SVD-based Privacy Preserving Data Updating in Collaborative Filtering

Xiwei Wang, Jun Zhang

***Abstract*—Collaborative Filtering technique is widely adopted by online service providers in their recommender systems. This technique provides recommendations based on users' transaction history. To provide decent recommendations, many online merchants (data owner) 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, with increasing transaction data, data owner should be able to handle data growth efficiently without sacrificing privacy.**

In this paper, 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.

***Index Terms*—data growth, SVD, updating, collaborative filtering, privacy.**

I. 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[1] could better sell their products to the users who have visited their websites before. Most recommender systems are based on collaborative filtering(CF for short) techniques, e.g., item/user correlation based CF[1], SVD (singular value decomposition) based latent factor CF[2]. All of them require user transaction history so that some model could be trained and used to provide recommendations. In order to provide decent recommendations, 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 what they are interested in and to what extent they like or dislike the items. Thus privacy preserving collaborative filtering algorithms[3], [4], [5] were proposed to tackle the problem. In a broader area, i.e., PPDM (Privacy-Preserving Data Mining), data perturbation techniques[6], [7] are applied to the data that will be published. To protect users' privacy, online merchants should process their data by perturbing the values in it with some strategy before they release it to the third party.

In addition to privacy issue, the fast growth of the collected data has also to be addressed as a problem to the data owner. Once there is new data arriving, e.g., new items or new users' transaction records, it should be appended to the existing data. To protect privacy, data owner needs to redo the perturbation which may result in a high computational cost. Moreover, if he redoes the perturbation on the whole data, he needs to resend the full perturbed data to the third party. This process also requires the model to be rebuilt on the site of the third party. It can be quite a big burden to provide fast real-time recommendations.

In this paper, we propose a privacy preserving data updating scheme in collaborative filtering. This scheme is based on truncated SVD updating algorithms[8], [9] and randomization techniques 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. We also take into account the imputation of missing values during the updating process to provide high quality data for accurate recommendations. Results of the experiments that are conducted on MovieLens dataset[2] and Jester dataset[10] 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 paper is organized as follows. Section II outlines the related work. Section III defines the problem and related notations. Section IV describes the main idea of the proposed scheme. Section V presents the experiments on two datasets and discusses the results. Some concluding remarks and future work are given in Section VI.

II. Related Work

SVD-based CF[2], [11] is one of the most successful latent factor CF models. It focuses on reducing dimensionality of the user-item rating matrix such that some "latent factors", which best explain user preferences, can be discovered. The basic idea of SVD-based models is to factorize the user-item rating matrix into two lower rank matrices, i.e., a "user factor" matrix UF and an "item factor" matrix IF. Thus, each user i and item j can be represented as a vector UF_i (the i-th row of UF) and IF_j (the j-th row of IF), respectively. The prediction of a rating from user i left on item j is made by taking the inner product of UF_i and IF_j.

A significant issue in most CF models is the privacy leakage. Users' private information is 100% accessible without any disguise to the provider of recommender systems.

Manuscript received March 02, 2012; revised April 16, 2012.

Xiwei Wang is with Department of Computer Science, University of Kentucky, Lexington, KY, 40506-0633 USA. e-mail: J.Wang@uky.edu

Jun Zhang is a Professor in the Department of Computer Science, University of Kentucky, Lexington, KY, 40506-0633 USA. e-mail: jzhang@cs.uky.edu

[1]The term "online merchant" and "data owner" will be used interchangeably as they refer to the same thing in this context.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Canny[3] firstly 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[4] 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. They then perform the CF algorithms on the perturbed data to obtain the predictions. 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 on it before releasing it to a third party.

In this framework, data owner also needs to take care of the fast data growth and should make sure the privacy protection is still kept at a reasonable level after data updating. Among all data perturbation methods, SVD is acknowledged as a feasible 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 Brand[8] demonstrated a fast low-rank modifications of the thin singular value decomposition. This algorithm can update an SVD with new rows/columns and compute its low rank approximation very efficiently. Tougas and Spiteri[13] proposed 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, better scalability, and better real-time performance of the model. Our scheme is similar to this model but we have modified the SVD updating algorithm plus randomization and post-processing so that it can be incorporated into SVD-based CF smoothly.

III. Problem Description

In this paper, we discuss the privacy issue of data updating for collaborative filtering purpose. The data is referred to as the user-item rating matrix (it is typically very sparse), denoted by R. Suppose we have m users and n items, then R has a dimension of $m \times n$, i.e., $R \in \mathbb{R}^{m \times n}$. 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

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

Nonetheless, data owner could not simply release T or F because they contain the real user ratings which are users' private information. He can not directly release R' or R'' either due to both the scalability and privacy issue. 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.

