Journal of Computer and System Sciences 81 (2015) 717–733

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

SVD-based incremental approaches for recommender systems

Xun Zhou [a,d], Jing He [b], Guangyan Huang [c], Yanchun Zhang [b,d,*]

[a] UCAS-VU Joint Lab for Social Computing and E-Health Research, University of Chinese Academy of Sciences, Beijing, China
[b] Centre for Applied Informatics, College of Engineering and Science, Victoria University, Australia
[c] School of Information Technology, Deakin University, Australia
[d] School of Computer Science, Fudan University, Shanghai, China

ARTICLE INFO

Article history:
Received 16 July 2014
Received in revised form 30 October 2014
Accepted 6 November 2014
Available online 16 December 2014

Keywords:
Singular value decomposition
Incremental algorithm
Recommender system
Experimental evaluation

ABSTRACT

Due to the serious information overload problem on the Internet, recommender systems have emerged as an important tool for recommending more useful information to users by providing personalized services for individual users. However, in the "big data" era, recommender systems face significant challenges, such as how to process massive data efficiently and accurately. In this paper we propose an incremental algorithm based on singular value decomposition (SVD) with good scalability, which combines the Incremental SVD algorithm with the Approximating the Singular Value Decomposition (ApproSVD) algorithm, called the Incremental ApproSVD. Furthermore, strict error analysis demonstrates the effectiveness of the performance of our Incremental ApproSVD algorithm. We then present an empirical study to compare the prediction accuracy and running time between our Incremental ApproSVD algorithm and the Incremental SVD algorithm on the MovieLens dataset and Flixster dataset. The experimental results demonstrate that our proposed method outperforms its counterparts.

1. Introduction

1.1. Background

With the popularity of the Internet and advances in information technology, information from websites tends to be too general and people require more personalized information. In order to meet users' demand for personalized services, personalized recommender systems are a powerful tool to solve the information overload problem. Collaborative filtering is one of the most important techniques used in recommender systems. Its principle is to recommend likely new information to an active user by considering other similar users' interests. It is based on the assumption that if two users have similar interests then the two users will probably share the same information. The advantages of collaborative filtering are as follows: first, it is independent of the contents of recommended items; second, it can be closely integrated with social networks; third, it has good accuracy in terms of recommendations.

The common challenge of collaborative filtering and other types of recommender systems is how to deal with massive data to make accurate recommendations. There are three difficulties [1]: (1) the huge amount of data, which requires the algorithm to respond quickly; (2) the sparsity of data, the ratings provided by the users or information which can be used to indicate interests are actually very sparse, compared with the large number of users and items in a recommender system; (3) the dynamic nature of data, which requires the algorithm to update quickly and accurately. The recommender system

* Corresponding author at: Centre for Applied Informatics, College of Engineering and Science, Victoria University, Australia.
E-mail address: yzhangvu@gmail.com (Y. Zhang).

http://dx.doi.org/10.1016/j.jcss.2014.11.016
0022-0000/

constantly acquires new data, and the active user's interests and concerns are constantly changing. In other words, when the user is active, such as browsing, clicking, rating, the training data is growing significantly. Therefore, these changes must be considered in the recommender system.

To solve the first two difficulties, many clustering or data dimensionality reduction methods have been proposed, including the clustering-based model, such as K-means clustering [2], minimum spanning tree [3], partition around medoids (PAM) [4], and so on; the matrix factorization-based model, such as singular value decomposition (SVD) [5], non-negative matrix factorization (NMF) [6], and so on; and the co-clustering method [1,7–9], which can cluster multiple dimensions simultaneously. These methods can effectively reduce the sparsity of data and reduce the consumption of online computation while improving accuracy, compared with the traditional method which calculates the similarity between users or items directly. However, the above methods have a disadvantage that they are offline computations, which cannot handle online and dynamic problems efficiently, so they are not adequate solutions to the aforementioned third difficulty.

To demonstrate the importance of online and dynamic computations for a recommender system, we give an example to illustrate this. For example, when a user enters a movie rating recommender system, which includes some ratings of movies rated by users, then users, movies and ratings are static in a recommender system. However, there are many new movies released every day that are constantly put into the recommender system, so we would like to predict how a user would rate a new movie in order to recommend the movies he/she might prefer. If a user can often obtain some good movie recommendations from a recommender system, he/she will trust this recommender system and will be happy to interact with the recommender system frequently. In this case, valid ratings from that active user in recommender system are growing, so that recommender systems can give more accurate predictions to a user. The above is a positive circle.

In this paper, we apply an incremental approach for the following reasons. If we simply replace the offline process with an online process, we should compute the singular value decomposition of some original unchanged matrix repeatedly each time. With the incremental approach, we only need to compute the singular value decomposition of the incremental part based on the singular value decomposition of the previous matrix, which can solve the problem of computational efficiency. Actually, the incremental approach is a good solution to resolve the aforementioned first and third difficulty.

1.2. Contributions

In the face of constantly updated information and changing users' interests, this paper proposes an incremental algorithm based on SVD called the Incremental ApproSVD, which integrates the new Incremental SVD algorithm with our previous ApproSVD algorithm [10].

How to conduct the Incremental ApproSVD algorithm for a recommender system is detailed below. The original static user-movie-rating data can be converted into a user-movie rating matrix B_1, and the newly entered user-movie-rating data can be converted into an added user-movie rating matrix B_2. Therefore, the practical problem is transformed into a matrix updating problem. First, the B_1, B_2 are considered as input in the Incremental ApproSVD algorithm. For B_1, we extract a constant number of columns from B_1, and scale the columns appropriately to form a relatively smaller matrix C_1. For B_2, the process of forming a relatively smaller matrix C_2 is similar to that of C_1. Second, we consider two matrices C_1 and C_2 as input and implement the Incremental SVD algorithm, which computes the SVD of $[C_1, C_2]$ based on the SVD of C_1. Finally, we obtain the left singular vectors of $[C_1, C_2]$, which are good approximations to the left singular vectors of $[B_1, B_2]$. To predict a user rating of a particular movie, we multiply the user preference vector by the movie feature vector, and obtain a predicted rating to indicate how much the user likes that movie. If the predicted rating is in the upper half of the rating range, the recommender system will recommend this particular movie to the user. If the predicted rating is in the lower half of the rating range, the recommender system will not recommend this particular movie to the user. Moreover, Section 4.3 conducts the error analysis and gives the upper bound. It is crucial that an appropriate number of columns of B_1 and B_2 are selected for sampling so as to reduce the upper bound.

Fig. 1 shows an example of the recommendation procedure based on the Incremental ApproSVD algorithm. The movie "Avatar" is a new movie entering a user-movie-rating recommender system, and we would like to predict whether Tom likes "Avatar". After implementing the Incremental ApproSVD algorithm, we can see that the predicted rating from Tom on "Avatar" is 2.4, which is in the lower half of the rating range $[1, 5]$. Therefore, the recommender system would not recommend "Avatar" to Tom.

The reason why the Incremental ApproSVD algorithm adopts column sampling to reduce the column number is that, after sampling some columns of the original rating matrix based on an appropriate sampling probability, the most representative columns are kept in the original rating matrix, which can effectively reduce the size of the original matrix. However, the user-movie-rating recommender system is growing fast, and many new movies enter the recommender system every day, so we should consider how to predict the new movies to a target user in order to keep the continuity of recommendation, and not just predict existing movies.

The contributions of this paper include the following three aspects: (1) it proposes an incremental algorithm, Incremental ApproSVD, which can predict unknown ratings when new items are dynamically entering a recommender system; (2) one of the most important features of our algorithm is that it can be easily realized, because it is a suboptimal approximation with lower running time compared with the Incremental SVD; (3) it gives the upper bound of error between the actual ratings and the predicted ratings generated by Incremental ApproSVD. Experiments show the advantages of our algorithm under different parameters on the MovieLens dataset and Flixster dataset.

