Handling the Data Growth with Privacy Preservation in Collaborative Filtering

Xiwei Wang and Jun Zhang

Abstract The emergence of electric business facilitates people in purchasing merchandises over the Internet. To sell the products better, online service providers use recommender systems to provide recommendations to customers. Most recommender systems are based on collaborative filtering (CF) technique. This technique provides recommendations based on users' transaction history. Due to the technical limitations, many online merchants ask a third party to help develop and maintain recommender systems instead of doing that themselves. Therefore, they need to share their data with these third parties and users' private information is prone to leaking. Furthermore, the fast data growth should be handled by the data owner efficiently without sacrificing privacy. In this chapter, we propose a privacy preserving data updating scheme for collaborative filtering purpose and study its performance on two different datasets. The experimental results show that the proposed scheme does not degrade recommendation accuracy and can preserve a satisfactory level of privacy while updating the data efficiently.

Keywords Collaborative filtering · Data growth · Missing value imputation · Non-negative matrix factorization · Privacy preservation · Singular value decomposition · Updating

1 Introduction

A recommender system is a program that utilizes algorithms to predict users' purchase interests by profiling their shopping patterns. With the help of recommender systems, online merchants (also referred to as data owners) could better sell their

X. Wang (✉) · J. Zhang
Department of Computer Science, University of Kentucky, Lexington, KY40506-0633, USA
e-mail: xiwei@cs.uky.edu

J. Zhang
e-mail: jzhang@cs.uky.edu

G.-C. Yang et al. (eds.), *IAENG Transactions on Engineering Technologies*,
Lecture Notes in Electrical Engineering 229, DOI: 10.1007/978-94-007-6190-2_18,

products to the returning customers. Most recommender systems are based on collaborative filtering (CF) techniques, e.g., item/user correlation based CF [9], SVD (singular value decomposition) based latent factor CF [10]. Due to the technical limitations, many online merchants buy services from professional third parties to help build their recommender systems. In this scenario, merchants need to share their commercial data with the third party which has the potential for privacy leakage of user information. Typical private information in transaction data includes, but is not limited to, the ratings of a user left on particular items and items that this user has rated. People would not like others (except for the website where they purchased the products) to know this information. Thus privacy preserving collaborative filtering algorithms [4, 8] were proposed to tackle the problem. It is obvious that the data should be perturbed by online merchants before they release it to the third party.

Besides the privacy issue, data owner has to take care of the fast growing data as well. Once new data arrives, e.g., new items or new users' transaction records, it should be appended to the existing data. To protect privacy, data owner needs to do the perturbation. If he just simply redoes the perturbation on the whole data, he needs to resend the full perturbed data to the third party. This process not only takes a large amount of time to perturb data but also requires the model to be rebuilt on the site of the third party. It is infeasible to provide fast real-time recommendations.

In this chapter, we propose a privacy preserving data updating scheme in collaborative filtering. This scheme is based on truncated SVD updating algorithms [2, 7] which can provide privacy protection when incorporating new data into the original one in an efficient way. We start with the pre-computed SVD of the original data matrix. New rows/columns are then built into the existing factor matrices. We try to preserve users' privacy by truncating new matrix together with randomization and post-processing. Two missing value imputation methods during the updating process are studied as well. Results of the experiments that are conducted on MovieLens dataset [10] and Jester dataset [6] show that our scheme can handle data growth efficiently and keep a low level of privacy loss. The prediction quality is still at a good level compared with most published results.

The remainder of this chapter is organized as follows. Section 2 outlines the related work. Section 3 defines the problem and related notations. Section 4 describes the main idea of the proposed scheme. Section 5 presents the experiments on two datasets and discusses the results. Some concluding remarks and future work are given in Sect. 6.