IV. Privacy Preserving Data Updating Scheme

In this section, we will present the data updating scheme in collaborative filtering 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". However, since pure imputation will typically generate same values and fill the empty entries with these values, the matrix is vulnerable to attack. This also raises another kind of private information – "what are the actual ratings that a user left on particular items". In this scenario, randomization and truncated SVD techniques are used to do a second phase perturbation that solves the problem. On one hand, random noise can alter the rating values a bit while leaving the distribution unchanged. This is very crucial in data mining applications because data utility must be guaranteed when doing perturbation. On the other hand, truncated SVD is a naturally ideal choice in data perturbation. It captures the latent properties of a matrix and eliminates the useless noise. If given a well-chosen truncation rank, SVD can provide a good balance between data privacy and its utility.

As stated in the previous section, new data could be treated as new rows or columns in the matrix. They should be appended to the original matrix, i.e., R and further perturbed to protect users' privacy. Hence, we will discuss our scheme by row updating and column updating separately.

A. Row/User Updating

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,

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

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.

As mentioned in Section III, the user-item rating matrix is a sparse matrix so before we do SVD on it, we should first impute the missing values. Similar with [2], we use the column mean, i.e., the mean rating value of each item, to fill the empty entries. These mean values are held in a vector $\vec{r}_{mean} = (\bar{r}_1, \cdots, \bar{r}_n)$ and will be used to help update SVD.

For new rows T, before incorporating it into the existing matrix, an imputation step is performed. In this step, the

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

empty entries should be filled with values that have knowledge from both the mean values of existing matrix and the ratings in new data. We use (4) to calculate new column mean.

$$\bar{r}'_j = \frac{m \times \bar{r}_j + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} r_{ij}}{m + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} 1} \tag{4}$$

Note that the new column mean does not affect the old matrix, which should be kept unchanged as the third parties hold the perturbed old matrix and data owner only releases the perturbed new data.

Thus, 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 (5):

$$\begin{bmatrix} \hat{R} \\ \hat{T} \end{bmatrix} \rightarrow \hat{R}' \tag{5}$$

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 [13] to update the matrix. First, a QR factorization is performed on $\ddot{T} = (I_n - \hat{V}_k \cdot \hat{V}_k^T) \cdot \dot{T}^T$, where I_n is an $n \times n$ identity matrix. Thus we have $Q_T \cdot S_T = \ddot{T}$, 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

$$\begin{aligned} \hat{R}' &= \begin{bmatrix} \hat{R} \\ \hat{T} \end{bmatrix} \approx \begin{bmatrix} \hat{R}_k \\ \hat{T} \end{bmatrix} \approx \begin{bmatrix} \hat{R}_k \\ \dot{T} \end{bmatrix} \\ &= \begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix} \begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix}^T \end{aligned} \tag{6}$$

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

$$\begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix}_{(k+p)\times(k+p)} \approx U'_k \cdot \Sigma'_k \cdot V_k'^T \tag{7}$$

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

$$\hat{R}'_k = \left(\begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} \cdot U'_k \right) \cdot \Sigma'_k \cdot \left(\begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix} \cdot V'_k \right)^T \tag{8}$$

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.

$$\Delta\hat{r}'_{k,ij} = \begin{cases} validMinValue & if \;\; \hat{r}'_{k,ij} < validMinValue \\ validMaxValue & if \;\; \hat{r}'_{k,ij} > validMaxValue \\ \hat{r}'_{k,ij} & otherwise \end{cases} \tag{9}$$

In (9), $\hat{r}'_{k,ij}$ is the (i,j)-th entry of $\hat{R}'_k$. $validMinValue$ and $validMaxValue$ depend on particular dataset. For MovieLens dataset, $validMinValue$ = 0 and $validMaxValue$ = 5; for Jester dataset, $validMinValue$ = -10 and $validMaxValue$ = 10. Eventually, the perturbed and updated user-item rating matrix, $\Delta\hat{R}'_k \in \mathbb{R}^{(m+p)\times n}$ with $\Delta\hat{r}'_{k,ij}$ as its entries, is generated.

In our scheme, we assume the third party owns $\hat{R}_k$ so we only send ΔT ($\Delta T = \Delta\hat{R}'_k(m+1:m+p,:) \in \mathbb{R}^{p\times n}$)[2] to them.

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

Algorithm 1 summarizes the SVD-based row/user updating.