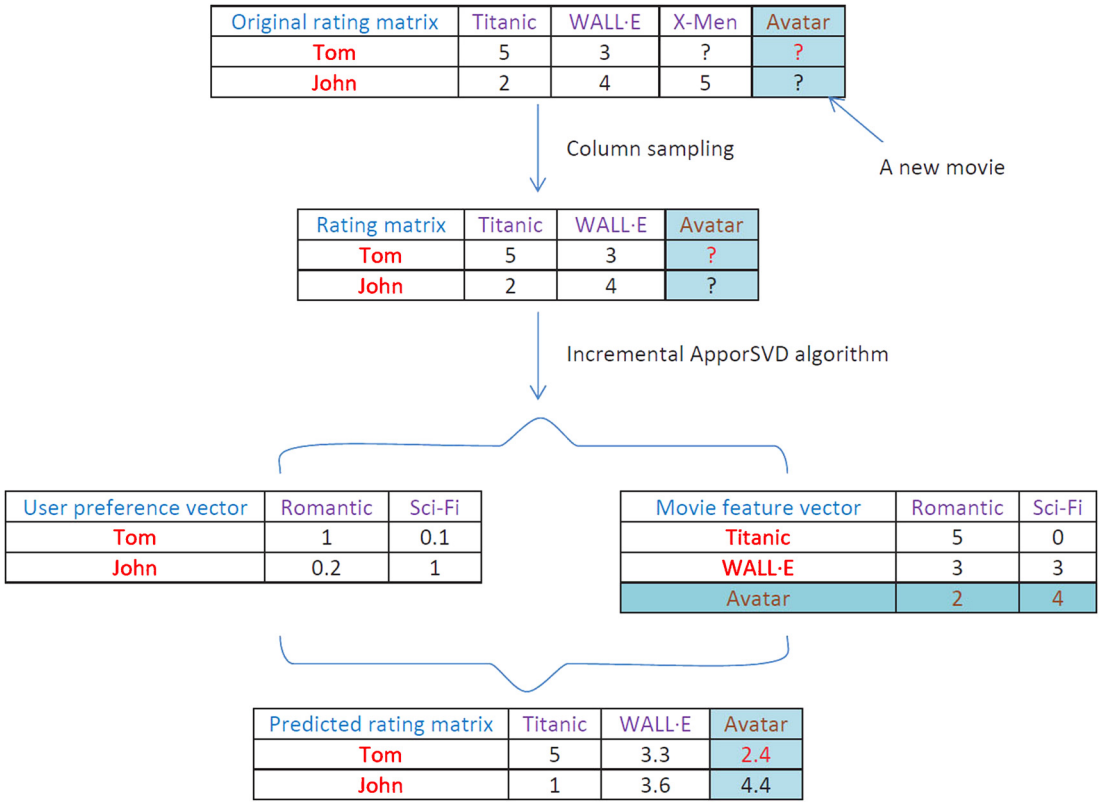

Fig. 1. Instance chart of recommendation procedure based on Incremental ApproSVD algorithm.

1.3. Organization

The remainder of this paper is organized as follows. First, Section 2 provides an overview of work related to the development of the incremental SVD and ApproSVD algorithms. Some preliminaries are introduced in Section 3. Section 4 presents the construction of two incremental algorithms, and gives the error analysis for the Incremental ApproSVD algorithm. Experimental evaluation results are reported in Section 5. Section 6 concludes the paper.

2. Related work

SVD is a basic mathematical method in data mining. SVD is usually calculated by batch, and the time complexity is $O(m^2n+n^3)$ [11] (m, n are the row size and column size of a matrix, respectively), meaning that all data must be processed immediately. Therefore, it is not feasible for very large dataset. Lanczos proposed an SVD method whose time complexity is $O(mnr)$ [11], where r is the rank of SVD. However, the Lanczos method requires the value of r in advance, and it is not accurate for small singular values [12,13].

In the last three decades, many scholars undertook research on how to update SVD by adding rows or columns [12,14–19]. SVD updating methods are mostly based on the Lanczos method, using Eq. (1) below. All uppercase letters in (1) denote general matrices and I denote the identity matrix. Zha and Simon [16] also apply Eq. (1), but the update is approximated and requires a dense SVD. Chandrasekaran et al. [20] use a similar method, however, their update is limited to a single vector and is vulnerable to a loss of orthogonality. Levy and Lindenbaum [21] exploit the relationship between the QR decomposition and the SVD to incrementally compute the left singular vectors in $O(mnr^2)$ time complexity. However, this is also vulnerable to a loss of orthogonality and results have only been reported for matrices having a few hundred columns

$$\begin{aligned}
&[U, J]\begin{bmatrix}\operatorname{diag}(s) & L\\ 0 & K\end{bmatrix}\begin{bmatrix}V & 0\\ 0 & I\end{bmatrix}^T\\
&\quad=\left[U, \left(I-UU^T\right)C/K\right]\begin{bmatrix}\operatorname{diag}(s) & U^TC\\ 0 & K\end{bmatrix}\begin{bmatrix}V & 0\\ 0 & I\end{bmatrix}^T\\
&\quad=\left[U\operatorname{diag}(s)V^T, C\right]=[M, C].
\end{aligned}\tag{1}$$

In previous work [10], we put forward an approximating SVD algorithm called ApproSVD, which combines the dimension reduction technique of SVD with the approximate method. The trick behind the ApproSVD algorithm is to sample some rows of a user-item matrix, rescale each row by an appropriate factor to form a relatively smaller matrix, and then reduce the dimensionality of the smaller matrix. However, ApproSVD cannot dynamically process growing massive data. Therefore, we will propose an incremental algorithm in Section 4.2 to solve this problem.

3. Preliminaries

3.1. Singular value decomposition (SVD)

SVD is a matrix factorization technique commonly used for producing low-rank approximations. Given a matrix $A \in \mathrm{R}^{m\times n}$ with $\mathrm{rank}(A) = r$, the Singular Value Decomposition of A is defined as the following:

$$A = USV^T, \tag{2}$$

where $U \in \mathrm{R}^{m\times m}$, $V \in \mathrm{R}^{n\times n}$ and $S \in \mathrm{R}^{m\times n}$. The matrices U, V are orthogonal, with their columns being the eigenvectors of AA^T and A^TA, respectively. The middle matrix S is a diagonal matrix with r nonzero elements, which are the singular values of A. Therefore, the effective dimensions of these three matrices U, S and V are $m \times r$, $r \times r$ and $n \times r$, respectively. The initial diagonal r elements $(\sigma_1, \sigma_2, \ldots, \sigma_r)$ of S have the property that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$.

An important property of SVD, which is particularly useful in recommender system, is that it can provide the optimal approximation to the original matrix A using three smaller matrices multiplication. By keeping the first k largest singular values in S and the remaining smaller ones set to zero, we denote this reduced matrix by S_k. Then by deleting the corresponding columns of U and V, which are the last $r-k$ columns of U and V, we denote these two reduced matrices by U_k and V_k, respectively. The truncated SVD is represented as

$$A_k = U_k S_k {V_k}^T, \tag{3}$$

which is the closest rank-k approximation to the original matrix A for any unitarily invariant norm [22,23].

3.2. The Eckart–Young theorem

Definition 1. Let $A = (a_{i,j})$ be an $n \times n$ square matrix. The trace of the matrix A is defined to be the sum of the main diagonal of A:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}. \tag{4}$$

Note that the trace is a linear transformation from the space of square matrices to the real numbers. In other words, for square matrices A and B, it is true that $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$.

Definition 2. The Frobenius norm $\|\cdot\|_F$ of $A \in \mathrm{R}^{m\times n}$ is defined as the following:

$$\|A\|_F^2 = \sum_{i,j} |a_{ij}|^2 = \mathrm{tr}\big(AA^T\big) = \mathrm{tr}\big(A^TA\big). \tag{5}$$

Note that the Frobenius norm satisfies the property $\|A\|_F^2 = \sigma_1^2 + \ldots + \sigma_r^2$, where $\sigma_1, \ldots, \sigma_r$ are the nonzero singular values of A.

Theorem 1. *(See Eckart and Young [24].) Let the SVD of A be given by Eq. (2). If $k < r = \mathrm{rank}(A)$ and $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, then*

$$\min_{\mathrm{rank}(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + \ldots + \sigma_r^2, \tag{6}$$

where σ_i is the i-th singular value, and u_i and v_i are the i-th columns of U and V, respectively.

4. Incremental algorithms

According to the existing rating D of user U for item I, we suppose that:

$$R^{D}_{User\times Item} : U \times I \to D. \tag{7}$$

This represents a two-dimensional prediction mapping function which can be converted into a user-item matrix with rating elements. Therefore, to calculate a prediction for any unknown rating, we can implement the matrix factorization and dimensionality reduction techniques.

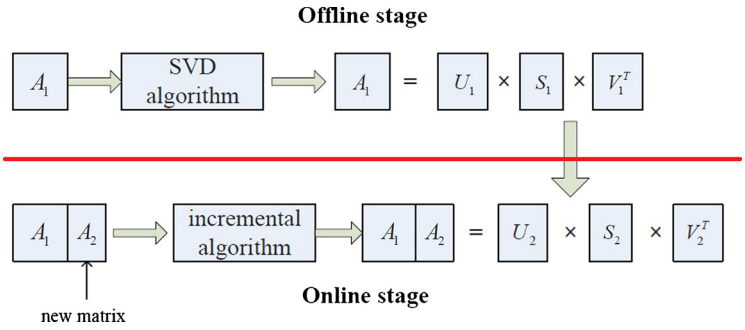

Fig. 2. Flow chart of offline and online stage of updating SVD.

In general, the entire algorithm works in two independent steps in a recommender system. The first step is the offline process and the second step is the online execution process. The user-user similarity and item-item similarity computation can be seen as the offline process of a collaborative filtering-based recommender system. Moreover, the actual prediction production can be seen as the online process. Usually, it is very time-consuming to do offline computations. Compared with the online process, the offline process is computed relatively infrequently. For example, a movie recommender system may calculate the user-user similarity or item-item similarity only once a day or even once a week. If the user-item ratings database is not dynamic and the user's taste does not change greatly over a short period of time, the similarity computation method may work well. Some scholars have demonstrated that the SVD-based dimensionality reduction algorithms can make the similarity formation process highly scalable while producing better results in most cases [25–27]. However, the SVD decomposition in the offline step has expensive computational requirements, which requires a running time of $O(m^3)$ for an $m \times n$ user-item matrix [14,28].