2 Related Work

Most CF models suffer from the privacy leakage issue. Users' private information is fully accessible without any disguise to the provider of recommender systems. Canny [4] first proposed the privacy preserving collaborative filtering (PPCF) that deals with the privacy leakage in the CF process. In his distributed PPCF model, users could control all of their data. A community of users can compute a public

"aggregate" of their data that does not expose individual users' data. Each user then uses local computation to get personalized recommendations. Nevertheless, most popular collaborative filtering techniques are based on a central server. In this scheme, users send their data to a server and they do not participate in the CF process; only the server needs to conduct the CF. Polat and Du [8] adopted randomized perturbation for both correlation-based CF and SVD-based CF to provide privacy protection. They use uniform distribution and Gaussian distribution to generate random noise and apply the noise to the original data. Differing from Canny, Polat and Du, we focus on the framework in which data owner has all original user data but he needs to do perturbation before releasing it to a third party [15].

In this framework, data owner should be able to handle the fast data growth without leaking the privacy. Among all data perturbation methods, SVD is acknowledged as a simple and effective data perturbation technique. Stewart [12] surveyed the perturbation theory of singular value decomposition and its application in signal processing. A recent work by Tougas and Spiteri [13] demonstrated a partial SVD updating scheme that requires one QR factorization and one SVD (on small intermediate matrices and thus not expensive in computation) per update. Based on their work, Wang et al. [14] presented an improved SVD-based data value hiding method and tested it with clustering algorithm on both synthetic data sets and real data sets. Their experimental results indicate that the introduction of the incremental matrix decomposition produces a significant increase in speed for the SVD-based data value hiding model. Our scheme is similar to this model but we have modified the SVD updating algorithm with missing value imputation and post-processing so that it can be incorporated into collaborative filtering process smoothly.

3 Problem Description

Suppose the data owner has a sparse user-item rating matrix, denoted by $R \in \mathbb{R}^{m \times n}$, where there are m users and n item. When new users' transactions are available, the new rows, denoted by $T \in \mathbb{R}^{p \times n}$, should be appended to the original matrix R, as

$$\begin{bmatrix} R \\ T \end{bmatrix} \rightarrow R' \tag{1}$$

Similarly, when new items are collected, the new columns, denoted by $F \in \mathbb{R}^{m \times q}$, should be appended to the original matrix R, as

$$\begin{bmatrix} R & F \end{bmatrix} \rightarrow R'' \tag{2}$$

In both cases, data owner could not simply release T or F because they contain the real user ratings. He can not directly release R' or R'' either due to the scalability and privacy issues. An ideal case is, suppose a perturbed version (SVD-based) of

R with privacy protection has been released, data owner only releases the perturbed incremental data, say $\tilde{T}$ and $\tilde{F}$ which preserves users' privacy and does not degrade the recommendation quality.

4 Privacy Preserving Data Updating Scheme

In this section, we will present the data updating scheme that could preserve the privacy during the whole process. We try to protect users' privacy in three aspects, i.e., missing value imputation, randomization-based perturbation and SVD truncation. The imputation step can preserve the private information—"which items that a user has rated". Meanwhile, a second phase perturbation done by randomization and truncated SVD techniques solves another problem—"what are the actual ratings that a user left on particular items". On one hand, random noise can alter the rating values a bit while leaving the distribution unchanged. On the other hand, truncated SVD is a naturally ideal choice for data perturbation. It captures the latent properties of a matrix and removes the useless noise.

In (1), we see that T is added to R as a series of rows. The new matrix R' has a dimension of $(m + p) \times n$. Assuming the truncated rank-k SVD of R has been computed previously,

$$R_k = U_k \Sigma_k V_k^T \tag{3}$$

where $U_k \in \mathbb{R}^{m \times k}$ and $V_k \in \mathbb{R}^{n \times k}$ are two orthogonal matrices; $\Sigma_k \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the largest k singular values on its diagonal.

Since SVD cannot work on an incomplete matrix, we should impute the missing values in advance. We would like to use two different imputation methods: mean value imputation and WNMTF (weighted non-negative matrix tri-factorization) imputation.

In mean value imputation [10], we calculate each column mean and impute all the missing values in every column with its mean value.

