2015

Data Privacy Preservation in Collaborative Filtering Based Recommender Systems

Xiwei Wang
University of Kentucky, xwsunrise@gmail.com

Recommended Citation

Wang, Xiwei, "Data Privacy Preservation in Collaborative Filtering Based Recommender Systems" (2015). *Theses and Dissertations--Computer Science*. 35.
https://uknowledge.uky.edu/cs_etds/35

Xiwei Wang, Student

Dr. Jun Zhang, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

DATA PRIVACY PRESERVATION IN COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEMS

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By

Xiwei Wang

Lexington, Kentucky

Director: Jun Zhang, Ph.D., Professor of Computer Science

Lexington, Kentucky

2015

ABSTRACT OF DISSERTATION

DATA PRIVACY PRESERVATION IN COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEMS

This dissertation studies data privacy preservation in collaborative filtering based recommender systems and proposes several collaborative filtering models that aim at preserving user privacy from different perspectives.

The empirical study on multiple classical recommendation algorithms presents the basic idea of the models and explores their performance on real world datasets. The algorithms that are investigated in this study include a popularity based model, an item similarity based model, a singular value decomposition based model, and a bipartite graph model. Top-N recommendations are evaluated to examine the prediction accuracy.

It is apparent that with more customers' preference data, recommender systems can better profile customers' shopping patterns which in turn produces product recommendations with higher accuracy. The precautions should be taken to address the privacy issues that arise during data sharing between two vendors. Study shows that matrix factorization techniques are ideal choices for data privacy preservation by their nature. In this dissertation, singular value decomposition (SVD) and nonnegative matrix factorization (NMF) are adopted as the fundamental techniques for collaborative filtering to make privacy-preserving recommendations. The proposed SVD based model utilizes missing value imputation, randomization technique, and the truncated SVD to perturb the raw rating data. The NMF based models, namely iAux-NMF and iCluster-NMF, take into account the auxiliary information of users and items to help missing value imputation and privacy preservation. Additionally, these models support efficient incremental data update as well.

A good number of online vendors allow people to leave their feedback on products. It is considered as users' public preferences. However, due to the connections between users' public and private preferences, if a recommender system fails to distinguish real customers from attackers, the private preferences of real customers can be exposed. This dissertation addresses an attack model in which an attacker holds real customers' partial ratings and tries to obtain their private preferences by cheating recommender systems. To resolve this problem, trustworthiness information is incorporated into NMF based collaborative filtering techniques to detect the attackers and make reasonably different recommendations to the normal users and the attackers. By doing so, users' private preferences can be effectively protected.

KEYWORDS: collaborative filtering, data update, matrix factorization, privacy, trustworthiness

Xiwei Wang

May 7, 2015

DATA PRIVACY PRESERVATION IN COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEMS

By

Xiwei Wang

Jun Zhang, Ph.D.

Director of Dissertation

Miroslaw Truszczynski, Ph.D.

Director of Graduate Studies

May 7, 2015

Date

ACKNOWLEDGEMENTS

Upon finishing my dissertation, I would like to express my gratitude to people who encouraged, assisted, inspired, and cared about me during my PhD life at the University of Kentucky. Without the guidance from my advisor and other committee members, help from friends, and support from my family, I would never have been able to finish my dissertation.

First of all, I would like to express my sincere appreciation to my academic advisor, Dr. Jun Zhang, for his excellent guidance, patience, encouragement, and caring. It is Dr. Zhang who led me to study the topics in privacy preserving collaborative filtering and trained me to become a researcher in this field. Because of his broad knowledge and keen professional insight, I could conduct the research in a very effective way. This allowed me to achieve promising academic success in the past six years. Dr. Zhang makes me feel at home in the United States and I am truly honored to work and study under his supervision.

Special thanks to Dr. Jinze Liu for her advice and financial support in my second academic year. Dr. Liu inspired me to read papers and start my research in the area of online recommender systems. Without her help, I would never have been able to continue my degree and begin such an interesting and meaningful research topic.

Next, I would like to thank the other faculty members of my Advisory Committee: Dr. Ruigang Yang (Department of Computer Science) and Dr. Sen-ching Cheung (Department of Electrical and Computer Engineering) for their helpful comments on my dissertation.

Then, I would like to thank my research collaborators: Dr. Yin Wang, Lawrence Technological University, for providing me with the opportunity of being a workshop committee member and reviewer, as well as the suggestions on writing academic papers; Dr. Nirmal Thapa, TIBCO Software Inc., for his innovative ideas and discussions during the first few years of my study; Mr. Kiho Lim, University of Kentucky, for his ideas on privacy-preserving vehicle communications.

Also, I would like to thank Dr. Jun Zhang, Dr. William Brent Seales, and Dr. Neil Moore for their strong recommendation letters, advice, and encouragements in my job search.

I extend my gratitude to the Elbert C. Ray eStudio for their excellent writing service. The two experts, Mr. James Marinelli and Mr. Clinton Woodson, were very friendly, easygoing, and professional. Their expertise in English writing has benefited me to a great extent. The help they provided me includes not only polishing my dissertation, but also improving my abilities in writing and speaking English.

Thanks also go to all the members in the Laboratory for High Performance Scientific Computing & Computer Simulation and the Laboratory for Computational Medical Imaging & Data Analysis during my study: Dr. Yin Wang, Dr. Xuwei Liang, Dr. Changjiang Zhang, Dr. Dianwei Han, Dr. Ning Cao, Dr. Nirmal Thapa, Dr. Lian Liu, Dr. Ruxin Dai, Dr. Pengpeng Lin, and Mr. Qi Zhuang. I want to thank them all for their helpful suggestions and creating a friendly working environment.

Last but not least, I would like to express my thanks and love to my parents. I thank them for giving me my life as well as their continuous support, encouragement, and endless love.