In order to overcome the expensive computation drawback of SVD decomposition, we introduce some incremental algorithms based on SVD with reasonable prediction quality, which can reduce the running time when making predictions. The key point of the incremental algorithms is to ensure highly scalable overall performance, which is very significant for the growing user-item matrix.

The incremental algorithm based on SVD shown in Fig. 2 is divided into an offline procedure and an online procedure. The offline stage is computationally intensive, and is performed only once. Finally, it can obtain three matrices U_1, S_1 and V_1 after performing SVD algorithm on A_1. The online stage is performed once the new matrix A_2 enters, and also produces three matrices U_2, S_2 and V_2 after performing the incremental algorithm on the updated matrix $[A_1, A_2]$, using the results of the offline part.

4.1. Incremental SVD algorithm

Consider a rating matrix A and the SVD decomposition $A = USV^T$, where $U \in \mathrm{R}^{m\times m}$, $S \in \mathrm{R}^{m\times n}$ and $V \in \mathrm{R}^{n\times n}$. Let a matrix $A_1 \in \mathrm{R}^{m\times p}$ be some column vectors to add to the matrix A. Therefore, we can write

$$A' := [A, A_1], \tag{8}$$

where $A' \in \mathrm{R}^{m\times(n+p)}$ is the updated matrix based on the original matrix A. The trick behind the incremental SVD algorithm is to take advantage of our knowledge of the SVD of A to compute the SVD of A'. Then Eq. (8) can be rewritten as follows:

$$\begin{aligned} A' &= [A, A_1] = \left[USV^T, A_1\right] \\ &= U\left[S, U^T A_1\right]\begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix} \\ &= UF\begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix} \\ &= U\left(U_F S_F V_F^T\right)\begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix} \\ &= (UU_F)S_F\left(\begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix} V_F\right)^T, \end{aligned} \tag{9}$$

where $F := [S, U^T A_1]$, and $SVD(F) = U_F S_F V_F^T$. Finally, the SVD of A' can be calculated as

$$A' = U_{A'} S_{A'} V_{A'}^T, \tag{10}$$

Algorithm 1 Incremental SVD algorithm.

Input: matrices $A_1 \in \mathrm{R}^{m\times n_1}$, $A_2 \in \mathrm{R}^{m\times n_2}$; parameter $k \in Z^+$ satisfies $1 \le k \le \min\{m, n_1+n_2\}$.
Output: matrices $U_k \in \mathrm{R}^{m\times k}$, $S_k \in \mathrm{R}^{k\times k}$, $V_k \in \mathrm{R}^{(n_1+n_2)\times k}$.

Using SVD on matrix A_1, we get three matrices U_1, S_1, V_1;
$F \leftarrow [S_1, U_1^T A_2]$, using SVD to decompose F and keep the rank-k approximation, we get three matrices U_F, S_F, V_F;
$b \leftarrow size(S_1, 2)$;
if $b < n_1$ **then**
    $V_k \leftarrow [V_1, zeros(n_1, n_2); zeros(n_2, b), eye(n_2)] \cdot V_F$;
else
    $V_k \leftarrow [V_1, zeros(n_1, n_2); zeros(n_2, n_1), eye(n_2)] \cdot V_F$;
end if
$U_k \leftarrow U_1 \cdot U_F$;
$S_k \leftarrow S_F$;
Return user eigenvector of $[A_1, A_2]$, i.e. $U_k\sqrt{S_k}(i)$ and item eigenvector of $[A_1, A_2]$, i.e. $\sqrt{S_k}V_k^T(j)$.

where $U_{A'} := UU_F$, $S_{A'} := S_F$ and $V_{A'} := \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix} V_F$. The matrices $U_{A'}$, $V_{A'}$ are orthogonal, which is a very important property so that we can keep on updating the matrix A' through a similar process.

Let a matrix $A_2 \in \mathrm{R}^{m\times q}$ be some column vectors to add to the matrix A'. Therefore, we can write

$$A'' := [A, A_1, A_2] = \left[A', A_2\right], \tag{11}$$

where $A'' \in \mathrm{R}^{m\times(n+p+q)}$ is the updated matrix based on the matrix A'. The trick behind the incremental SVD algorithm is to take advantage of our knowledge of the SVD of A' to compute the SVD of A''. Then Eq. (11) can be rewritten as follows:

$$\begin{aligned} A'' &= \left[A', A_2\right] = \left[U_{A'}S_{A'}V_{A'}^T, A_2\right] \\ &= U_{A'}\left[S_{A'}, U_{A'}^T A_2\right]\begin{bmatrix} V_{A'}^T & 0 \\ 0 & I \end{bmatrix} \\ &= U_{A'}G\begin{bmatrix} V_{A'}^T & 0 \\ 0 & I \end{bmatrix} \\ &= U_{A'}\left(U_G S_G V_G^T\right)\begin{bmatrix} V_{A'}^T & 0 \\ 0 & I \end{bmatrix} \\ &= (U_{A'}U_G)S_G\left(\begin{bmatrix} V_{A'} & 0 \\ 0 & I \end{bmatrix} V_G\right)^T, \end{aligned} \tag{12}$$

where $G := [S_{A'}, U_{A'}^T A_2]$, and $SVD(G) = U_G S_G V_G^T$. Finally, the SVD of A'' can be calculated as

$$A'' = U_{A''}S_{A''}V_{A''}^T, \tag{13}$$

where $U_{A''} := U_{A'}U_G$, $S_{A''} := S_G$ and $V_{A''} := \begin{bmatrix} V_{A'} & 0 \\ 0 & I \end{bmatrix} V_G$. Notice that, the matrices $U_{A''}$, $V_{A''}$ are orthogonal too.

Berry et al. [29] detail the process of the SVD-updating method and give the SVD-updating example for a term-document matrix. Similarly, we can view a term-document matrix as a user-item matrix. SVD-updating incorporates new user or item information into an existing recommender system using three factor matrices. That is, SVD-updating exploits the previous singular values and singular vectors of the original user-item matrix A as an alternative to recomputing the SVD of the updated matrix A'. The update procedure of the Incremental SVD algorithm is shown in Algorithm 1.

4.2. Incremental ApproSVD algorithm

Based on the ApproSVD algorithm proposed in our previous paper [10], we combine it with the Incremental SVD and propose the Incremental ApproSVD algorithm, which could solve the scalability problem in a recommender system. In the Incremental ApproSVD algorithm, the most important aspect is how to choose the column sampling probabilities $\{p_i'\}_{i=1}^{n_1}$ and $\{p_j''\}_{j=1}^{n_2}$. In Section 5.2, when undertaking performance comparisons between two algorithms, for B_1, we choose the column sampling probabilities as $p_i' = nnz(B_1^{(i)})/nnz(B_1)$, for $i = 1, \ldots, n_1$, where $nnz(B_1^{(i)})$ denotes the number of nonzero elements in the i-th column of matrix B_1 and $nnz(B_1)$ denotes the number of nonzero elements in matrix B_1; for B_2, we choose the column sampling probabilities as $p_j'' = nnz(B_2^{(j)})/nnz(B_2)$, for $j = 1, \ldots, n_2$, where $nnz(B_2^{(j)})$ denotes the number of nonzero elements in the j-th column of matrix B_2 and $nnz(B_2)$ denotes the number of nonzero elements in matrix B_2. The process of Incremental ApproSVD algorithm is described in Algorithm 2.

The flow chart of the Incremental ApproSVD algorithm is shown in Fig. 3, where the marks ①, …, ⑩ correspond to the step numbers in Algorithm 2.

Algorithm 2 Incremental ApproSVD algorithm.

Input: matrices $B_1 \in \mathbb{R}^{m\times n_1}$, $B_2 \in \mathbb{R}^{m\times n_2}$; parameters $c_1, c_2 \in Z^+$ satisfy $c_1 \le n_1$, $c_2 \le n_2$; parameter $k \in Z^+$ satisfies $1 \le k \le \min\{m, c_1 + c_2\}$; for B_1, column sampling probabilities $\{p'_i\}_{i=1}^{n_1}$ satisfy $p'_i \ge 0$ and $\sum_{i=1}^{n_1} p'_i = 1$; for B_2, column sampling probabilities $\{p''_j\}_{j=1}^{n_2}$ satisfy $p''_j \ge 0$ and $\sum_{j=1}^{n_2} p''_j = 1$.

Output: $H_k \in \mathbb{R}^{m\times k}$.