In WNMTF [5, 16] imputation, we use WNMTF to factorize an incomplete matrix $A \in \mathbb{R}^{m \times n}$ into three factor matrices, i.e., $W \in \mathbb{R}^{m \times l}$, $G \in \mathbb{R}^{l \times t}$, and $H \in \mathbb{R}^{n \times t}$, where l and t are the column ranks of W and H. The objective function of WNMTF is

$$min_{W \geq 0, G \geq 0, H \geq 0} f(A, Y, W, G, H) = \|Y \circ (A - WGH^T)\|_F^2 \tag{4}$$

where $Y \in \mathbb{R}^{p \times n}$ is the weight matrix that indicates the value existence in the matrix A.

The update formula corresponds to (4) is

$$W_{ij} = W_{ij} \frac{[(Y \circ A)HG^T]_{ij}}{\{[Y \circ (WGH^T)]HG^T\}_{ij}} \tag{5}$$

$$G_{ij} = G_{ij} \frac{[W^T(Y \circ A)H]_{ij}}{\{W^T[Y \circ (WGH^T)]H\}_{ij}} \tag{6}$$

$$H_{ij} = H_{ij} \frac{[(Y \circ A)^T WG]_{ij}}{\{[Y \circ (WGH^T)]^T WG\}_{ij}} \tag{7}$$

With either of the above two imputation methods, we obtain the imputed matrices: $\hat{R}$ (with its factor matrices $\hat{U}_k$, $\hat{\Sigma}_k$ and $\hat{V}_k$) and $\hat{T}$. Now the problem space has been converted from (1) to (8):

$$\begin{bmatrix} \hat{R} \\ \hat{T} \end{bmatrix} \rightarrow \hat{R}' \tag{8}$$

After imputation, random noise that obeys Gaussian distribution is added to the new data $\hat{T}$, yielding $\dot{T}$. Then we follow the procedure in Tougas and Spiteri [13] to update the matrix. First, a QR factorization is performed on $\ddot{T} = (I_n - \hat{V}_k \hat{V}_k^T)\dot{T}^T$, where I_n is an $n \times n$ identity matrix. Thus we have $Q_T S_T = \ddot{T}$, in which $Q_T \in \mathbb{R}^{n \times p}$ is an orthonormal matrix and $S_T \in \mathbb{R}^{p \times p}$ is an upper triangular matrix. Then

$$\begin{aligned} \hat{R}' &= \begin{bmatrix} \hat{R} \\ \hat{T} \end{bmatrix} \approx \begin{bmatrix} \hat{R}_k \\ \hat{T} \end{bmatrix} \approx \begin{bmatrix} \hat{R}_k \\ \dot{T} \end{bmatrix} \\ &= \begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix} \begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix}^T \end{aligned} \tag{9}$$

Next, we compute the rank-k SVD on the middle matrix, i.e.,

$$\begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix}_{(k+p)\times(k+p)} \approx U'_k \Sigma'_k V'^T_k \tag{10}$$

Since $(k + p)$ is typically small, the computation of SVD should be very fast. Same as in Wang et al. [14], we compute the truncated rank-k SVD of $\hat{R}'$ instead of a complete one,

$$\hat{R}'_k = \left(\begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} U'_k \right) \Sigma'_k \left(\begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix} V'_k \right)^T \tag{11}$$

In CF context, the value of all entries should be in a valid range. For example, a valid value r in MovieLens should be $0 < r \leqslant 5$. Therefore, after obtaining the truncated new matrix $\hat{R}'_k$, a post-processing step is applied to it so that all invalid values will be replaced with reasonable ones. The processed version of $\hat{R}'_k$ is denoted by $\Delta\hat{R}'_k$.

In our scheme, we assume the third party owns $\hat{R}_k$ so we only send ΔT ($\Delta T = \Delta\hat{R}'_k(m + 1 : m + p, :) \in \mathbb{R}^{p \times n}$)[1] to them.

The following algorithm summarizes the SVD-based row/user updating.