Algorithm 1 Privacy Preserving Row Updating

Require:
Pre-computed rank-k SVD of $\hat{R}$: $\hat{U}_k, \hat{\Sigma}_k$ and $\hat{V}_k$;
Item mean of $\hat{R}$: $\vec{r}_{mean}$;
New data $T \in \mathbb{R}^{p\times n}$;

Ensure:
SVD for updated full matrix: $\hat{U}'_k, \Sigma'_k$ and $\hat{V}'_k$;
Perturbed new data: ΔT;
Updated item mean vector: $\vec{r}'_{mean}$;

1: Impute the missing values in T with formula (4) and update item mean vector $\rightarrow \hat{T}, \vec{r}'_{mean}$;
2: Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{T} \rightarrow \dot{T}$;
3: Perform QR factorization on $\ddot{T} = (I_n - \hat{V}_k \cdot \hat{V}_k^T) \cdot \dot{T}^T \rightarrow Q_T \cdot S_T$;
4: Perform SVD on $\ddot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix} \rightarrow \ddot{\Sigma} \approx U'_k \cdot \Sigma'_k \cdot V_k'^T$;
5: Compute $\left(\begin{bmatrix} \hat{U}_k & 0 \\ 0 & I_p \end{bmatrix} \cdot U'_k \right) \rightarrow \hat{U}'_k$
Compute $\left(\begin{bmatrix} \hat{V}_k & Q_T \end{bmatrix} \cdot V'_k \right) \rightarrow \hat{V}'_k$
6: Compute the rank-k approximation of $\hat{R}' \rightarrow \hat{R}'_k = \hat{U}'_k \cdot \Sigma'_k \cdot \hat{V}_k'^T$;
7: Process the invalid values by formula (9) $\rightarrow \Delta\hat{R}'_k$;
8: $\Delta\hat{R}'_k(m+1:m+p,:) \rightarrow \Delta T$;
9: Return $\hat{U}'_k, \Sigma'_k, \hat{V}'_k, \Delta T$ and $\vec{r}'_{mean}$.

B. Column/Item Updating

Column updating is similar with row updating, whereas there are several differences between them. Note that we use item mean to impute the missing values in raw user-item rating matrix. In row updating, the mean values will change when new rows/users are added while in column updating, the mean values only depend on the new columns/items. With this property, it is not necessary to keep an item mean vector in column updating.

Like (5), we have the following task:

$$\begin{bmatrix} \hat{R} & \hat{F} \end{bmatrix} \rightarrow \hat{R}'' \tag{10}$$

Algorithm 2 depicts the SVD-based column/item updating.

Data owner should keep the updated SVD for new user-item rating matrix ($\hat{U}'_k, \Sigma'_k$ and $\hat{V}'_k$ for row updating, $\hat{U}''_k, \Sigma''_k$ and $\hat{V}''_k$ for column updating) and the perturbed new data matrix (ΔT for row updating, ΔF for column updating). Moreover, the updated item mean $\vec{r}'_{mean}$ is also supposed to be held by data owner if a row updating has been performed.

As shown in both algorithms, three perturbation techniques are combined together to preserve users' privacy. Imputation at the beginning removes all the missing values. Adding random noise to the imputed data makes values different from each other. Truncated SVD updating eliminates the factors that are not so important to the data. This process keeps the data utility and protects the data privacy at the same time. Three techniques make contributions to privacy preservation in different aspects. We will study the effect of the proposed schemes in next section.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Algorithm 2 Privacy Preserving Column Updating

Require:
Pre-computed rank-k SVD of $\hat{R}$: $\hat{U}_k, \hat{\Sigma}_k$ and $\hat{V}_k$;
New data $F \in \mathbb{R}^{m \times q}$;

Ensure:
SVD for updated full matrix: $\hat{U}_k'', \Sigma_k''$ and $\hat{V}_k''$;
Perturbed new data: ΔF;

1: Impute the missing values in F with corresponding item mean values $\rightarrow \hat{F}$;
2: Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{F} \rightarrow \dot{F}$;
3: Perform QR factorization on $\ddot{F} = (I_m - \hat{U}_k \cdot \hat{U}_k^T) \cdot \dot{F} \rightarrow Q_F \cdot S_F$;
4: Perform SVD on $\dot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & \hat{U}_k^T \cdot \dot{F} \\ 0 & R_F \end{bmatrix} \rightarrow \dot{\Sigma} \approx U_k'' \cdot \Sigma_k'' \cdot V_k''^T$;
5: Compute $\left(\begin{bmatrix} \hat{U}_k & Q_F \end{bmatrix} \cdot U_k'' \right) \rightarrow \hat{U}_k''$
Compute $\left(\begin{bmatrix} \hat{V}_k & 0 \\ 0 & I_q \end{bmatrix} \cdot V_k'' \right) \rightarrow \hat{V}_k''$
6: Compute the rank-k approximation of $\hat{R}'' \rightarrow \hat{R}_k'' = \hat{U}_k'' \cdot \Sigma_k'' \cdot \hat{V}_k''^T$;
7: Process the invalid values like formula (9) ($\hat{r}'_{k,ij}$ are now entries in $\hat{R}_k''$) $\rightarrow \Delta \hat{R}_k''$;
8: $\Delta \hat{R}_k''(:, n+1 : n+q) \rightarrow \Delta F$;
9: Return $\hat{U}_k'', \Sigma_k'', \hat{V}_k''$ and ΔF.