1: **for** $t = 1 \to c_1$ **do**
2: Pick $i_t \in 1, \ldots, n_1$ under sampling probabilities p'_α, $\alpha = 1, \ldots, n_1$ (i_t denotes the column index of B_1);
3: $C_1^{(t)} \leftarrow B_1^{(i_t)} / \sqrt{c_1 p'_{i_t}}$ (a column vector $B_1^{(i)}$ denotes the i-th column of B_1);
4: **end for**
5: **for** $t = 1 \to c_2$ **do**
6: Pick $j_t \in 1, \ldots, n_2$ under sampling probabilities p''_α, $\alpha = 1, \ldots, n_2$ (j_t denotes the column index of B_2);
7: $C_2^{(t)} \leftarrow B_2^{(j_t)} / \sqrt{c_2 p''_{j_t}}$ (a column vector $B_2^{(j)}$ denotes the j-th column of B_2);
8: **end for**
9: C_1 and C_2 can replace A_1 and A_2 as the **Input** of Algorithm 1. Run Algorithm 1 and skip Steps 3–8, we can get U_k;
10: $H_k \leftarrow U_k$;
11: **Return** $H_k \in \mathbb{R}^{m\times k}$.

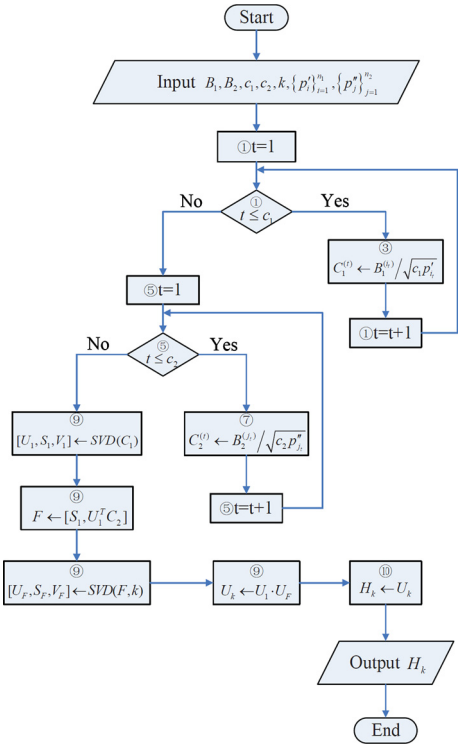

Fig. 3. Flow chart of Incremental ApproSVD.

4.3. Error analysis

As stated above, approximating $[B_1, B_2]$ by $[B_1, B_2]_k$ generates an error equal to $\|[B_1, B_2] - [B_1, B_2]_k\|_F^2 = \sum_{t=k+1}^{r} \sigma_t^2([B_1, B_2])$, where $r = \text{rank}([B_1, B_2])$ and $\sigma_t([B_1, B_2])$ are the singular values of $[B_1, B_2]$. Therefore, $[B_1, B_2]_k$ is the optimal rank-k approximation to $[B_1, B_2]$. We will show that, except for this error, the matrix $H_k H_k^T [B_1, B_2]$ has an additional error that mainly depends on $\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\|_F$.

Theorem 2. *Suppose $B_1 \in \mathbb{R}^{m\times n_1}$, $B_2 \in \mathbb{R}^{m\times n_2}$ and let C_1, C_2, H_k be constructed from the Incremental ApproSVD algorithm. Then*

$$\left\|[B_1, B_2] - H_k H_k^T [B_1, B_2]\right\|_F^2 \le \left\|[B_1, B_2] - [B_1, B_2]_k\right\|_F^2 + 2\sqrt{k}\left\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\right\|_F. \tag{14}$$

Proof. Before starting the proof, let us recall some basic properties of a matrix in the following: $\|X\|_F^2 = \text{tr}(XX^T) = \text{tr}(X^T X)$, for any matrix X; $\text{tr}(X + Y) = \text{tr}(X) + \text{tr}(Y)$, for square matrices X and Y. From the Incremental ApproSVD algorithm, we also have $H_k^T H_k = I_k$.

Therefore, $\|[B_1, B_2] - H_k H_k^T [B_1, B_2]\|_F^2$ can be expressed as

$$\begin{aligned}
&\left\|[B_1, B_2] - H_k H_k^T [B_1, B_2]\right\|_F^2 \\
&= \operatorname{tr}\left(\left([B_1, B_2] - H_k H_k^T [B_1, B_2]\right)^T \left([B_1, B_2] - H_k H_k^T [B_1, B_2]\right)\right) \\
&= \operatorname{tr}\left([B_1, B_2]^T [B_1, B_2] - 2[B_1, B_2]^T H_k H_k^T [B_1, B_2] + [B_1, B_2]^T H_k H_k^T H_k H_k^T [B_1, B_2]\right) \\
&= \operatorname{tr}\left([B_1, B_2]^T [B_1, B_2]\right) - \operatorname{tr}\left([B_1, B_2]^T H_k H_k^T [B_1, B_2]\right) \\
&= \left\|[B_1, B_2]\right\|_F^2 - \left\|[B_1, B_2]^T H_k\right\|_F^2. \qquad (15)
\end{aligned}$$

The error between $\|[B_1, B_2]^T H_k\|_F^2$ and $\sum_{t=1}^{k} \sigma_t^2([C_1, C_2])$ can be produced by the following:

$$\begin{aligned}
&\left|\left\|[B_1, B_2]^T H_k\right\|_F^2 - \sum_{t=1}^{k} \sigma_t^2\left([C_1, C_2]\right)\right| \\
&\le \sqrt{k}\left(\sum_{t=1}^{k}\left(\left|[B_1, B_2]^T h^t\right|^2 - \sigma_t^2\left([C_1, C_2]\right)\right)^2\right)^{\frac{1}{2}} \\
&= \sqrt{k}\left(\sum_{t=1}^{k}\left(\left|[B_1, B_2]^T h^t\right|^2 - \left|[C_1, C_2]^T h^t\right|^2\right)^2\right)^{\frac{1}{2}} \\
&= \sqrt{k}\left(\sum_{t=1}^{k}\left(h^{tT}\left([B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\right)h^t\right)^2\right)^{\frac{1}{2}} \\
&\le \sqrt{k}\left\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\right\|_F. \qquad (16)
\end{aligned}$$

The error between $\sum_{t=1}^{k} \sigma_t^2([C_1, C_2])$ and $\sum_{t=1}^{k} \sigma_t^2([B_1, B_2])$ can also be produced by the following:

$$\begin{aligned}
&\left|\sum_{t=1}^{k} \sigma_t^2\left([C_1, C_2]\right) - \sum_{t=1}^{k} \sigma_t^2\left([B_1, B_2]\right)\right| \\
&\le \sqrt{k}\left(\sum_{t=1}^{k}\left(\sigma_t^2\left([C_1, C_2]\right) - \sigma_t^2\left([B_1, B_2]\right)\right)^2\right)^{\frac{1}{2}} \\
&= \sqrt{k}\left(\sum_{t=1}^{k}\left(\sigma_t\left([C_1, C_2][C_1, C_2]^T\right) - \sigma_t\left([B_1, B_2][B_1, B_2]^T\right)\right)^2\right)^{\frac{1}{2}} \\
&\le \sqrt{k}\left(\sum_{t=1}^{m}\left(\sigma_t\left([C_1, C_2][C_1, C_2]^T\right) - \sigma_t\left([B_1, B_2][B_1, B_2]^T\right)\right)^2\right)^{\frac{1}{2}} \\
&\le \sqrt{k}\left\|[C_1, C_2][C_1, C_2]^T - [B_1, B_2][B_1, B_2]^T\right\|_F. \qquad (17)
\end{aligned}$$

Combining the results of inequality (16) with (17), the error between $\|[B_1, B_2]^T H_k\|_F^2$ and $\sum_{t=1}^{k} \sigma_t^2([B_1, B_2])$ can be produced by the following:

$$\left|\left\|[B_1, B_2]^T H_k\right\|_F^2 - \sum_{t=1}^{k} \sigma_t^2\left([B_1, B_2]\right)\right| \le 2\sqrt{k}\left\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\right\|_F. \qquad (18)$$

According to (18), formula (15) can be rewritten and yields formula (14) by the following:

$$\begin{aligned}
&\left\|[B_1, B_2] - H_k H_k^T [B_1, B_2]\right\|_F^2 \\
&= \left\|[B_1, B_2]\right\|_F^2 - \left\|[B_1, B_2]^T H_k\right\|_F^2 \\
&\le \left|\left\|[B_1, B_2]\right\|_F^2 - \sum_{t=1}^{k} \sigma_t^2\left([B_1, B_2]\right)\right| + \left|\sum_{t=1}^{k} \sigma_t^2\left([B_1, B_2]\right) - \left\|[B_1, B_2]^T H_k\right\|_F^2\right| \\
&\le \left\|[B_1, B_2] - [B_1, B_2]_k\right\|_F^2 + 2\sqrt{k}\left\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\right\|_F. \qquad (19)
\end{aligned}$$

Now, we complete the proof. □

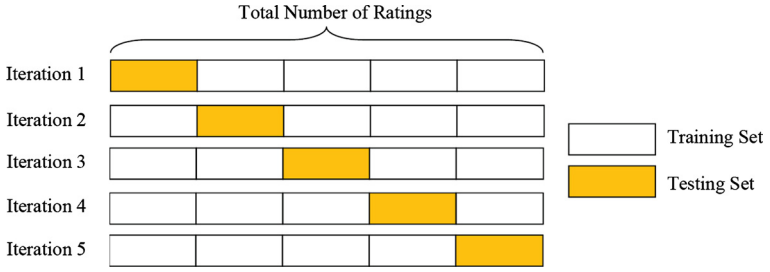

Fig. 4. Five-fold cross validation.