[1] $\Delta\hat{R}'_k(m + 1 : m + p, :)$ is a Matlab notation that means the last p rows of $\Delta\hat{R}'_k$

Privacy Preserving Row Updating Algorithm {

INPUT:

Pre-computed rank-k SVD of $\hat{R}$: $\hat{U}_k$, $\hat{\Sigma}_k$ and $\hat{V}_k$;

New data $T \in \mathbb{R}^{p \times n}$;

Parameters for Gaussian random noise: μ and σ;

OUTPUT:

SVD for updated full matrix: $\hat{U}'_k$, Σ'_k and $\hat{V}'_k$;

Perturbed new data: ΔT;

Impute the missing values in $T \rightarrow \hat{T}$;

Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{T} \rightarrow \dot{T}$;

Perform QR factorization on $\ddot{T} = (I_n - \hat{V}_k \hat{V}_k^T)\dot{T}^T \rightarrow Q_T S_T$;

Perform SVD on $\ddot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix} \rightarrow \ddot{\Sigma} \approx U'_k \Sigma'_k V_k'^T$;

Compute $\left(\begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} \cdot U'_k \right) \rightarrow \hat{U}'_k$

Compute $\left(\begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix} \cdot V'_k \right) \rightarrow \hat{V}'_k$

Compute the rank-k approximation of $\hat{R}' \rightarrow \hat{R}'_k = \hat{U}'_k \Sigma'_k \hat{V}_k'^T$;

Process the invalid values $\rightarrow \Delta \hat{R}'_k$;

$\Delta \hat{R}'_k(m+1 : m+p, :) \rightarrow \Delta T$;

Return $\hat{U}'_k$, Σ'_k, $\hat{V}'_k$, *and* ΔT.

}

Like row updating, the column/item updating algorithm is presented as follows.

Privacy Preserving Column Updating Algorithm {

INPUT:

Pre-computed rank-k SVD of $\hat{R}$: $\hat{U}_k$, $\hat{\Sigma}_k$ and $\hat{V}_k$;

New data $F \in \mathbb{R}^{m \times q}$;

Parameters for Gaussian random noise: μ and σ;

OUTPUT:

SVD for updated full matrix: $\hat{U}''_k$, Σ''_k and $\hat{V}''_k$;

Perturbed new data: ΔF;

Impute the missing values in $F \rightarrow \hat{F}$;

Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{F} \rightarrow \dot{F}$;

Perform QR factorization on $\ddot{F} = (I_m - \hat{U}_k \hat{U}_k^T)\dot{F} \rightarrow Q_F S_F$;

Perform SVD on $\dot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & \hat{U}_k^T \dot{F} \\ 0 & R_F \end{bmatrix} \rightarrow \dot{\Sigma} \approx U''_k \Sigma''_k V_k''^T$;

Compute $\left(\begin{bmatrix} \hat{U}_k & Q_F \end{bmatrix} U''_k \right) \rightarrow \hat{U}''_k$

Compute $\left(\begin{bmatrix} \hat{V}_k & 0 \\ 0 & I_q \end{bmatrix} V''_k \right) \rightarrow \hat{V}''_k$

Compute the rank-k approximation of $\hat{R}'' \rightarrow \hat{R}''_k = \hat{U}''_k \Sigma''_k \hat{V}_k''^T$;

Process the invalid values $\rightarrow \Delta \hat{R}''_k$;

$\Delta \hat{R}''_k(:, n+1 : n+q) \rightarrow \Delta F$;

Return $\hat{U}''_k$, Σ''_k, $\hat{V}''_k$ and ΔF.

}

Data owner should keep the updated SVD of new user-item rating matrix ($\hat{U}'_k$, Σ'_k and $\hat{V}'_k$ for row updating, $\hat{U}''_k$, Σ''_k and $\hat{V}''_k$ for column updating) for future update and the perturbed new data matrix (ΔT for row updating, ΔF for column updating) as a reference.

5 Experimental Study

In this section, we discuss the test dataset, prediction model, evaluation strategy and experimental results.