V. Experimental Study

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

A. Data Description

As most research papers in collaborative filtering, we adopt both MovieLens[2] and Jester[10] datasets as the test data. The public MovieLens dataset has 3 subsets, i.e., 100K(100,000 ratings), 1M(1,000,000 ratings) and 10M(10,000,000 ratings). We use the first subset to test the algorithms. It 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). If we store both sets in matrices, they are expected to be very sparse(with the sparsity of 93.7% in the training matrix).

The Jester datasets are from a web-based joke recommendation system, which is developed by the University of California, Berkeley[10]. There are also 3 subsets, namely jester-data-1, jester-data-2 and jester-data-3. We take the first set for test. It has 24,983 users and 100 jokes with 1,810,455 ratings ranging from -10 to +10. We randomly select 80% ratings as the training set and use the rest as test set. Compared with MovieLens, the Jester dataset is not so sparse (with the sparsity of 27.5% in the training matrix).

B. Prediction Model and Error Measurement

In our experiments, we build the prediction model by SVD-based CF algorithm[2]. Since SVD cannot deal with missing values, they are treated as zeros if we don't do any pre-processing. A typical way to impute missing values is using item mean. For each column (corresponds to an item, whereas a row corresponds to a user), the mean value is calculated from existing ratings and all the missing values in this column are filled by the mean value.

For a dense matrix A, its rank-k SVD is $A \approx \tilde{U}_k \cdot \tilde{\Sigma}_k \cdot \tilde{V}_k^T$. Compute the user factor matrix($UF \in \mathbb{R}^{m \times k}$) and item factor matrix($IF \in \mathbb{R}^{n \times k}$):

$$UF = \tilde{U}_k \cdot \sqrt{\tilde{\Sigma}_k}, \quad IF = \tilde{V}_k \cdot \sqrt{\tilde{\Sigma}_k} \tag{11}$$

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

$$p'_{ij} = (\tilde{U}_k \cdot \sqrt{\tilde{\Sigma}_k})_i \cdot (\tilde{V}_k \cdot \sqrt{\tilde{\Sigma}_k})_j^T \tag{12}$$

To ensure the predicted rating is in the valid range, the same boundary check like (9) is applied:

$$p_{ij} = \begin{cases} validMinValue & if \;\; p'_{ij} < validMinValue \\ validMaxValue & if \;\; p'_{ij} > validMaxValue \\ p'_{ij} & otherwise \end{cases} \tag{13}$$

When testing the prediction accuracy, we first use the training set to obtain user factor matrix UF and item factor matrix IF; then for every rating in the test set, we compute the corresponding predicted value and measure the difference. We do this for all the ratings in test set and thus the MAE(mean absolute error)[15], [16] is calculated:

$$MAE = \frac{1}{|TestSet|} \sum_{r_{ij} \in TestSet} |r_{ij} - p_{ij}| \tag{14}$$

In this paper, MAE is the measurement criterion of prediction accuracy. The lower the value is, the higher the accuracy we get.

C. 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 paper, the privacy measure based on differential entropy is adopted. This privacy measure was firstly proposed by Agrawal and Aggarwal[17] and was applied to measure the privacy loss in collaborative filtering by Polat and Du[4].

In [17], 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 $\Pi(Y|X) = 2^{h(Y|X)}$. They also proposed the conditional privacy loss of Y after revealing X as:

$$\mathcal{P}(Y|X) = 1 - \Pi(Y|X)/\Pi(Y) = 1 - 2^{h(Y|X)}/2^{h(Y)} \tag{15}$$

Similar with Polat and Du's work, we take $\Pi(Y|X)$ and $\mathcal{P}(Y|X)$ as privacy measure to quantify the privacy in the experiments.

D. Evaluation Strategy

We would like to test our scheme in several aspects: the prediction accuracy in recommendation, the privacy protection level, how to split the new data in updating, when to recompute SVD, and randomization degree with its effect 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

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a particular ratio ρ_1. The first ρ_1 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 $\rho_1 = 40\%$, 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 (5). Then a rank-k SVD and the item mean vector are computed on this matrix. We call the rank-k approximation of $\hat{R}$ the starting matrix. These data structures are utilized as the input of ***Algorithm 1***. 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 change much faster 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 experiment, the 60% rows are repetitively added to the starting matrix in several times, depending on another split ratio, say ρ_2, of the new data. For example, if $\rho_2 = 1/10$, the new data will be added to the starting matrix in 10 rounds. The final matrix (starting matrix + $\Delta T_1 + \cdots + \Delta T_{10}$) is the perturbed and updated matrix.