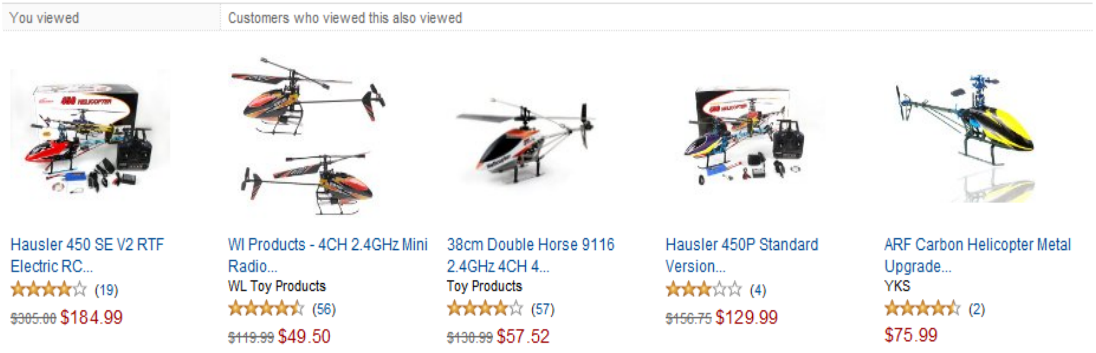
Table of Contents

List of Tables

List of Figures

1 Introduction

As technology develops, life becomes easier in many aspects. One of them is the emergence of electronic commerce, which not only helps sellers save resources and time but also facilitates customers in choosing and buying merchandise. Different kinds of promotions have been adopted by merchants to advertise their products. Conventional stores like Walmart and Sam's Club present popular products, e.g., batteries, gift cards, and magazines at the checkout line in addition to offering discounts. This is a typical way of product recommendations. Same as conventional stores, online shops provide recommendations to their customers as well. However, for returning customers, online stores are superior to the conventional ones with respect to product recommendations. This is due to the fact that the former have users'[1] purchase history on file which is very helpful in recommending merchandise. Online shopping websites often use recommender systems to do this task.



You viewed

Hausler 450 SE V2 RTF Electric RC...
★★★★☆ (19)
~~$305.00~~ $184.99

Customers who viewed this also viewed

WI Products - 4CH 2.4GHz Mini Radio...
WL Toy Products
★★★★☆ (56)
~~$119.99~~ $49.50

38cm Double Horse 9116 2.4GHz 4CH 4...
Toy Products
★★★★☆ (57)
~~$130.99~~ $57.52

Hausler 450P Standard Version...
★★★☆☆ (4)
~~$156.75~~ $129.99

ARF Carbon Helicopter Metal Upgrade...
YKS
★★★★☆ (2)
$75.99

Figure 1.1: Product recommendation on Amazon.com

A recommender system is a program that utilizes algorithms to predict users' purchase preferences by profiling their shopping patterns. There are many research publications about recommender systems since the mid-1990s [2]. Various approaches

[1] The terms "customer" and "user"are used interchangeably as they refer to the same thing in this context. This interchangeability also applies to "product" and "item".

and models have been proposed and applied to real world applications. Most recommender systems are based on collaborative filtering (CF) techniques [23, 40], e.g., item/user correlation based CF's [62], singular value decomposition (SVD) based latent factor CF's [64], and nonnegative matrix factorization (NMF) based CF's [13, 87].

With CF, previous transactions are analyzed in order to establish connections between users and products. When recommending items to a user, the CF based recommender systems try to find information related to this user to compute ratings for every possible item. Items with the highest rating scores will be presented to the user.

In many online recommender systems, it is inevitable for data owners to expose their data to other parties. For instance, due to the lack of easy-to-use technology, some online merchants buy services from professional recommendation service providers to help build their recommender systems. In addition, many shops share their real time data with business associates for better product recommendations. Such examples include two or more online bookstores that sell similar books, and online movie rental websites that have similar movies in their systems. In these scenarios, exposed data can cause privacy leakage of user information if no pre-processing is done. Typical private information includes the ratings that a user has given to particular items and on which items that this user has rated, or in general, user's preferences. People would not like others (except the website where they purchased the products from) to know what they are interested in and to what extent they like or dislike the items. This is the most fundamental privacy problem in collaborative filtering. Thus privacy-preserving collaborative filtering algorithms [12, 59, 53] were proposed to resolve the problem. However, different from datasets for general data mining tasks, the rating matrices in collaborative filtering are typically very incomplete, meaning that there are a large number of missing values. Accordingly, data owners should complete two tasks before releasing the data to a third party: imputing missing ratings and perturbing the whole data.

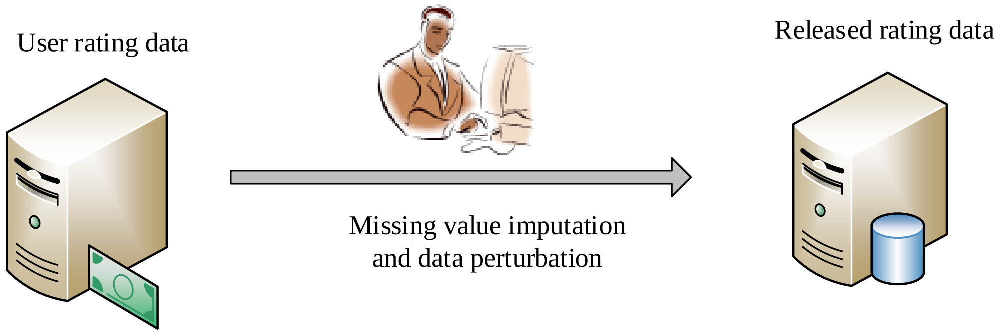


Figure 1.2: Missing value imputation and data perturbation

In addition, there are a few other problems and points with the CF based recommender systems that would be studied in this dissertation:

(1) Managing fast data growth. In the collaborative filtering context, data may grow in two aspects: the new item arrival and the new user arrival. It requires the data owners to complete the aforementioned two tasks on the new data in a timely manner. In other words, every time when the new data arrives, data owners only need to perform some incremental data update process on it and send the imputed and perturbed new data to the third parties.

(2) Utilizing auxiliary information. In some datasets, e.g., the MovieLens dataset [64], the Sushi preference dataset [35], and the LibimSeTi dating agency dataset [11], auxiliary information of users or items, e.g., users' demographic data and items' category data, are also provided. This information, if properly used, can improve the recommendation accuracy, especially when the original rating matrix is extremely incomplete.

(3) Distinguishing between real users and attackers with the use of social networks. It is known to people that on many online shopping websites, customers can leave feedback on the products they purchased. This is treated as users' public preferences. Due to the connections between people's public preferences and private preferences, if a recommender system fails to distinguish the real customers from the attackers, it would be highly possible that the attackers can obtain users' private preferences by cheating the system. Trustworthiness information in social networks can be used to help identify attackers for privacy preservation purposes.