5.1 Data Description

As most research papers in collaborative filtering, we adopt both 100 K MovieLens [10] and Jester [6] datasets as the test data. The 100 K MovieLens dataset has 943 users and 1,682 items. The 100,000 ratings, ranging from 1 to 5, were divided into two parts: the training set (80,000 ratings) and the test set (20,000 ratings). The Jester datasets has 24,983 users and 100 jokes with 1,810,455 ratings ranging from -10 to $+10$. In our experiment, we pick up 5,000 users together with their ratings and randomly select 80 % ratings as the training set while use the rest as test set.

5.2 Prediction Model and Error Measurement

In our experiments, we build the prediction model by SVD-based CF algorithm [10]. For a dense matrix A, its rank-k SVD is $A \approx \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^T$. Compute the user factor matrix($UF \in \mathbb{R}^{m \times k}$) and item factor matrix($IF \in \mathbb{R}^{n \times k}$):

$$UF = \tilde{U}_k \sqrt{\tilde{\Sigma}_k}, \quad IF = \tilde{V}_k \sqrt{\tilde{\Sigma}_k} \tag{12}$$

The predicted rating for user i left on item j is computed by taking the inner product of the ith row of UF and the jth row of IF:

$$p_{ij} = (\tilde{U}_k \sqrt{\tilde{\Sigma}_k})_i (\tilde{V}_k \sqrt{\tilde{\Sigma}_k})_j^T \tag{13}$$

When testing the prediction accuracy, we build the SVD model on training set; then for every rating in the test set, we compute the corresponding predicted value and measure the difference. We use mean absolute error (MAE) [3, 11] as the specific error metric (the lower the better).

5.3 Privacy Measurement

When we measure the privacy, we mean to what extent the original data could be estimated if given the perturbed data. In this chapter, we use the privacy measure that was first proposed by Agrawal and Aggarwal [1] and was applied to measure the privacy loss in collaborative filtering by Polat and Du [8].

In [1], they proposed $\Pi(Y) = 2^{h(Y)}$ as the privacy inherent in a random variable Y with $h(Y)$ as its differential entropy. Thus given a perturbed version of Y, denoted by X, the average conditional privacy(also referred to as Privacy Level) of Y given X is $\Pi(Y|X) = 2^{h(Y|X)}$.

Similar with Polat and Du's work, we take $\Pi(Y|X)$ as privacy measure to quantify the privacy in the experiments. Note that in this chapter, Y corresponds to all the existing values in the training set, meaning that we do not consider the missing values (treated as zeros in this case). This is slightly different from [15].

5.4 Evaluation Strategy

The proposed scheme is tested in several aspects: the prediction accuracy in recommendation, the privacy protection level, when to recompute SVD, and randomization degree with its impact in perturbation, etc.

To test when to recompute SVD, data in training set, which is viewed as a rating matrix, is split into two sub sections with a particular ratio ρ. The first ρ data is assumed to be held by the third party; the remaining data will be updated into it. For instance, when a row split is performed with $\rho = 40\,\%$, the first 40 % rows in training set is treated as R in (1). An imputation process should be done on this data without the knowledge from the remaining 60 % data, yielding $\hat{R}$ in (8). Then a rank-k SVD is computed for this matrix. We call the rank-k approximation of $\hat{R}$ the starting matrix. These data structures are utilized as the input of row updating algorithm. Results are expected to be different with varying split ratio in the training data. If the result is too far from the predefined threshold or the results starts to degrade at some point, a re-computation should be performed.

However, we don't update the remaining 60 % rows in training set in one round since data in real world application grows in small amount compared with the existing one. In our updating experiments, the 60 % rows are repetitively added to the starting matrix in several times, with 1/9 of the rows in each time [15].

We evaluate the algorithms on both MovieLens and Jester datasets by investigating the time cost of updating, prediction error and privacy measure on the final matrix. The machine we use is equipped with Intel® Core™ i5-2405S processor, 8GB RAM and is installed with UNIX operating system.

