on user-item-tag tripartite graph, Physica A 389 (2010) 179–186.
[11] M.S. Shang, Z.K. Zhang, T. Zhou, Y.C. Zhang, Collaborative filtering with diffusion-based similarity on tripartite graphs, Physica A 389 (6) (2010) 1259–1264.
[12] M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendations, Communications of the ACM 40 (3) (1997) 66–72.
[13] P. Melville, R.J. Mooney, R. Nagarajan, Content-boosted collaborative filtering for improved recommendations, in: Proceedings of the18th National Conference on Artificial Intelligence, 2002, pp. 187–192.
[14] M. Pazzani, A framework for collaborative, content-based and demographic filtering, Artificial Intelligence Review 13 (5) (1999) 393–408.
[15] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowledge-Based Systems 22 (4) (2009) 261–265.
[16] D. Parra-Santander, P. Brusilovsky, Evaluation of collaborative filtering algorithms for recommending articles, in: Proceedings of Web 3.0: Merging Semantic Web and Social Web at HyperText'09, 2009, pp. 3–6.
[17] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer in Computer 42 (8) (2009) 30–37.
[18] B.M. Sarwar, G. Karypis, J.A. Konstan, J.T. Riedl, Application of dimensionality reduction in recommender systems–a case study, in: ACM WebKDD Workshop, 2000.
[19] T. Li, J.D. Wang, A privacy-preserving collaborative filtering algorithm based on non-negative matrix factorization, Information and Control 37 (6) (2008) 660–664.
[20] S. Zhang, W.H. Wang, J. Ford, F. Makedon, Learning from incomplete ratings using non-negative matrix factorization, in: SDM, 2006, pp. 548–552.
[21] G. Shani, D. Heckerman, R.I. Brafman, An MDP-based recommender system, Journal of Machine Learning Research 6 (2005) 1265–1295.
[22] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Information Retrieval 4 (2) (2001) 133–151.
[23] T. Nathanson, E. Bitton, K. Goldberg, Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering, in: Proceedings of the 2007 ACM conference on Recommender systems, 2007, pp. 149–152.
[24] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by latent semantic analysis, Journal of the American Society of Information Science 41 (6) (1990) 391–407.
[25] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[26] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[27] Y. Mao, L.K. Saul, Modeling distances in large-scale networks by matrix factorization, in: Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement, 2004, pp. 278–287.

[28] C. Boutsidis, E. Gallopoulos, SVD based initialization: a head start for nonnegative matrix factorization, Pattern Recognition 41 (4) (2008) 1350–1362.
[29] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Computational Statistics & Data Analysis 52 (1) (2007) 155–173.
[30] O. Okun, H. Priisalu, Nonnegative matrix factorization for pattern recognition, in: Proceedings of the 5th IASTED International Conference on Visualization, Imaging and Image Processing, 2005, pp. 546–551.
[31] A.N. Langville, C.D. Meyer, R. Albright, Initializations for the nonnegative matrix factorization, in: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.
[32] I.T. Jolliffe, Principal Component Analysis, second ed., Springer, NY, 2002.
[33] J.G. Liu, T. Zhou, Q. Guo, B.H. Wang, Overview of the evaluated algorithms for the personal recommendation systems, Complex Systems and Complexity Science 6 (3) (2009) 1–10.
[34] Y.Y. Yao, Measuring retrieval effectiveness based on user preference of documents, Journal of the American Society for Information Science 46 (2) (1995) 133–145.
[35] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (2004) 5–53.
[36] MovieLens. Available from: <http://www.grouplens.org/node/12>. November 2006.
[37] Jester. Available from: <http://www.eigentaste.berkeley.edu/dataset/>. 2009.