From Theorem 2, the error analysis shows that when parameter k grows bigger, the first part $\|[B_1, B_2] - [B_1, B_2]_k\|_F^2$ will reduce and the second part $2\sqrt{k}\|[B_1, B_2][B_1, B_2]^T - [C_1, C_2][C_1, C_2]^T\|_F$ will increase which are on the right side of inequality (14). As mentioned above, $[B_1, B_2]_k$ is the optimal rank-k approximation to $[B_1, B_2]$. So, the error between $[B_1, B_2]$ and $H_k H_k^T [B_1, B_2]$ is mainly determined by the second part of the right side of (14).

In the Incremental ApproSVD algorithm, the most important aspect is how to choose the column sampling probabilities $\{p_i'\}_{i=1}^{n_1}$ and $\{p_j''\}_{j=1}^{n_2}$, which is described in previous Section 4.2. The column sampling probabilities can be used to choose columns to be sampled in one pass and $O(c_1 + c_2)$ additional space and time.

5. Experimental evaluations

This section describes the experimental validation of our personalized recommendation algorithm – the Incremental ApproSVD algorithm. We first propose our experimental platform – the datasets, the evaluation metric, and the computational environment. Then, we introduce our experimental process, followed by the results and discussions.

5.1. Experimental platform

Datasets. We use two real datasets to conduct the experiment: MovieLens and Flixster.

The first dataset we used is provided by the MovieLens group [30]. MovieLens is a web-based research recommender system which commenced in autumn 1997. Each week, hundreds of users visit MovieLens to rate and receive recommendations for movies. The dataset can be converted into a user-item matrix that has 943 rows (users) and 1682 columns (movies), in which approximately 6.3% of entries are filled. The matrix has 100,000 ratings and all unrated items have a value of zero. The ratings range from 1 to 5, where 1 represents dislike and 5 represents a strong preference.

The second dataset we used is now publicly available at http://www.sfu.ca/~sja25/datasets/. Flixster is a dataset of movie ratings from the Flixster commercial website [31], which contains ratings expressed by users during the period from November 2005 to November 2009. The dataset has 786,936 users, 48,794 items and 8,196,077 ratings. We select a small portion of original Flixster dataset which satisfies the following two conditions at the same time: (1) keep a user who rated at least 250 items; (2) keep an item that has at least 30 ratings. The dataset could be converted into a user-item matrix that has 8465 rows (users), 9602 columns (movies) and 5,326,788 nonzero elements (ratings), in which approximately 6.55% of entries are filled. Possible rating values in Flixster are 10 discrete numbers in the range $[0.5, 5]$ with step size 0.5, where the minimum number represents dislike and the maximum number represents strong preference.

Available rating entries in each dataset are randomly divided into five partitions for five-fold cross validation, which is shown in Fig. 4. In other words, we randomly divide all ratings evenly into five disjoint folds and apply four folds together to train our algorithm, and use the remaining fold as a test set to evaluate the performance. We repeat this process five times for each dataset so that each fold is used as a test set once. Algorithms are then performed on training cases to make predictions for test cases.

Evaluation metric. The primary purpose of recommender systems is to predict users' underlying preferences and interests and recommend the right items to users to find their likes from a growing number of items. There are a lot of metrics to measure various aspects of recommendation performance. Two popular metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are used to measure the closeness of predicted ratings to the true ratings. If r_{ui} denotes the true rating on item i by user u, and $\hat{r}_{ui}$ denotes the predicted rating on item i by user u, MAE and RMSE of N corresponding rating-prediction pairs are defined as:

$$MAE = \frac{\sum_{i=1}^{N} |r_{ui} - \hat{r}_{ui}|}{N}, \tag{20}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (r_{ui} - \hat{r}_{ui})^2}{N}}. \tag{21}$$

The smaller MAE and RMSE values correspond to the higher accuracy of the recommender system. As the error is squared before being summed in the calculation expression of RMSE, it tends to penalize large errors more heavily.

Table 1
Performance comparisons for the MovieLens dataset under the same k.

Incremental ApproSVD								Incremental SVD					
n_1	n_2	c_1	c_2	k	RMSE	MAE	Time (sec.)	n_1	n_2	k	RMSE	MAE	Time (sec.)
900	100	500	50	10	**0.9673**	**0.7686**	**748.37**	500	50	10	0.9511	0.7530	810.88
900	100	600	50	10	0.9643	0.7658	750.51	600	50	10	0.9502	0.7522	1016.88
900	100	700	50	10	0.9612	0.7630	752.11	700	50	10	0.9500	0.7530	1038.56
900	100	800	50	10	0.9580	0.7604	756.47	800	50	10	0.9507	0.7537	1043.09

Table 2
Performance comparisons for the MovieLens dataset under different k.

Incremental ApproSVD								Incremental SVD					
n_1	n_2	c_1	c_2	k	RMSE	MAE	Time (sec.)	n_1	n_2	k	RMSE	MAE	Time (sec.)
900	100	800	50	10	**0.9580**	**0.7604**	**756.47**	800	50	10	0.9507	0.7537	1043.09
900	100	800	50	100	0.9815	0.7799	765.95	800	50	100	0.9779	0.7759	1057.94
900	100	800	50	400	1.0013	0.7991	768.37	800	50	400	0.9963	0.7949	1310.70
900	100	800	50	600	1.0024	0.8003	772.22	800	50	600	0.9971	0.7956	1333.10

Environment. All codes were written in MATLAB R2013b. All our experiments were executed on a computer with a 2.90 GHz Intel(R) Core(TM) i5-4570S CPU processor and 8 GB of RAM, running on 64-bit Microsoft Windows 7 Enterprise Edition.

5.2. Evaluation results and discussions

As described above, the most important aspect of the Incremental ApproSVD algorithm is the column sampling procedure. From Theorem 1, we find that it holds no matter how the column sampling probabilities $\{p'_i\}_{i=1}^{n_1}$ and $\{p''_j\}_{j=1}^{n_2}$ are chosen.

In order to obtain a lower value of MAE and RMSE with less running time, we choose the column sampling probabilities as $p'_i = nnz(B_1{}^{(i)})/nnz(B_1)$, $i = 1, \ldots, n_1$ and $p''_j = nnz(B_2{}^{(j)})/nnz(B_2)$, $j = 1, \ldots, n_2$. Then, we scale the columns prior to including them in C_1 and C_2. Furthermore, we compute the SVD of $[C_1, C_2]$ based on the SVD of C_1 by implementing the Incremental SVD algorithm. Finally, we obtain the matrix H_k whose columns are the left singular vectors of $[C_1, C_2]$. So we can predict the rating of user u on movie i by $H_k H_k^T[B_1, B_2](u, i)$. We substitute 1 for the elements less than 1 and 5 for the elements more than 5 in matrix $H_k H_k^T[B_1, B_2]$ when conducting experiments on the MovieLens dataset, since the ratings range from 1 to 5 in the MovieLens dataset. Moreover, we substitute 0.5 for the elements less than 0.5 and 5 for the elements more than 5 in matrix $H_k H_k^T[B_1, B_2]$ when conducting experiments on the Flixster dataset, since the ratings range from 0.5 to 5 in the Flixster dataset.

Following are the RMSE, MAE and running time for the MovieLens 100 K dataset, using five-fold cross validation, changing parameters c_1, k for our Incremental ApproSVD and parameters n_1, k for Incremental SVD. The performance comparisons between the two algorithms are shown in Tables 1–2.

In Table 1, the first column is the column size of original matrix B_1, and we select the same value $n_1 = 900$. The second column is the column size of added matrix B_2, and we select the same value $n_2 = 100$. The third column is the number of columns picked for matrix C_1 from B_1, and we select four values $c_1 = 500, 600, 700$ and 800. The fourth column is the number of columns picked for matrix C_2 from B_2, and we select the same value $c_2 = 50$. The fifth column is the dimension of eigenvector, and we select the same value $k = 10$. The right side of three adjacent columns are the prediction accuracy values and running time of the Incremental ApproSVD algorithm. The ninth column is the column size of original matrix A_1, and we select four values $n_1 = 500, 600, 700$ and 800, which is the same size as that of matrix C_1 in the Incremental ApproSVD algorithm. The tenth column is the column size of added matrix A_2, and we select the same value $n_2 = 50$, which is the same size as that of matrix C_2 in the Incremental ApproSVD algorithm. The eleventh column is the dimension of eigenvector, and we select the same value $k = 10$. The rightmost three columns are the prediction accuracy values and running time of the Incremental SVD algorithm. From Table 1, by observing the performance of Incremental ApproSVD, we can see that RMSE and MAE values decrease with the increase of c_1. Moreover, the running time is increasingly modest. By observing the performance of Incremental SVD, we can see that the trends of RMSE and MAE values are not significant with the increase of n_1 and the running time is increasingly sharp.