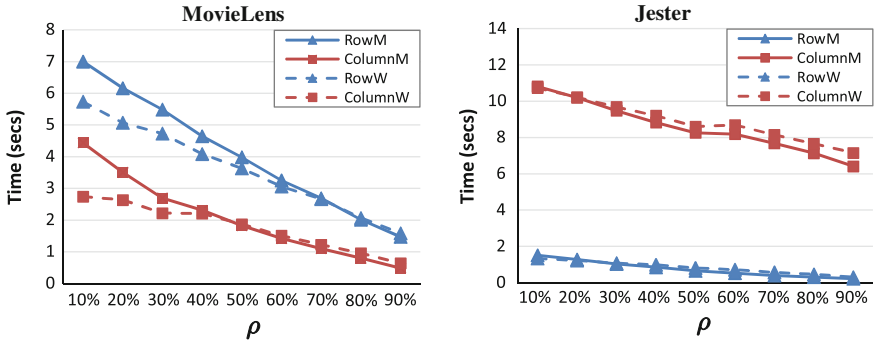

Fig. 1 Time cost variation with split ratio ρ

5.5 Results and Discussion

5.5.1 Split Ratio Study

Owing to the inherent property of SVD updating algorithms, errors are generated in each run. The data owner should be aware of the suitable time to recompute SVD for whole data so that the quality of the data can be kept. We study this problem by experimenting with the split ratio ρ. Note we use 13 and 11 as the truncation rank for MovieLens and Jester datasets. To be consistent, in WNMTF imputation, both l and t are set to 13 for MovieLens and 11 for Jester. As a reference, the maximum number of iterations in WNMTF is set to 10.

The time cost for updating new data with varying ρ is plotted in Fig. 1, where "RowM" and "ColumnM" represent row update and column update with mean value imputation while "RowW" and "ColumnW" represent the updates with WNMTF imputation. It is expected that updating fewer rows/columns takes less time.

Furthermore, the figure indicates the relation between the time cost of row and column updating—it depends on dimensionality of row and column. For example, the MovieLens data has more columns (1,682 items) than rows (943 users) while the Jester data has much fewer columns (100 items) than rows (24,983 users). We observed each step of both row and column updating algorithms and found that, when the number of columns is greater than the number of rows, steps 1 and 3 in row updating algorithm need more time than that in column updating algorithm due to higher dimensionality and vice versa. Compared with the time cost for imputing and computing SVD on the complete raw training set (i.e., the non-incremental case), our scheme runs much faster on both datasets (See Table 1). As for the imputation methods in this scenario, it is intuitive that WNMTF takes much shorter time than mean value imputation. Nevertheless, the difference is not that apparent in total time cost of row/column updating algorithms(i.e., the incremental case). This is because imputation time is a smaller part of the total time cost than the time for computing incremental SVD.

Table 1 Time cost of prediction on raw training data

Dataset	Imputation method	Imputation time	SVD time	Total time
MovieLens	Mean value	18.2617	0.3562	18.6179
	WNMTF	0.8764	0.3483	1.2247
Jester	Mean value	16.7078	0.1105	16.8183
	WNMTF	1.2392	0.1196	1.3588