We evaluate the algorithms on both MovieLens and Jester datasets by testing the time cost of updating, prediction error and privacy measure on the final matrix.

E. Results and Discussion

1) Truncation Rank(k) in SVD: Due to the characteristic of SVD-based CF, the rank of the truncated matrix, i.e., k must be chosen in advance. Most papers report that $k = 13$ is an optimal choice for MovieLens dataset and $k = 11$ for Jester dataset. We verified this by probing k in {2, 5, ..., 25, 50, 100} and computing the corresponding MAE numbers[2]. The results on MovieLens are shown in ***Figure 1***. Note that this experiment has nothing to do with updating since we performed the SVD-based prediction on the full imputed training data.

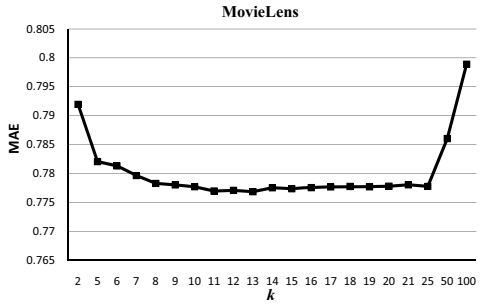

Fig. 1. MAE variation with different rank-k

The curve shows the mean absolute errors with different k's and the lowest MAE(0.7769) is reached when $k = 13$. Same experiment on Jester also confirms that the lowest MAE(3.2871) is reached when $k = 11$. Accordingly, we used 13 and 11 as the truncation rank for MovieLens and Jester datasets, respectively, in the following experiments.

2) Split Ratio ρ_2: We start from the split ratio(ρ_2) of new data. We fix ρ_1 at 40% meaning the first 40% data in training set is treated as starting matrix, while the remaining 60% will be added to it. ρ_2 is set to 1/10, 1/9, 1/8, ...,1/2, 1. The greater ρ_2 is, the fewer rounds are needed in updating.

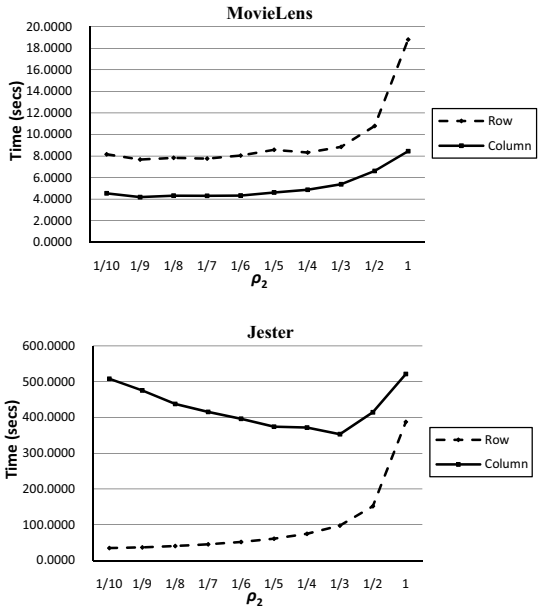

Fig. 2. Time cost variation with split ratio ρ_2

Figure 2 illustrates the time cost with different split ratio ρ_2. We use "Row" to represent the row updating and "Column" to refer to column updating. Note that we set $\mu = 0$ and $\sigma = 0$ to eliminate randomization effect in both algorithms. We will report the results with randomization in Section V-E4.

The curves of MovieLens data are generally in ascending trend with rising split ratio and the row updating takes longer time than column updating. In Jester data, the column updating reaches the shortest time when $\rho_2 = 1/3$ and the row updating takes less time than column updating. It is clear that except for column updating in Jester data, updating new data in more rounds with less data in each round can decrease the time cost. However, the split ratio cannot simply be determined only based on this factor. The prediction accuracy and privacy protection level should play even more crucial role during this process.

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 ***Algorithm 1*** need more time than that in ***Algorithm 2*** due to higher dimensionality and vice versa. Nevertheless, compared with the time cost for imputing and computing SVD on raw training set(we run all the experiments on the same machine), which is 44.9744(imputation) + 2.2866(SVD) = 47.261s for MovieLens and 1592.8075(imputation) + 3.7552(SVD) = 1596.5627s for Jester, our scheme run much faster in both row and column updating.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