The two curves shown in Fig. 5 demonstrate the different trend of RMSE values according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the MovieLens dataset. Two parameters c_1 and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 5 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

The two curves shown in Fig. 6 demonstrate the different trend of MAE values according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the MovieLens dataset. Two parameters

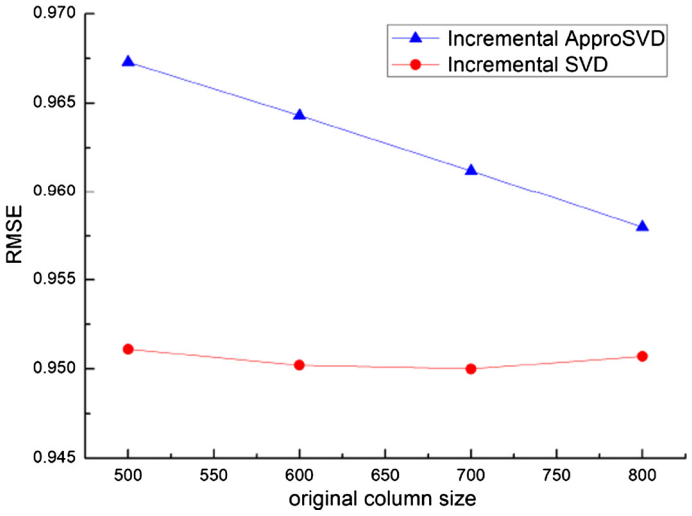

Fig. 5. RMSE values for MovieLens dataset with the change of original column size.

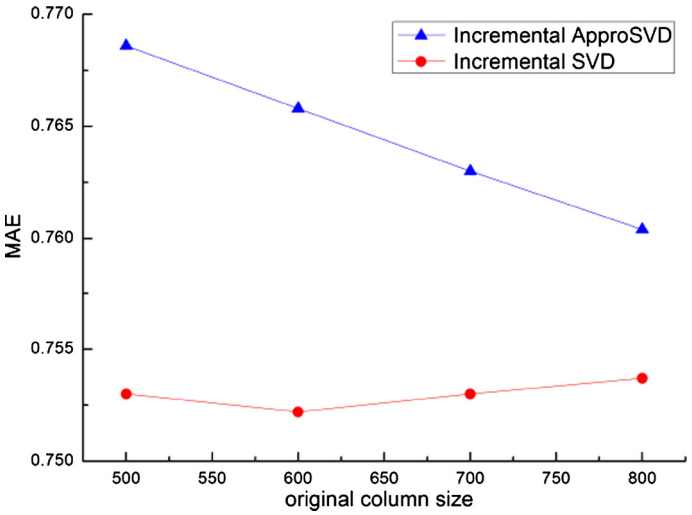

Fig. 6. MAE values for the MovieLens dataset with the change of original column size.

c_1 and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 6 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

The two curves shown in Fig. 7 demonstrate the different trend of running time (seconds) according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the MovieLens dataset. Two parameters c_1 and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 7 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

In Table 2, the first, the second, the fourth and the tenth columns are the same as that in the above Table 1. The third column is the number of columns picked for matrix C_1 from B_1, and we select the same value $c_1 = 800$. The fifth column is the dimension of eigenvector, and we select four values $k = 10, 100, 400$ and 600. The right side of the three adjacent columns are the prediction accuracy values and running time of the Incremental ApproSVD algorithm. The ninth column is the column size of original matrix A_1, and we select the same value $n_1 = 800$, which is the same size as that of matrix C_1 in the Incremental ApproSVD algorithm. The eleventh column is the dimension of eigenvector, and we select four values $k = 10, 100, 400$ and 600. The rightmost three columns are the prediction accuracy values and running time of the Incremental SVD algorithm. From Table 2, we can see that RMSE and MAE values increase with the increase of k for both Incremental ApproSVD and Incremental SVD. Moreover, the running time of Incremental ApproSVD is increasingly modest. However, the running time of Incremental SVD is increasingly sharp.

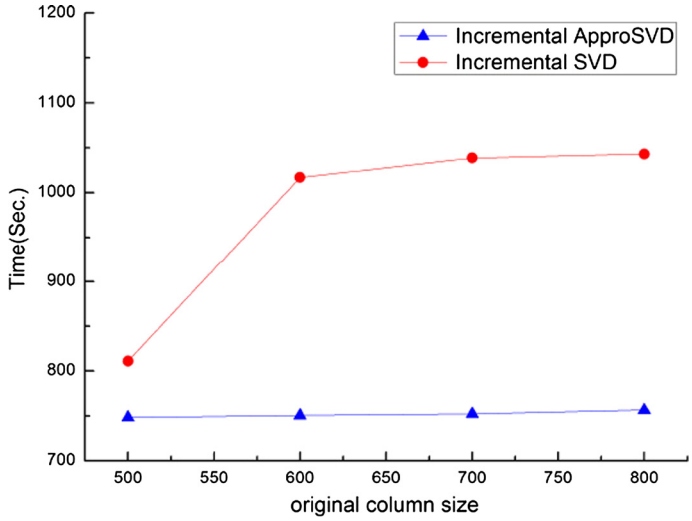

Fig. 7. Running time for the MovieLens dataset with the change of original column size.

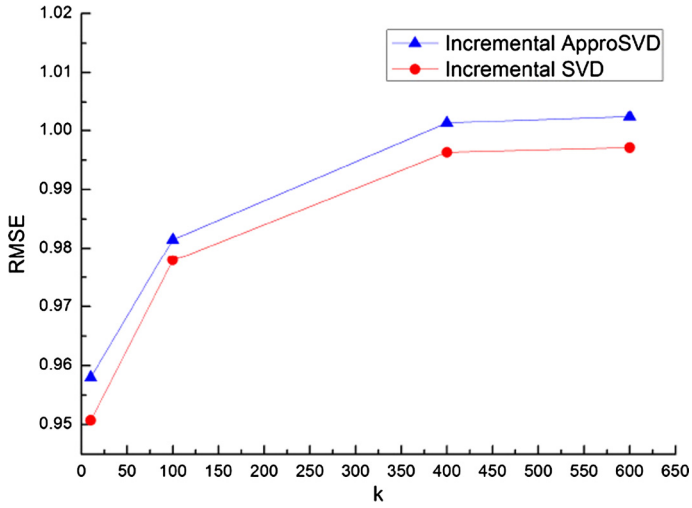

Fig. 8. RMSE values for the MovieLens dataset with the change of k.

The two curves shown in Fig. 8 demonstrate the different trend of RMSE values according to a gradual increase in k for the MovieLens dataset.

The two curves shown in Fig. 9 demonstrate the different trend of MAE values according to a gradual increase in k for the MovieLens dataset.

The two curves shown in Fig. 10 demonstrate the different trend of running time according to a gradual increase in k for the MovieLens dataset.

In short, for the same sized matrices and the same reduced dimension, although RMSE and MAE values of Incremental ApproSVD are a little higher than that of Incremental SVD, the running time of Incremental ApproSVD is much shorter than that of Incremental SVD. Therefore, the overall performance of Incremental ApproSVD outperforms that of Incremental SVD for the MovieLens dataset.

Following are the RMSE, MAE and running time for the Flixster dataset, using five-fold cross validation, changing parameters c_1, k for our Incremental ApproSVD and parameters n_1, k for Incremental SVD. The performance comparisons between the two algorithms are shown in Tables 3–4.

In Table 3, all columns represent the same meanings as described by Table 1. From Table 3, by observing the performance of Incremental ApproSVD, we can see that RMSE and MAE values decrease with the increase of c_1. Furthermore, little change has occurred for the running time. By observing the performance of Incremental SVD, we can see that the trends of RMSE and MAE values are not significant with the increase of n_1 and the running time is increasingly sharp.

The two curves shown in Fig. 11 demonstrate the different trend of RMSE values according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the Flixster dataset. Two parameters c_1

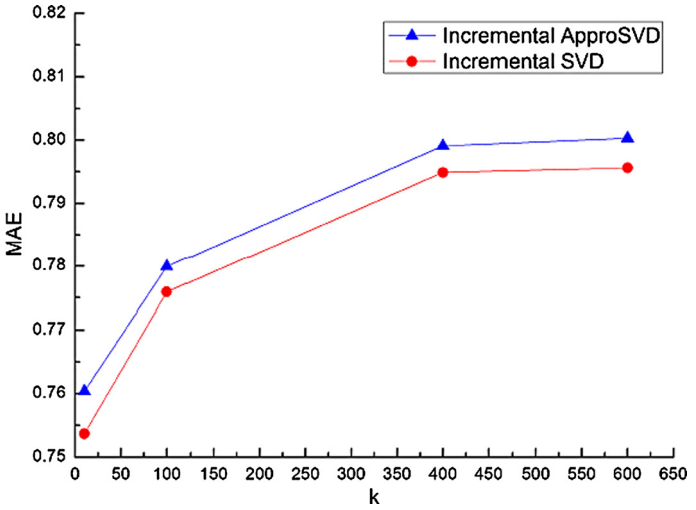

Fig. 9. MAE values for the MovieLens dataset with the change of k.