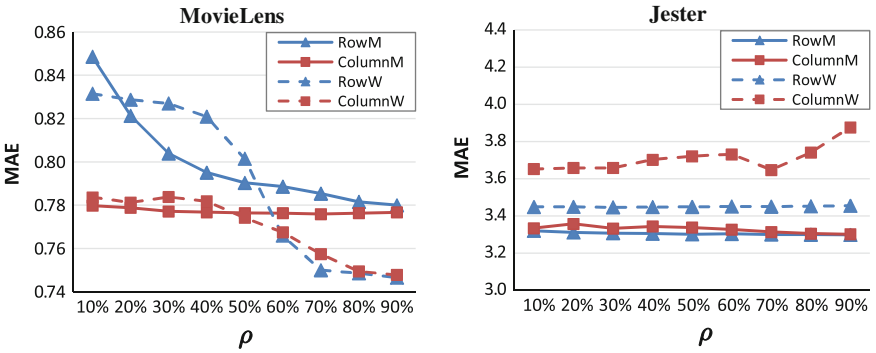

Fig. 2 MAE variation with split ratio ρ_1

Figure 2 shows the mean average error. For MovieLens dataset, the column update with mean value imputation almost stays at the same errors while the row update has a descending trend. With the WNMTF imputation, both updates produce smaller errors with rising split ratio. It is interesting that after some points, WNMTF provides lower prediction errors than mean value imputation. Thing is different for Jester data where MAE keeps stable in all updates and mean value imputation performs better. We get this difference because Jester has denser rating matrix than MovieLens meaning the former relies less on the new data than the latter.

The privacy level with varying split ratio is displayed in Fig. 3. As mentioned before, we only consider the existing ratings in the training set and eliminating the missing values when calculating the privacy level. This makes the results different from [15]. The privacy level without taking into account missing values does not change much with increasing split ratio. The curves look fluctuating because we reduced the interval of Y-axis. In this experiment, we notice that WNMTF imputation produces higher privacy level than mean value imputation. It can be attributed to more changes on ratings done by WNMTF. Since mean value imputation does not alter the existing values whereas WNMTF recompute the whole data, the latter perturbs the data to a deeper extent.

Now we can decide when to recompute SVD for the whole data according to Figs. 2 and 3. For mean value imputation, the MAEs on both datasets drop more slowly after $\rho \geq 50\,\%$ and there is no apparent variation of the slope for privacy measure curves, the re-computation can be performed when ρ reaches 50 %. For

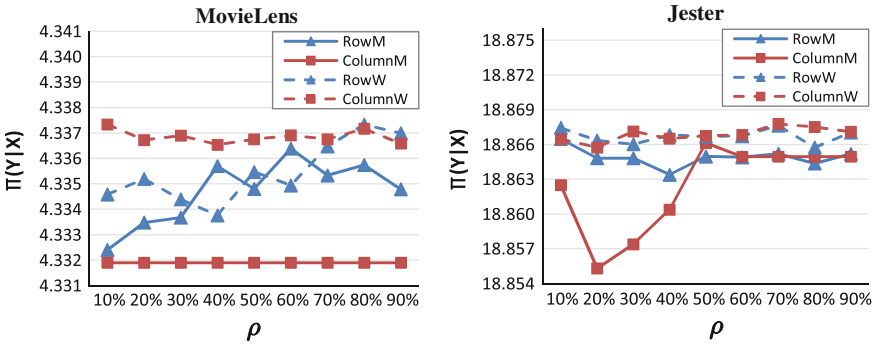

Fig. 3 Privacy level variation with split ratio ρ_1

WNMTF imputation, the MAEs keep decreasing all the way in MovieLens dataset, so the re-computation is not needed so far; the MAE for column update in Jester data increases when $\rho \geq 70\,\%$, meaning that the pre-computed matrices of SVD starts to degrade and thus a re-computation is helpful.

5.5.2 Role of Randomization in Data Updating

So far, we have not applied randomization technique to our data updating scheme. In this section, we study the role of randomization (Gaussian noise with μ and σ as its parameters in both row and column updating algorithms) in both data quality and privacy preservation. In the following experiments, ρ is fixed to $40\,\%$ and we use the WNMTF to impute the missing values. We probe μ in $\{0, 1\}$ and σ in $\{0.1, 1\}$ for both datasets. Table 2 collects the statistics of the test.

In this table, the results of row and column updating with randomization is compared with the non-randomized version. As can be seen, after applying random noise to new data before updating it, the privacy level in all cases improves to a certain extent. Nevertheless, we lost some utility of the data which results in greater MAEs at the same time. Hence, the parameters should be carefully chosen to deal with the

Table 2 Randomization in data updating on MovieLens dataset

Parameters		Row updating		Column updating	
μ	σ	MAE	$\Pi(Y\|X)$	MAE	$\Pi(Y\|X)$
0	**0**	**0.8209**	**4.3338**	**0.7818**	**4.3365**
0	0.1	0.8210	4.3341	0.7818	4.3368
0	1	0.8292	4.3355	0.7980	4.3390
1	0.1	1.0584	4.3342	1.0447	4.3372
1	1	1.0158	4.3361	0.9738	4.3389

trade-off between data utility and data privacy. Moreover, the results indicate that the expectation μ affects the results more than the standard deviation σ. We suggest data owner determine μ first and then tweak σ.