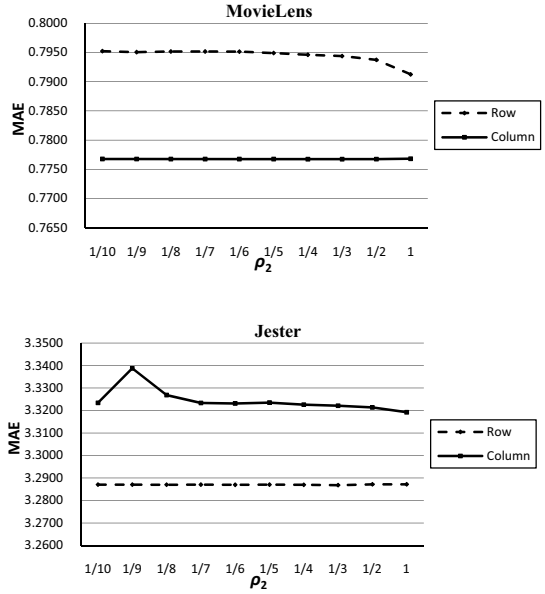

Fig. 3. MAE variation with split ratio ρ_2

The mean average error charted in ***Figure 3*** keeps stable with different split ratio ρ_2 which implies the quality of the updated data with respect to prediction accuracy is not affected so much by ρ_2. We also got similar results on privacy measure, see ***Figure 4***.

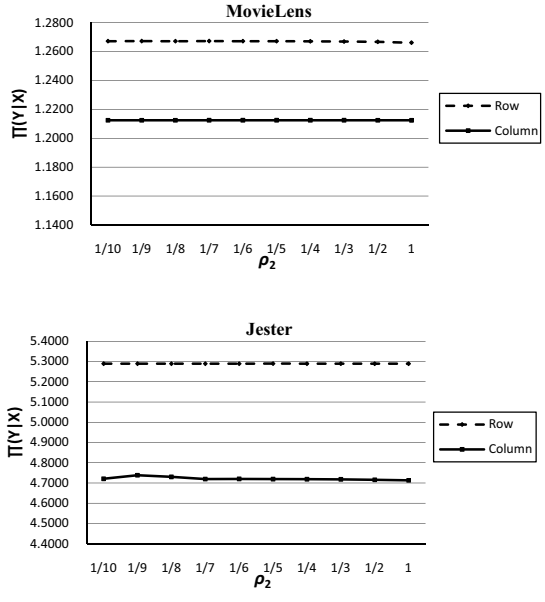

Fig. 4. Privacy level variation with split ratio ρ_2

Based on the experimental results of split ratio ρ_2 study, we use $\rho_2 = 1/9$ in both row and column updating for MovieLens data in the following experiments. The Jester data will work with $\rho_2 = 1/10$ in row updating and $\rho_2 = 1/3$ in column updating.

3) Split Ratio ρ_1: 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_1.

The time cost for updating new data with varying ρ_1 is plotted in ***Figure 5***. It is expected that updating fewer rows/columns takes less time. While different types of split ratio were tested, the relation between time cost of row and column updating did not change.

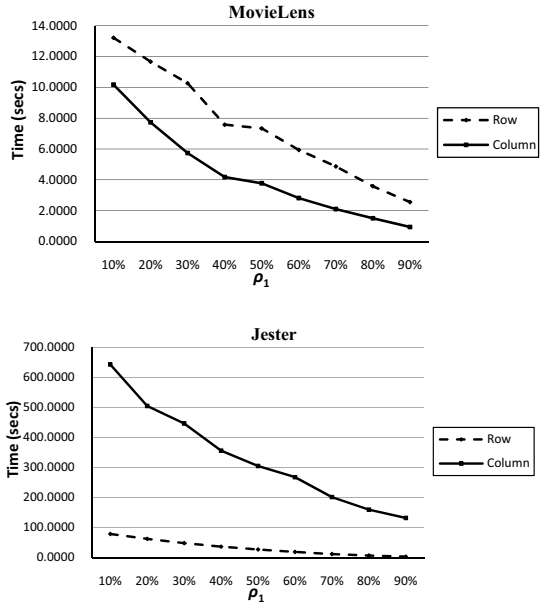

Fig. 5. Time cost variation with split ratio ρ_1

Figure 6 shows the mean average error. The curve in MovieLens data has a descending trend in row updating but keeps at a stable level in column updating. The case is different for Jester data where MAE is basically decreasing for column updating and keep stable for row updating with rising split ratio ρ_1. It indicates that with fewer ratings in the starting matrix, the prediction model depreciates in profiling users' preferences and thus leads to a lower prediction accuracy. As for the question "what affects MAE more, row(user) or column(item)?", it also depends on the dimensions of row and column. Since the total amount of information stored in a rating matrix is fixed, suppose every matrix entry contributes the same amount of information as others, the fewer users(or items) we have, the more information each user(or item) contributes. In MovieLens data, the row dimension is lower than the column dimension. In this scenario, users play more important role than item because there are fewer users than items and each user contributes more than each item does. Therefore, with the increasing number of users, MAE drops. On the other hand, in Jester data, row has a higher dimension than item so items are more critical and have greater effect on errors. Compared with the MAE of unperturbed training matrix, which are 0.7769(MovieLens) and 3.2871(Jester), our updating schemes get 0.7951(row updating), 0.7768(column updating) for MovieLens data and 3.2870(row updating), 3.3221(column updating) for Jester data when $\rho = 40\%$, respectively. The MAEs are still as good as the published

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

results. If using better prediction model, we believe the MAE could be lower.