1.1 Dissertation Organization

First of all, a preliminary empirical study on several collaborative filtering algorithms is presented in Chapter 2. Browsing history datasets from an American retargeting company are used as the test datasets in this study to verify their performance in binary rating data. Chapter 3 describes an SVD based privacy-preserving data update scheme in collaborative filtering and discusses the experimental results on the MovieLens dataset [64] and the Jester dataset [22]. An improved data update approach, which adopts NMF as its fundamental technique, is proposed in Chapter 4. Chapter 5 addresses the rank determination issue that arises from the NMF based approach and shows a solution to this issue. Chapter 6 proposes an attack model in online recommender systems and presents a trust-aware privacy-preserving recommender system that neutralizes the attack. The future work and concluding remarks are discussed in 7.

1.2 Related Work

Collaborative filtering techniques have been extensively studied by many researchers. In [42], Yehuda divided the collaborative filtering techniques into two subareas: the neighborhood approaches and the latent factor models.

The neighborhood approaches focus on the relationships between either users or items. There are user based models [25] and item based models [18] in this scheme. For user based models, the recommendation algorithms compute the similarity degrees between users in order to obtain the like-minded users, i.e., the "neighbors", of the active users[2] who will receive recommendations. Items that have been purchased by neighbors will be recommended to this user. Thus the key point of user based neighborhood models is to find the "neighbors". However, the computational cost of these types of models grows fast with the increasing number of users and items. With millions of users and items, it will take a substantial amount of time to compute the user similarities [18]. One way to solve this problem is to build the recommendation models based on the items. The basic principle is very close to the user based models. It attempts to compute the item-item similarity matrix and the system will recommend items which are similar to the ones that have been purchased by the same user in the past. Since the number of items is much smaller than the number of users in most cases, the item based models are more scalable than the user based ones. Papagelis et al. [55] also showed that the former models resulted in better performance in prediction accuracy compared to the latter ones.

Different from the neighborhood approaches, the latent factor models transform users and items to the same latent factor spaces. The characteristics of users and items are "extracted" and represented at the latent level when factorizing the user-item rating matrix. The singular value decomposition (SVD) [57], the principal component analysis (PCA) [22], and the nonnegative matrix factorization (NMF) [13, 87] are three typical techniques for the latent factor models. They attempt to reduce the dimensionality of the rating matrices and utilize the reduced matrices. By doing so, noise can be decreased and some trivial factors are eliminated such that the system only retains the information that is essential to the recommendations.

However, a significant issue in most CF models is the privacy leakage. Canny [12] first proposed the privacy-preserving collaborative filtering (PPCF) that addresses this issue in the CF process. In his PPCF model, users could control all of their data.

[2] In many papers, the term "active users" and "test users" are used interchangeably. They both represent the users whose preferences will be predicted.

Users in a community are able to compute a public “aggregate” of their data, in which no individual user’s data is exposed. Local computation is performed by each user to get the personalized recommendations. Besides Canny’s model, the PPCF has been studied in both distributed systems [12, 84, 6, 83, 48, 65] and centralized systems [60, 47, 34, 14]. While most peer-to-peer (P2P) environments adopt distributed recommender systems, centralized systems are widely used by almost all of the most popular online vendors, e.g., eBay, Amazon, and Newegg. In centralized systems, 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 [59, 60] applied randomized perturbation techniques to the SVD based collaborative filtering to provide privacy-preserving recommendations. In their method, uniform or Gaussian noise is added to the users’ real ratings and then the server predicts the unknown ratings by the perturbed data.

In this framework, the data owner also needs to manage the fast data growth and should ensure that privacy protection is still kept at a reasonable level after the data update. Among all data perturbation methods, SVD is acknowledged as a feasible and effective data perturbation technique. Stewart [67] surveyed the perturbation theory of singular value decomposition and its application in signal processing. Brand [9] demonstrated a fast low-rank modification 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 [72] proposed a partial SVD update scheme that requires one QR factorization and one SVD in each update. Since both factorizations are performed on small intermediate matrices, the computation cost is not expensive. Based on their work, Wang et al. [75] presented an improved SVD based data value hiding method and tested it with clustering algorithms on both synthetic datasets and real datasets. Their experimental results indicate that, by introducing the incremental matrix decomposition, the efficiency of the SVD based data value hiding model is significantly increased. It also provides better scalability and better real-time performance of the model. The scheme proposed in Chapter 3 is similar to this model but it modifies the SVD update algorithm and comes with randomization and post-processing techniques so it can be incorporated into the SVD based CF smoothly.

In addition to SVD, NMF has also been studied in collaborative filtering. Zhang et al. [87] applied NMF to collaborative filtering to learn the missing values in the rating matrix. They treated NMF as a solution to the expectation maximization (EM) problems. Chen et al. [13] proposed an orthogonal nonnegative matrix tri-factorization (ONMTF) [20] based collaborative filtering algorithm. Their algorithm also takes into account the user similarity and item similarity. To study how differently NMF would perform from SVD, based on Polat's model [60], Li et al. [47] used NMF instead of SVD as the fundamental collaborative filtering technique and they obtained better results than Polat's method. While both methods were demonstrated to perturb the data to a reasonable level and keep the prediction precision, it is not clear how much contribution can be made by the methods to the real world recommender systems. Different from Polat's and Li's work, Kaleli et al. [34] proposed a privacy-preserving naive Bayesian classifier (PPNBC) CF approach. The approach employs randomized response techniques (RRT) [76] to protect users' privacy while producing referrals by a naive Bayesian classifier (NBC). In their scheme, the RRT is applied to both the one-group scheme and the multi-group scheme for data distortion purposes. The distorted data is then fed to NBC for recommendations. Adding to [34], Bilge et al. [8] utilized pre-processing to improve the privacy-preserving recommendations. In their method, the masked data is pre-processed by identifying the most similar items to each item off-line; some of the unrated items' entries are also filled to improve the density. Since the increasing numbers of features and groups would degrade the online performance of the multi-group PPNBC significantly, decreasing the amount of data involved in CF can be very beneficial. With the varying number of neighbors, the central server can decide how many items would be recommended to the users.