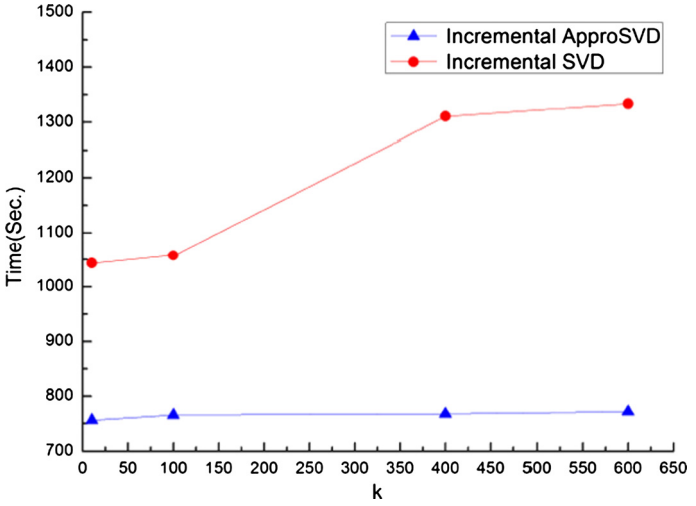

Fig. 10. Running time for the MovieLens dataset with the change of k.

Table 3
Performance comparisons for the Flixster dataset under the same k.

Incremental ApproSVD								Incremental SVD					
n_1	n_2	c_1	c_2	k	RMSE	MAE	Time (sec.)	n_1	n_2	k	RMSE	MAE	Time (sec.)
900	100	500	50	10	0.9321	0.7212	8.03	500	50	10	0.9209	0.7091	39.52
900	100	600	50	10	0.9281	0.7173	8.18	600	50	10	0.9140	0.7038	55.82
900	100	700	50	10	0.9213	0.7111	8.30	700	50	10	0.9144	0.7041	88.36
900	100	800	50	10	**0.9183**	**0.7084**	**8.54**	800	50	10	0.9143	0.7040	111.89

Table 4
Performance comparisons for the Flixster dataset under different k.

Incremental ApproSVD								Incremental SVD					
n_1	n_2	c_1	c_2	k	RMSE	MAE	Time (sec.)	n_1	n_2	k	RMSE	MAE	Time (sec.)
900	100	800	50	10	**0.9183**	**0.7084**	**8.54**	800	50	10	0.9143	0.7040	111.89
900	100	800	50	100	0.9651	0.7467	8.66	800	50	100	0.9799	0.7589	112.16
900	100	800	50	400	1.0055	0.7830	8.81	800	50	400	1.0306	0.8066	115.62
900	100	800	50	600	1.0089	0.7864	8.96	800	50	600	1.0333	0.8095	124.76

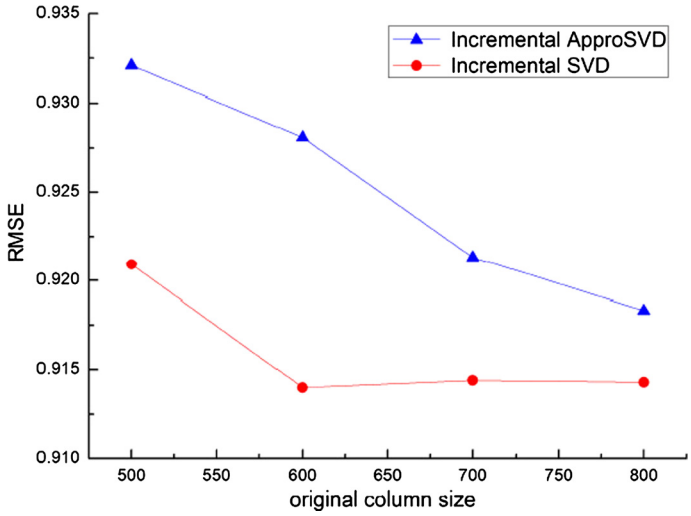

Fig. 11. RMSE values for the Flixster dataset with the change of original column size.

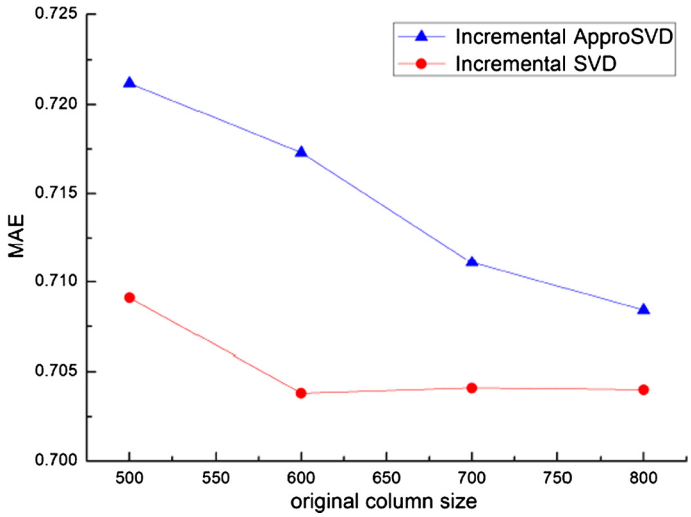

Fig. 12. MAE values for the Flixster dataset with the change of original column size.

and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 11 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

The two curves shown in Fig. 12 demonstrate the different trend of MAE values according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the Flixster dataset. Two parameters c_1 and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 12 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

The two curves shown in Fig. 13 demonstrate the different trend of running time (seconds) according to constant value $k = 10$, gradual increasing in c_1 for Incremental ApproSVD and n_1 for Incremental SVD for the Flixster dataset. Two parameters c_1 and n_1 denote the column size of the original matrix before performing the incremental algorithms. Therefore, in order to make Fig. 13 clearer, the horizontal axis is labeled as "original column size" which represents c_1 for Incremental ApproSVD and n_1 for Incremental SVD.

In Table 4, all columns represent the same meanings as described by Table 2. From Table 4, we can see that RMSE and MAE values increase with the increase of k for both Incremental ApproSVD and Incremental SVD. Moreover, the running time of Incremental ApproSVD has almost no change. However, the running time of Incremental SVD is increasingly sharp.

The two curves shown in Fig. 14 demonstrate the different trend of RMSE values according to a gradual increase in k for the Flixster dataset.

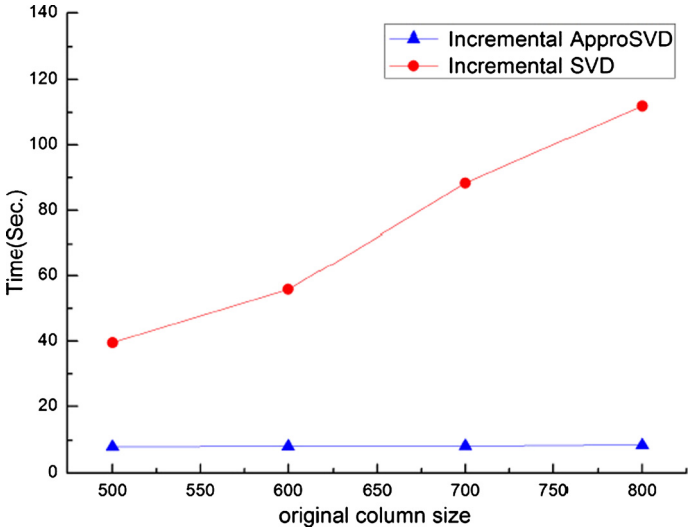

Fig. 13. Running time for the Flixster dataset with the change of original column size.

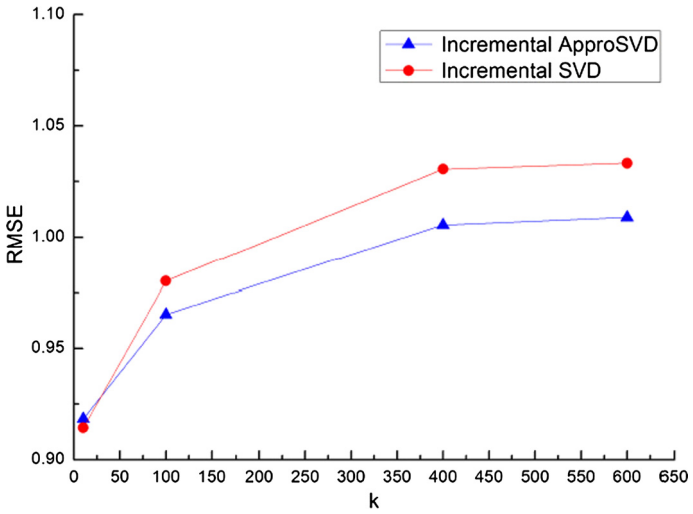

Fig. 14. RMSE values for the Flixster dataset with the change of original column size.