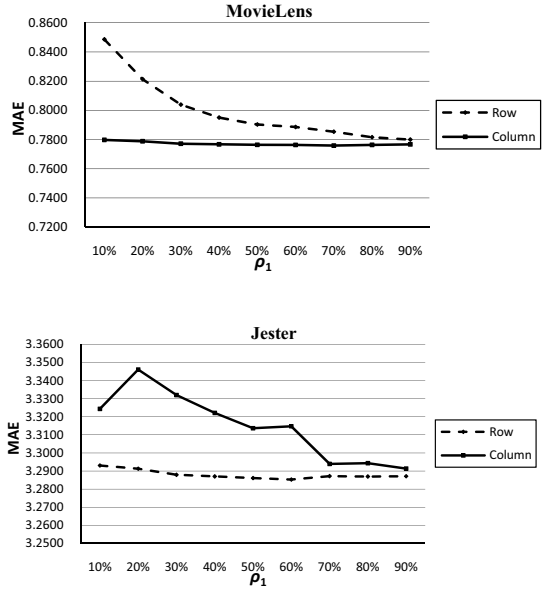

Fig. 6. MAE variation with split ratio ρ_1

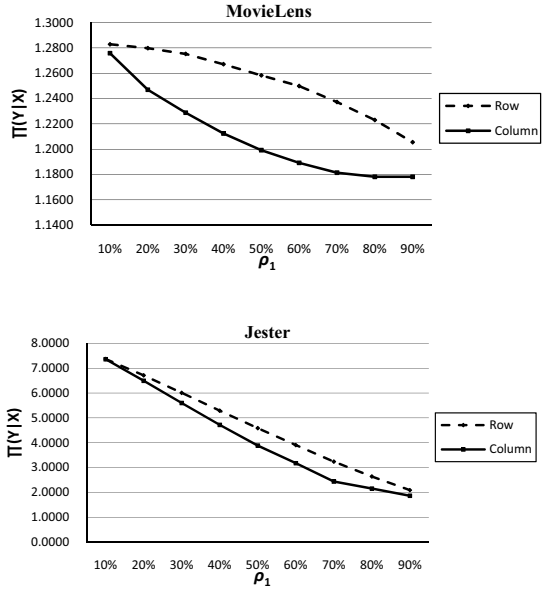

Fig. 7. Privacy level variation with split ratio ρ_1

The privacy level with varying split ratio is displayed in ***Figure 7***. It is clear that privacy level decreases with more data held in the starting matrix. As explained in the previous paragraphs, more data, especially more users in MovieLens or items in Jester, will make greater contribution to model construction that leaks more privacy. In this experiment, the privacy levels on both datasets are higher and change faster in row updating than that in column updating. The results imply that privacy with respect to users(rows) plays a dominant role in the whole process other than item privacy. This does make sense because when we talk about the privacy, we mean users' privacy not item.

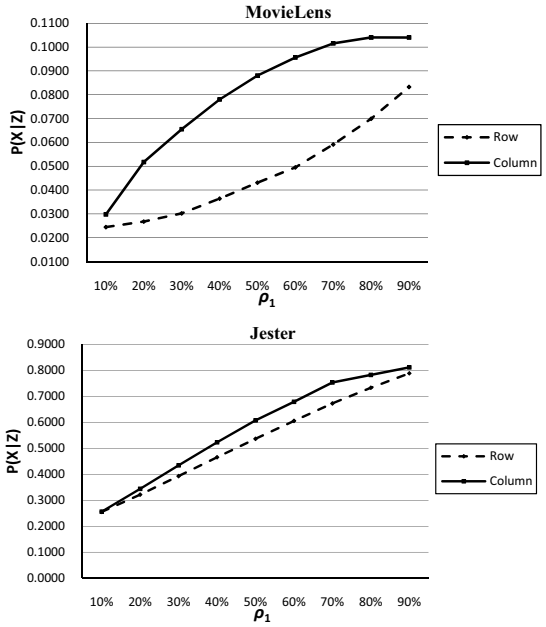

Fig. 8. Privacy loss variation with split ratio ρ_1

Corresponding to privacy level, the privacy loss of raw training data (Y) after revealing the perturbed and updated data (X) is presented in ***Figure 8***.

With growing split ratio, privacy loss increases where privacy level decreases. The curve looks like the upside-down version in ***Figure 7*** due to the relation between them, see (15).