As mentioned before, auxiliary information of users and items is helpful if properly utilized. The fast data update approach proposed in Chapter 4 first converts this information to the cluster membership indicator matrices which are then considered as constraints for updating factor matrices. Nirmal et al. [71] proposed explicit incorporation of the additional constraint, called the “clustering constraint”, into NMF in order to suppress the data patterns in the process of performing the matrix factorization. Their work is based on the idea that one of the factor matrices in NMF contains cluster membership indicators. The clustering constraint is another indicator matrix with altered class membership in it. This constraint then guides NMF in updating factor matrices. Based on this idea, the proposed model applies the user and item cluster membership indicators to nonnegative matrix tri-factorization (NMTF), which results in better imputation of the missing values.

With regard to the clustering algorithms, K-Means [51] is a popular and well studied approach that is easy to implement and is widely used in many domains. As the name of the algorithm indicates, K-Means needs the definition of “mean” prior to clustering. It minimizes a cost function by calculating the means of clusters. This makes K-Means most suitable for continuous numerical data. When given categorical data such as users’ demographic data and movies’ genre information, K-Means needs a pre-processing phase to make the data suitable for clustering. Huang [27] proposed a K-Modes clustering algorithm to extend the K-Means paradigm to the categorical domains. Their algorithm introduces new dissimilarity measures to handle categorical objects and replaces means of clusters with modes. Additionally, a frequency based method is used to update modes in the clustering process so that the clustering cost function is minimized. In 2005, Huang et al. [28] further applied a new dissimilarity measure to the K-Modes clustering algorithm to improve its clustering accuracy.

The fast data growth requires the clustering algorithms to update the clusters constantly. The number of clusters might be increased or decreased. Su et al. [68] proposed a fast incremental clustering algorithm by changing the radius threshold value dynamically. Their algorithm restricts the number of the final clusters and reads the original dataset only once. It also considers the frequency information of the attribute values in the inter-cluster dissimilarity measure. The approach proposed in Chapter 5 adopts their clustering algorithm with some modifications. It is known that the NMF based collaborative filtering algorithms need to determine the dimensions of the factor matrices and update them when necessary. It is not convenient for people to manually specify these values and the automated decision making is highly desired. To this purpose, the proposed method determines the number of clusters by an incremental clustering algorithm and uses them as the dimensions in NMF.

Recent work on using trustworthiness in collaborative filtering indicates that this information benefits prediction precision as well as privacy protection. It reveals the trust relationships between users and can be obtained from online social networks. Jamali et al. [31] proposed SocialMF, a recommender system that makes use of the matrix factorization technique with trust propagation in social networks. Similar to [31], [80] proposed TrustMF to handle data sparsity and cold start problems which happen commonly in collaborative filtering based recommender systems. In TrustMF, users are projected into low-dimensional latent feature spaces by the matrix factorization technique according to their relationships. By doing so, users' mutual influence on their own opinions is reflected in a more reasonable way. In [33], the authors presented a trust based recommendation scheme on vertically distributed data for privacy preservation. The scheme builds a trust web of users and then uses it to filter users' neighbors in order to protect the privacy. The proposed privacy-preserving recommender system framework in Chapter 6 incorporates trustworthiness into the weighted nonnegative matrix tri-factorizarion to improve prediction accuracy and privacy protection.

In [48] and [65], the authors proposed group based privacy-preserving recommender systems in which recommendations are made at the group level as opposed to the individual level. Their experimental results show that user groups can be used as the natural protective mechanism for achieving promising privacy protection. In [48], items are grouped in terms of their ratings so users' public interests and private interests can be separated. Then, preferences of group members are identified and aggregated. Recommendations are made by personalizing the group preferences locally to conceal users' private interests. Motivated by this framework, in Chapter 6, items and users are grouped according to the factor matrices of NMF in centralized systems to distinguish the real users from the attackers.

2 An Empirical Study of Recommendation Algorithms

In this chapter, several classical recommendation algorithms, namely the popularity based model, the item similarity based model, the SVD based model, and the bipartite graph model, are studied on the clicking history datasets of online shopping websites collected by an American retargeting company. A massive amount of literature in recommender systems examined the models on rating datasets, such as the Netflix movie rating data [4], the MovieLens dataset [64], and the Jester dataset [22]. While rating information is directly connected to people's preferences, it is not always available. Although a majority of online shopping websites provide rating mechanisms for people to leave feedback on products, there are a few companies that might only be able to collect users' clicking data due to technical restrictions. For example, the retargeting companies usually insert a piece of JavaScript code into the web pages of online vendors to keep track of users' clicking behaviors. They do not have the permission to obtain or utilize data other than this. The datasets from the retargeting company contains only clicking history, e.g., a certain user clicked the link to a particular product. It is interesting to investigate the above recommendation algorithms on binary browsing data instead of numerical rating data with respect to prediction accuracy.

2.1 Description of the Models

2.1.1 Notational Conventions

A matrix is used to store the clicking relationships between users and items, called the user-item rating[1] matrix, denoted by R. Assume there are m users and n items,

[1] Although the values in this matrix are not really ratings, the matrix is still called the rating matrix for consistency.

then $R \in \mathbb{R}^{m \times n}$. An entry r_{ij} is the click count that user i did on item j in the given time period. Though there is no definite proof that a user who clicks an item will buy it, a high click count may imply that the user is interested in it. Since each user may only click a few items and a single item only receives a small number of clicks, R is incomplete, meaning that there are many missing entries.

For user i and item j, the existing click count from i to j is denoted as r_{ij} and the predicted one is denoted as p_{ij}. The recommended items should be interesting to the active user, i.e., this user is likely to click the recommended items.

2.1.2 Item Popularity Based Model

The item popularity based approaches are very traditional ones in recommender systems. The main idea of the item popularity based models is to recommend the most popular, the most viewed, or the best selling items to users. Although the item popularity based models overlook users' preferences, these kinds of models are still effective to a certain degree, and are adopted as an auxiliary component in recommender systems by many famous online shopping sites, such as eBay, Amazon, etc.

The item popularity based model that is tested in this chapter maintains a popularity list for each data set, denoted by $L = \{t_j\}_{j=1,2\cdots,n}$. The elements in L are items in descending order in terms of their view counts, denoted by np_j.