As a summary, randomization technique can be used as an auxiliary step in SVD-based data updating scheme to provide better privacy protection. It brings in randomness that perturbs the data before SVD updating. Therefore, data will be perturbed twice (randomization + SVD) in addition to imputation during the updating process and thus can achieve a higher privacy level. However, with the latent factors captured by SVD, most critical information can be retained which ensures the data quality for recommendation.

6 Conclusion and Future Work

In this chapter, we present a privacy preserving data updating scheme for collaborative filtering purpose. It is an incremental SVD based scheme with randomization technique and could be utilized in updating user-item matrix and preserving the privacy at the same time. We try to protect users' privacy in three aspects, i.e., missing value imputation, randomization-based perturbation and SVD truncation. The experimental results on MovieLens and Jester datasets show that our proposed scheme could update new data into the existing (processed) data very fast. It can also provide high quality data for accurate recommendation while keep the privacy.

Future work will take into account users' latent factor to obtain a more reasonable imputation strategy in updating the data. Other matrix factorization techniques, e.g., non-negative matrix factorization, will be considered to explore an alternative way of updating the new data with privacy preservation so that a possible better scheme can be provided.

References

1. Agrawal D, Aggarwal, CC (2001) On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, PODS '01, ACM, pp 247–255
2. Brand Matthew (2006) Fast low-rank modifications of the thin singular value decomposition. Linear Algebra Appl 415(1):20–30
3. Breese JS, Heckerman D, Kadie CM (1998) Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, Microsoft Corporation
4. Canny J (2002) Collaborative filtering with privacy. In: Proceedings of the 2002 IEEE symposium on security and privacy, IEEE Computer Society, pp 45–57
5. Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix trifactorizations for clustering. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, SIGKDD '06, ACM, pp 126–135
6. Goldberg K, Roeder T, Gupta D, Perkins C (2001) Eigentaste: a constant time collaborative filtering algorithm. Inf Retr 4(2):133–151

7. Koch O, Lubich C (2007) Dynamical low-rank approximation. SIAM J Matrix Anal Appl 29(2):434–454
8. Polat H, Du W (2005) Privacy-preserving collaborative filtering. Int J Electron Commer 9(4):9–35
9. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of ACM 1994 conference on computer supported cooperative work, ACM, pp 175–186
10. Sarwar BM, Karypis G, Konstan JA, Riedl JT (2000) Application of dimensionality reduction in recommender systems—a case study. In: Proceedings of ACM WebKDD workshop, ACM
11. Shardanand U, Maes P (1995) Social information filtering: algorithms for automating "word of mouth". In: Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '95, ACM Press/Addison-Wesley Publishing Co, pp 210–217
12. Stewart GW (1990) Perturbation theory for the singular value decomposition. Technical report, Computer Science Department, University of Maryland
13. Tougas J, Spiteri RJ (2007) Updating the partial singular value decomposition in latent semantic indexing. Comput Stat Data Anal 52:174–183
14. Wang J, Zhan J, Zhang J (2008) Towards real-time performance of data value hiding for frequent data updates. In: Proceedings of the IEEE international conference on granular computing, IEEE Computer Society, pp 606–611
15. Wang X, Zhang J (2012) SVD-based privacy preserving data updating in collaborative filtering. In: Lecture notes in engineering and computer science: proceedings of the world congress on engineering 2012, WCE 2012, IAENG, pp 377–284
16. Zhang S, Wang W, Ford J, Makedon F (2006) Learning from incomplete ratings using non-negative matrix factorization. In: Proceedings of the sixth SIAM international conference on data mining, SIAM, pp 548–552