The two curves shown in Fig. 15 demonstrate the different trend of MAE values according to a gradual increase in k for the Flixster dataset.

The two curves shown in Fig. 16 demonstrate the different trend of running time according to a gradual increase in k for the Flixster dataset.

In short, for the same sized matrices and the same reduced dimension, although RMSE and MAE values of Incremental ApproSVD are a little bit higher than that of Incremental SVD in Table 3, and a little lower than that of Incremental SVD except for the case $k = 10$ in Table 4, the running time of Incremental ApproSVD is much shorter than that of Incremental SVD. Therefore, the overall performance of Incremental ApproSVD outperforms that of Incremental SVD for the Flixster dataset. When the sizes of matrices are growing bigger, the superiority of Incremental ApproSVD is much more obvious.

6. Conclusions

In this paper, we firstly describe an Incremental SVD algorithm which can exploit the previous singular values and singular vectors of the original matrix as an alternative to recomputing the SVD of the updated matrix. Secondly, we propose an incremental algorithm called Incremental ApproSVD, which is produced by combining the ApproSVD algorithm with the Incremental SVD algorithm. Thirdly, we give the mathematical analysis of the error between the actual ratings and the predicted ratings produced by the Incremental ApproSVD algorithm. Lastly, the evaluation results on the MovieLens 100 K dataset and Flixster dataset have demonstrated that the Incremental ApproSVD algorithm outperforms the Incremental SVD algorithm in terms of integrating the prediction accuracy and running time.

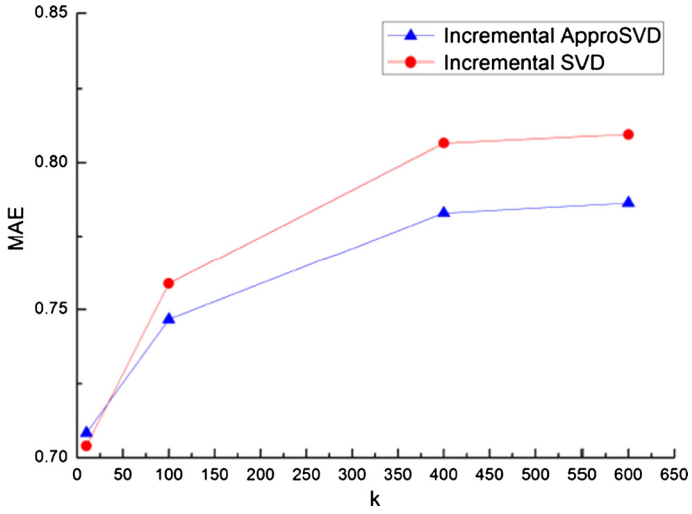

Fig. 15. MAE values for the Flixster dataset with the change of original column size.

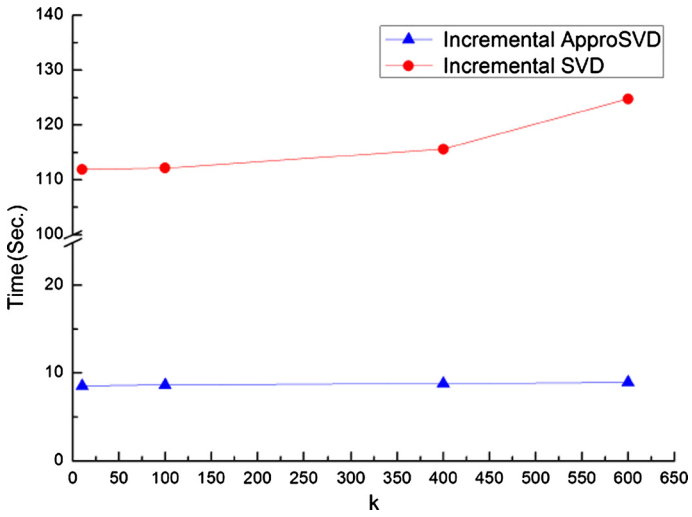

Fig. 16. Running time for the Flixster dataset with the change of original column size.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61272480 and 71072172).

References

[1] B. Marlin, Collaborative filtering: a machine learning perspective, Ph.D. thesis, University of Toronto, 2004.
[2] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, Z. Chen, Scalable collaborative filtering using cluster-based smoothing, in: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2005, pp. 114–121.
[3] G.N. Demir, A.S. Uyar, S.G. Ögüdücü, Graph-based sequence clustering through multiobjective evolutionary algorithms for web recommender systems, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, ACM, 2007, pp. 1943–1950.
[4] P.S. Chakraborty, A scalable collaborative filtering based recommender system using incremental clustering, in: Advance Computing Conference, IACC 2009, IEEE International, IEEE, 2009, pp. 1526–1529.
[5] S. Funk, Netflix update: try this at home, http://sifter.org/~simon/journal/20061211.html, 2006.
[6] S. Zhang, W. Wang, J. Ford, F. Makedon, Learning from incomplete ratings using non-negative matrix factorization, in: SDM, SIAM, 2006, pp. 549–553.
[7] Y. Cheng, G.M. Church, Biclustering of expression data, in: ISMB, vol. 8, 2000, pp. 93–103.
[8] G. Chen, F. Wang, C. Zhang, Collaborative filtering using orthogonal nonnegative matrix tri-factorization, Inf. Process. Manag. 45 (3) (2009) 368–379.
[9] H. Shan, A. Banerjee, Bayesian co-clustering, in: Eighth IEEE International Conference on Data Mining, ICDM'08, IEEE, 2008, pp. 530–539.
[10] X. Zhou, J. He, G. Huang, Y. Zhang, A personalized recommendation algorithm based on approximating the singular value decomposition (ApproSVD), in: Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, vol. 02, IEEE Computer Society, 2012, pp. 458–464.

[11] G.H. Golub, C.F. Van Loan, Matrix Computations, vol. 3, JHU Press, 2012.
[12] M.W. Berry, Large-scale sparse singular value computations, Int. J. Supercomput. Appl. 6 (1) (1992) 13–49.
[13] H. Nakayama, A. Hattori, Incremental learning and forgetting in RBF networks and SVMs with applications to financial problems, in: Knowledge-Based Intelligent Information and Engineering Systems, Springer, 2003, pp. 1109–1115.
[14] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Incremental singular value decomposition algorithms for highly scalable recommender systems, in: Fifth International Conference on Computer and Information Technology, Citeseer, 2002, pp. 27–28.
[15] C.G. Baker, K.A. Gallivan, P. Van Dooren, Low-rank incremental methods for computing dominant singular subspaces, Linear Algebra Appl. 436 (8) (2012) 2866–2888.
[16] H. Zha, H.D. Simon, On updating problems in latent semantic indexing, SIAM J. Sci. Comput. 21 (2) (1999) 782–791.
[17] C.-X. Ren, D.-Q. Dai, Incremental learning of bidirectional principal components for face recognition, Pattern Recognit. 43 (1) (2010) 318–330.
[18] Q. Zheng, X. Wang, W. Deng, J. Liu, X. Wu, Incremental projection vector machine: a one-stage learning algorithm for high-dimension large-sample dataset, in: AI 2010: Advances in Artificial Intelligence, Springer, 2011, pp. 132–141.
[19] M. Brand, Incremental singular value decomposition of uncertain data with missing values, in: Computer Vision ECCV 2002, Springer, 2002, pp. 707–720.
[20] S. Chandrasekaran, B. Manjunath, Y.-F. Wang, J. Winkeler, H. Zhang, An eigenspace update algorithm for image analysis, Graph. Models Image Process. 59 (5) (1997) 321–332.
[21] A. Levey, M. Lindenbaum, Sequential Karhunen–Loeve basis extraction and its application to images, IEEE Trans. Image Process. 9 (8) (2000) 1371–1374.
[22] G.I. Allen, L. Grosenick, J. Taylor, A generalized least-square matrix decomposition, J. Am. Stat. Assoc. 109 (505) (2014) 145–159.
[23] A. Dax, From eigenvalues to singular values: a review, Adv. Pure Math. 3 (2013) 8.
[24] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[25] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Application of dimensionality reduction in recommender system-a case study, in: Proceedings of the ACM Web KDD workshop on Web Mining for E-Commerce, ACM Press, New York, 2000, pp. 82–90.
[26] H. Polat, W. Du, SVD-based collaborative filtering with privacy, in: Proceedings of the 2005 ACM Symposium on Applied Computing, ACM, 2005, pp. 791–795.
[27] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, ACM, 2000, pp. 158–167.
[28] M.A. Ghazanfar, A. Prügel-Bennett, The advantage of careful imputation sources in sparse data-environment of recommender systems: generating improved SVD-based recommendations, Informatica (Slov.) 37 (1) (2013) 61–92.
[29] M.W. Berry, S.T. Dumais, G.W. O'Brien, Using linear algebra for intelligent information retrieval, SIAM Rev. 37 (4) (1995) 573–595.
[30] MovieLens, http://www.grouplens.org/node/73.
[31] Flixster dataset, http://www.cs.sfu.ca/~sja25/personal/datasets/.