For the simple implementation of the popularity based top-N recommendation, items corresponding to the first N elements in list L will be recommended. However, these recommended items may not be interesting to a user, which means less accurate predictions. Thus, a further filtering step should be employed in this model to improve the prediction accuracy. The filtering step introduces a new parameter h_i into the model, where

$$h_i = \frac{\text{number of distinct items viewed by user } i}{\text{total number of items viewed by user } i}. \tag{2.1}$$

The value of h_i reveals some of users' browsing habits, such as the user preferring to view an item just once, or preferring to view an item for several times during browsing. In the former case, the filtering step does not recommend the items that have been already viewed by the user. In the latter case, such items could also be presented to the user. A threshold h_t is set to determine whether a further filtering step is necessary for user i:

1. if $h_i < h_t$, recommend the top-N items in L to the user;

2. if $h_i \geq h_t$, perform the filtering step.

Case 2 means the number of distinct items viewed by user i is close to the total number of items viewed by this user. In other words, this user is not prone to click each item multiple times. Then the items that have been clicked by user i will be excluded from the recommendation list which is generated in the first step.

The filtering step is also applicable to other models, such as the item similarity-based model. To utilize the filtering step, other models are required to generate an ordered top-($2N$) item list for top-N recommendation. The top-($2N$) list will take the place of popularity list L.

2.1.3 Item Similarity Based Model

Among all recommender systems, the similarity based models are one of the easiest methods to implement. Papagelis et al. [55] showed that, in most cases, the item similarity based models produce better prediction accuracy than the user similarity based models. In the item similarity based model [55], when recommending items to a user i, the system first retrieves neighbors of the items that have been viewed by this user. It then selects the N most similar neighbors and recommends them to i.

In the real world scenarios, a notable challenge in recommender systems is the cold start problem [56]. It often occurs when users have presented very few to no opinions. To resolve this issue, the item popularity factor is incorporated into the similarity based model to handle new users. Eq. (2.2) calculates the relationship between user i and item j.

$$p_{ij} = \gamma \cdot \frac{1}{|S(j;i)|} \sum_{k \in S(j;i),\ \rho_{jk} > 0} \rho_{jk}^2 + (1 - \gamma) \cdot \frac{np_j}{N_g} \tag{2.2}$$

The first tier of the equation is the similarity score and the second tier is the popularity score. $S(j;i)$ is the set of items that were viewed by user i and are similar to item j. ρ_{jk} is the Pearson correlation coefficient [62] between item j and item k. The popularity score is the ratio between the view count of item j, denoted by np_j, and the global maximum view count, denoted by N_g. γ controls the weight of each part.

The formula for Pearson correlation coefficient is slightly modified to better describe the relationship between two items.

$$\rho_{jk} = \frac{\sum_{t=1}^{d} (x'_{jt} - \bar{x}_j)(x'_{kt} - \bar{x}_k)}{\sqrt{\sum_{t=1}^{d} (x'_{jt} - \bar{x}_j)^2 \sum_{t=1}^{d} (x'_{kt} - \bar{x}_k)^2}}, \tag{2.3}$$

where $x'_{jt} = x_{jt}(1 + \frac{1}{1 + \log nc_t})$ is a variation of x_{jt} (nc_t is the number of items that user i has viewed) and d is the dimensionality of an item vector x_j. Note that each entry in x_j corresponds to a user's click count on this item.

The modification on x_{jt} is based on the premise that users who have clicked fewer items make more contribution to the similarity computation than those who have clicked significantly more items.

2.1.4 SVD Based Latent Factor Model

The latent factor models [42] focus on reducing dimensionality of the user-item rating matrix in order to discover some "latent factors". These factors should best interpret user preferences with the least noise. They can be exploited to approximate the original rating values.

In Paterek's SVD based latent factor model [57], the user-item rating matrix is factorized 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 an f-dimensional factor vector UF_i (i-th row of UF) and IF_j (j-th row of IF), respectively [15]. The prediction of the rating left by user i on item j is made by taking inner product of UF_i and IF_j.

In order to obtain the user and item factor vectors, SVD is applied on the huge incomplete matrix R with all the missing values being set to zeros[2].

$$R_{m\times n} = U_{m\times r} \cdot S_{r\times r} \cdot V^T_{n\times r}, \tag{2.4}$$

where U and V are orthonormal matrices, S is a diagonal matrix with singular values on its diagonal and r is the rank of S.

With SVDLIBC (an SVD-package) [7], the dimension f ($f \leq r$) can be easily specified when decomposing the rating matrix. Hence, the user factor matrix and the item factor matrix are represented by

$$UF_{m\times f} = U_{m\times f} \cdot \sqrt{S_{f\times f}},\ IF_{n\times f} = V_{n\times f} \cdot \sqrt{S_{f\times f}} \tag{2.5}$$

and so a prediction can be made by $R_{ij} = UF_i \cdot IF_j^T$.

2.1.5 Bipartite Graph Model

In this graph model [26], users and items are represented as vertices of a graph and can be divided into two disjoint sets, the item set I and the user set U. Every edge is a connection between a vertex in U and one in I. It corresponds to an entry r_{ij} in user-item rating matrix R, as shown in Figure 2.1.

[2] This is very biased and the missing value problem will be addressed in the following chapters.

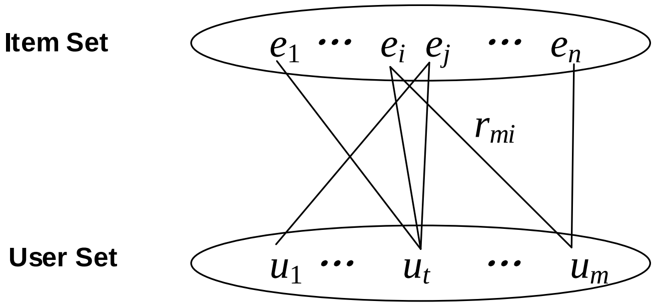


Figure 2.1: A bipartite graph

Sets I and U in the bipartite graph model are independent sets [7]. Therefore, the transition probability between each item pair e_j and e_k can be obtained by Eq. (2.6).

$$P(e_j|e_k) = \sum_{t=1}^{m}[P(e_j|u_t) \cdot P(u_t|e_k)], \tag{2.6}$$

where $P(e_j|u_t) = r_{tj}/\sum_{i=1}^{n} r_{ti}$, and $P(u_t|e_k) = r_{tk}/\sum_{t=1}^{m} r_{tk}$.