Now we can decide when to recompute SVD for the whole data according to ***Figure 6*** and ***Figure 7***. Since MAEs on both datasets drop more slowly after $\rho_1 \geq 50\%$ and there is no apparent variation of the slope for privacy measure curves, the re-computation can be performed when ρ_1 reaches 50%.

4) 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 ***Algorithms 1 and 2***) in both data quality and privacy preservation. In the following experiments, ρ_1 is fixed to 40% and ρ_2 is set to 1/9. We probe μ in $\{0, 1\}$ and σ in $\{0.1, 1\}$ for both datasets. ***Table I*** collects the statistics of the test.

In this table, 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, both privacy metrics ($\Pi(Y|X)$ and $\mathcal{P}(Y|X)$) in all cases improve 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 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 decide μ first and then tweak σ.

As a summary, randomization technique can be used as

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
RANDOMIZATION IN DATA UPDATING

		MovieLens Data						Jester Data					
		Row Updating			Column Updating			Row Updating			Column Updating		
μ	σ	MAE	$\Pi(Y\|X)$	$\mathcal{P}(Y\|X)$	MAE	$\Pi(Y\|X)$	$\mathcal{P}(Y\|X)$	MAE	$\Pi(Y\|X)$	$\mathcal{P}(Y\|X)$	MAE	$\Pi(Y\|X)$	$\mathcal{P}(Y\|X)$
0	0	0.7951	1.2671	0.0364	0.7768	1.2124	0.0780	3.2870	5.2894	0.4656	3.3221	4.7178	0.5233
0	0.1	0.7955	1.2792	0.0272	0.7781	1.2558	0.0450	3.2872	5.4717	0.4472	3.3390	4.9811	0.4967
0	1	0.8331	1.2927	0.0169	0.8219	1.2869	0.0214	3.3007	6.4436	0.3490	3.3542	6.1920	0.3744
1	0.1	1.0764	1.2808	0.0260	0.9837	1.2583	0.0431	3.3221	5.4706	0.4473	3.3629	5.0179	0.4930
1	1	1.0421	1.2926	0.0170	0.9258	1.2839	0.0236	3.3358	6.4410	0.3492	3.3799	6.2577	0.3678

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.

VI. CONCLUSION AND FUTURE WORK

Collaborative filtering is a powerful technique in recommender systems that recommend products to customers. However, many online merchants do not build such systems by themselves. Instead, they buy services from a third party to help them do it. Protecting users' privacy is one of the primary concerns in data sharing. Furthermore, data is growing in every second. How to update the new data into the existing one efficiently with privacy preservation is an inevitable issue in this case. To our best knowledge, there's little work focused on it.

In this paper, we present a privacy preserving data updating scheme for collaborative filtering purpose. It is an SVD-updating based scheme with randomization technique and could be utilized in updating incremental 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 mining underlying data pattern so that a possible better scheme can be provided.

REFERENCES

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*. ACM, 1994, pp. 175–186.

[2] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender systems – a case study," in *Proceedings of ACM WebKDD Workshop*. ACM, 2000.

[3] J. Canny, "Collaborative filtering with privacy," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2002, pp. 45–57.

[4] H. Polat and W. Du, "Privacy-preserving collaborative filtering," *International Journal of Electronic Commerce*, vol. 9, no. 4, pp. 9–35, 2005.

[5] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the netflix prize contenders," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. ACM, 2009, pp. 627–636.

[6] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. ACM, 2000, pp. 439–450.

[7] S. Xu, J. Zhang, D. Han, and J. Wang, "Singular value decomposition based data distortion strategy for privacy protection," *Knowledge and Information Systems*, vol. 10, pp. 383–397, 2006.

[8] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Applications*, vol. 415, no. 1, pp. 20–30, May 2006.

[9] O. Koch and C. Lubich, "Dynamical low-rank approximation," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 434–454, 2007.

[10] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, July 2001.

[11] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD Cup and Workshop*, ser. KDDCup '07. ACM, 2007, pp. 39–42.

[12] G. Stewart, "Perturbation theory for the singular value decomposition," Computer Science Department, University of Maryland, Tech. Rep., 1990.

[13] J. Tougas and R. J. Spiteri, "Updating the partial singular value decomposition in latent semantic indexing," *Computational Statistics & Data Analysis*, vol. 52, pp. 174–183, 2007.

[14] J. Wang, J. Zhan, and J. Zhang, "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, 2008, pp. 606–611.

[15] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Microsoft Research, Microsoft Corporation, Tech. Rep., 1998.

[16] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '95. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[17] D. Agrawal and C. C. Aggarwal, "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*, ser. PODS '01. ACM, 2001, pp. 247–255.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)