All item nodes now form a finite Markov chain with transition matrix $P = [a_{jk}]_{j,k=1,\cdots,n}$, where $a_{jk} = P(e_j|e_k)$ [44], i.e., the probability that this chain ends in the specific item node e_j with initial node e_k is a_{jk}. Therefore, given the previous click history of user i, the probability for a certain item j that this user might be interested in can be predicted according to

$$p_{ij} = \sum_{k=1}^{m}(e_{jk} \cdot T_{ik}), \tag{2.7}$$

where T_i represents the initial state vector for user i in a Markov chain and T_{ik} is the component corresponding to item k. Note that $T_{ik} = r_{ik}/\sum_{j=1}^{n} r_{ij}$.

In order to penalize the users (or items) with a large number of clicks, the penalization parameter α is introduced in the model [26, 46]. It is based on a similar premise that was discussed in the item similarity based model when computing item similarities. Transition probability with penalization parameter α is:

$$P(e_j|u_t) = r_{tj}/(\sum_{i=1}^{n} r_{ti})^{\alpha},\ P(u_t|e_k) = r_{tk}/(\sum_{t=1}^{m} r_{tk})^{\alpha} \tag{2.8}$$

2.2 Experimental Study

2.2.1 Data Description

The dataset in the experiments was gathered by a retargeting company for research purposes. It consists of the browsing history from 139 online shopping websites in one week (08/08/2010 – 08/14/2010). In the dataset, each row represents a transaction, which has four attributes, product ID, website ID, user ID, and date.

Among 139 sites, 4 were selected for test purposes. Statistics are shown in Table 2.1

Table 2.1: Statistics of the data

Site ID	# of Users	# of Items	# of Clicks
3699	20,471	499	134,982
5202	148.409	1,004	300,757
8631	112,738	94	1,559,529
9093	70,049	2,303	120,836

Each dataset is divided into three subsets, the training set, test set and last transaction set. The training set is obtained from the original dataset by removing 1000 active users and their accompanying data. In order to ensure the items that have been viewed by active users also exist in the training set, the items should occur at least 15 times in the training set after the active users' data has been removed. The last transactions of the removed active users form the last transaction set and the remaining data form the test set.

The goal is to train the models using the training set and apply the models on the data in the test set to predict the last transaction of the active users.

2.2.2 Evaluation Strategy

In general, there are two ways to evaluate the prediction accuracy of the recommendation algorithms: hit rates (or recall rates) for top-N recommendation, and error measurement (e.g., root mean square error and mean absolute error) for rating value predictions. Since the datasets are different from Netflix and MovieLens, which provide real ratings, it is more reasonable to make top-N recommendations rather than rating value predictions on the clicking data. Accordingly, the prediction accuracy is evaluated in terms of the hit rates.

The recommended item set is named the predicted set. The hit rate of recommendations (higher is better) is calculated as follows,

$$h_i = \frac{\text{number of correctly predicted active users}}{\text{total number of active users}} \tag{2.9}$$

Within the item popularity based model, an item popularity list is constructed by collecting statistics on the clicking history. The filtering step is applied on this list to obtain the final recommended items.

In the item similarity based model, the parameter γ is tweaked to get the best ratio of similarity score and popularity score. γ is chosen from the interval of [0, 1] with step size 0.1.

The bipartite graph model first builds a probability transition matrix with Eqs. (2.6) and (2.8). The prediction is computed based on the Markov chain in the matrix.

To investigate the influence of the filtering step on other models, this step is applied to the ordered top-$(2N)$ recommendation lists generated by other models to produce new top-N lists.

2.2.3 Results and Discussion

Prior to the comparisons of prediction accuracy, the parameters in the item based similarity model and the bipartite graph model are studied on the website with site ID 5202.

2.2.3.1 Parameter Study

(1) γ in the item based similarity model

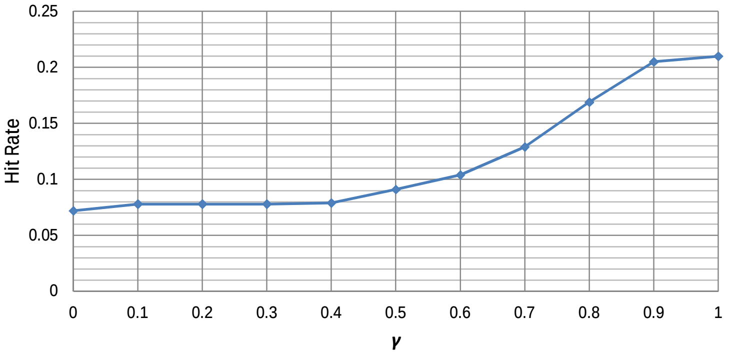


Figure 2.2: Hit rates with different γ on site 5202[3]

The curve in Figure 2.2 shows that with γ increasing, better hit rates are reached. The popularity score does not seem to be more effective than the similarity score. Nevertheless, as stated before, the purpose of using the popularity score is to provide recommendations for new users who have almost no preference. In the experiments, the hit rates are tested by applying the models on users that have clicking records in both the test set and the last transaction set. This test methodology does not necessarily focus on the new user problem. Thus, the popularity score is eliminated by setting γ to 1.0, which means this model will recommend items based on the similarity based score only in later experiments.

[3] "the website with site ID xxxx" and "site xxxx" are used interchangeably.

(2) α in the bipartite graph model

In this model, α penalizes the users or items with lots of clicks. Therefore with a larger α, the corresponding probabilities in Eq. (2.8) become smaller. Table 2.2 shows the hit rates with different α.

Table 2.2: Hit rates with different α on site 5202

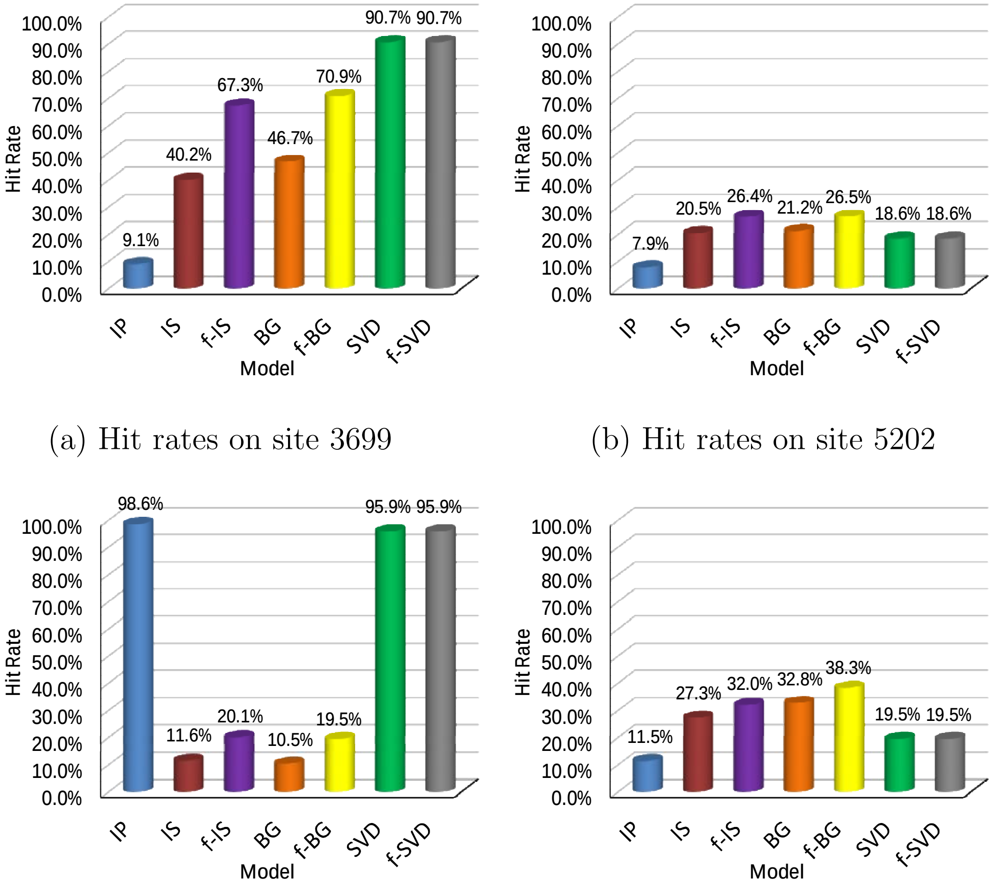
α	Hit Rate
0.5	20%
1	**21.2%**
2	14.2%

In the test datasets, the number of distinct items that each user has clicked does not vary remarkably. Hence the penalization parameter α does not have significant effects on the hit rates. Nevertheless, in grocery shopping [46], a customer purchasing a large number of a specific product reflects a higher interest in this product. Generally speaking, $\alpha = 1$ is suitable for the cases in which the range of values in the rating matrix is not wide.

2.2.3.2 Prediction on Datasets

The four models were first tested on site 3699. Figure 2.3(a) shows the hit rates. IP denotes the item popularity based model; IS denotes the item similarity based model; f-IS denotes the item similarity based model with the filtering step; BG is for the bipartite graph model; f-BG is for the bipartite graph model with the filtering step; SVD is for the SVD-based latent factor model; lastly, f-SVD is for the SVD-based latent factor model with the filtering step.

On this site, the SVD based model with 60 factors achieved the highest hit rate, which is significantly better than other models. More factors were also tested on it but no better results were obtained. This means the first 60 factors were able to capture the most critical latent properties of the items in this dataset.



(a) Hit rates on site 3699

(b) Hit rates on site 5202

(c) Hit rates on site 8631

(d) Hit rates on site 9093

Figure 2.3: Hit rates in top-10 recommendation

The bipartite graph model reached a hit rate of 46.7%, which is close to the results of the item similarity based model. Essentially, BG has a similar principle with IS since they both need to build an item-item matrix. The difference lies on the viewpoint of entries in the matrix – transition probability in BG and item similarity in IS. In fact, some IS models obtain the similarities by computing the conditional probability between items and users.

The item popularity based model performed worst on this dataset. This is because there are 499 items but only 10 items were recommended to each active user. However, the filtering step in IP worked quite well with IS and BG. The hit rate is 67.3% for f-IS (70.9% for f-BG) where the top-20 recommendation by the IS model made a hit rate of 67.6% (71.0% for BG). It means that almost all irrelevant items were filtered out while the correct ones were retained. The investigation on user browsing habits reveals that most users clicked distinct items just once. Therefore they may not be interested in the items they have already clicked.

It is worth mentioning that in the IS model, the neighbors of items that have been viewed by a user may have already been viewed by the same person. Accordingly, the IS based recommendation is not entirely suitable to these kinds of users and a further filtering step is needed.

Nevertheless, the filtering step had no effect on the SVD based model. It can be inferred that the latent factors in SVD not only captured the users' click count information but also the clicking patterns, i.e., the browsing habit, so no filtering step is needed.

For the website with site ID 5202, the results charted in Figure 2.3(b) are quite different from those on 3699. All the models with filtering step except f-SVD performed better than others did. IS and BG, f-IS and f-BG had very similar hit rates, respectively. IP produced the worst prediction accuracy once again. However, SVD with 70 factors, the champion of the previous experiment, only achieved a hit rate of 18.6%. This indicates that the latent factors did not capture the correlations between users and items very well. The f-SVD again had no improvement on SVD.

Figure 2.3(d) presents the results on the website with site ID 9093. The bipartite graph model with filtering step performed best. SVD with 100 factors had a similar hit rate on site 5202. The results show that on some datasets, the local relationships among items that were obtained by BG and IS-like models play a more crucial role in predicting the next item. Whereas on some other datasets, capturing the global effects that were obtained by SVD-like latent factor models is more important.

The prediction results on site 8631 plotted in Figure 2.3(c) look completely different. SVD and IP models had very high hit rates compared to others. In this case, 94 factors, which is the same as the number of items, were used in the SVD model. Note that this website has special properties – it has very few items (94) and a large number of users (112,738). Examining the recommended item list of the SVD model shows that it only generated one item for each user. In other words, the top-1 recommendation of SVD on this site provided correct predictions for 95.9% users. The item popularity based model performed even better since the popular items are welcomed by most users – this differs from that in the first dataset. f-SVD was not employed on this site due to the fact that there was no top-20 list for the filtering step to work on.

An interesting question remains: why did BG and IS perform significantly worse on this dataset compared to others? In [29], Huang et al. discussed the sparsity of the user-item rating matrix which can be used to answer this question. Due to the small number of items and the large number of users, most customers have only clicked a few items. Then the number of edges in the BG model connecting to these users is small. Hence, the transition probability will no longer represent the similarity of products. This also happens in the IS model – an item is similar to almost all other items with very close similarity degrees. If customers in this dataset tend to be interested in several popular items, the prediction accuracy can be very low. This also explains the positive results of the IP model in this case.

Furthermore, if the dataset has many items but very few customers who have viewed several items, the number of edges associated with most customers would be very high. Most entries in the transition matrix will then have small and close values, which will prevent the model from discovering closely related items.

As a summary of top-10 recommendations, Table 2.3 gives the performance statistics of seven models[4] on four datasets. The models are ranked by their hit rates –

[4] Since SVD and f-SVD have the same hit rates, SVD is used to represent both SVD and f-SVD in this table.

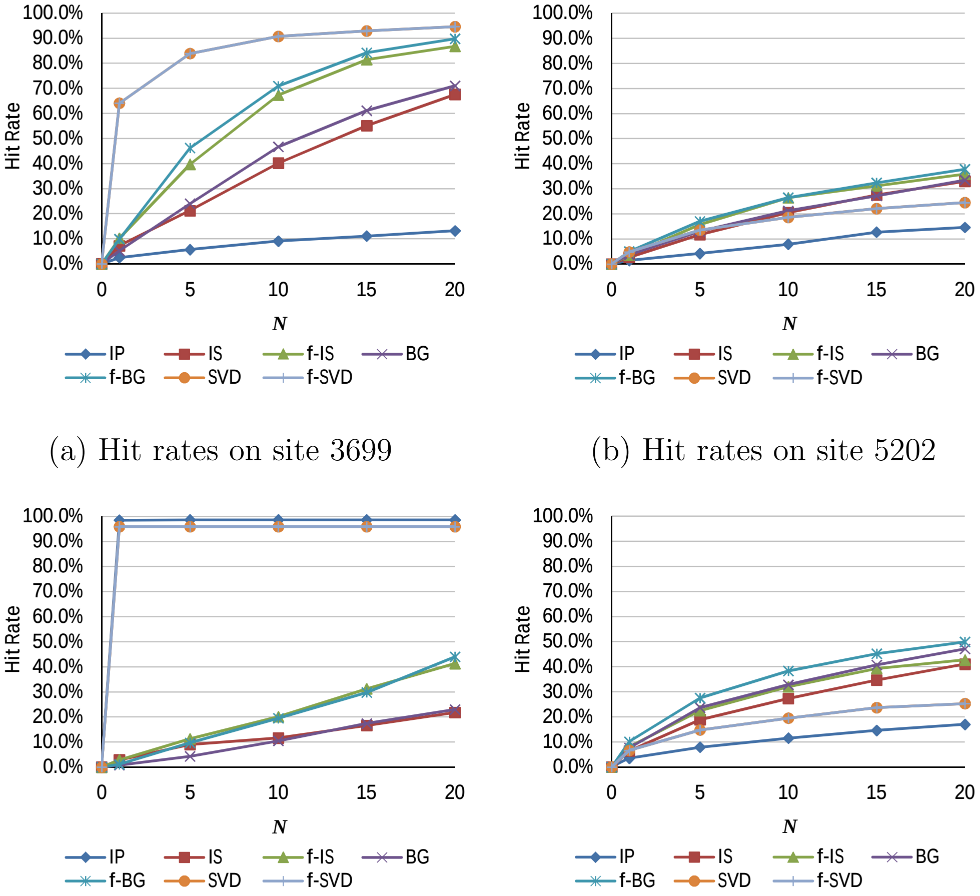
Table 2.3: Performance list

	Performance Rank					
Dataset	IP	IS	f-IS	BG	f-BG	SVD
3699	6	5	3	4	2	**1**
5202	6	4	2	3	**1**	5
8631	**1**	5	3	6	4	2
9093	6	4	3	2	**1**	5

the model that performs best is ranked 1 and the one which performs worst is ranked 6. It is expected that the IP model has the lowest rank (rank 6 in total) since it is only based on the popularity of items. The f-BG and f-IS models attained the first two places. SVD also worked well in most cases. It can be seen that some models predicted very accurately for only certain datasets while f-BG and f-IS models had higher average hit rates than others. Thus, the models with the filtering step, which take into consideration human behavior patterns, can be employed by most recommendation tasks to achieve satisfactory results.

In the end, the hit rates with different N's in top-N recommendation were studied. The predictions with seven models were performed on four sites. Figure 2.4 shows the variance.

The figures show the increasing trends with greater N for all models on all sites. The differences lie in the slope of the curves. The models with the filtering step – f-BG and f-IS had about 20% $\sim$ 50% improvement in accuracy on the original models, BG and IS. In this experiment, site 8631 is still a special one compared to others because the hit rates of SVD and IP reached 95.9% and 98.6% when $N = 5$, respectively. That means, for these two models, the top-5 recommended items were accurately predicted and the hit rates for top-1 predictions are 95.2% and 98.5%. Consequently, on this website, the top-5 recommendations are preferable for SVD and IP models since the number of items recommended to users are expected to be small – users may not be interested in a top-50 or top-100 recommendation list as they are not helpful at all.



(a) Hit rates on site 3699

(b) Hit rates on site 5202

(c) Hit rates on site 8631

(d) Hit rates on site 9093

Figure 2.4: Hit rates in top-N recommendation

2.3 Summary

In this chapter, several classical recommendation algorithms are studied on the datasets from an American retargeting company, to find a good strategy in model selection for specific datasets. The experimental results reveal that, if a dataset has few items but a large number of users, the SVD based model and the item popularity based model can be good choices. Whereas if a dataset has many items but fewer users, the bipartite graph model and models with the filtering step (except SVD) are suitable to it. However, if the designer of the recommender system wants to build one for general purposes, the bipartite graph model with the filtering step can be selected due to its higher average performance in the experiments. The results also show that the filtering step had no effect on the SVD based model which indicates that the latent factors can capture both rating information and user clicking patterns.

3 SVD Based Privacy-Preserving Data Update Scheme in Collaborative Filtering

It was mentioned in Chapter 1 that in some scenarios, data owners need to share their data with a third party. This behavior gives rise to the privacy leakage problem. There are two challenges during the data sharing process: (1) how to protect customers' private information while keeping data utility; (2) based on (1), how to handle data growth efficiently.

In this chapter, a privacy-preserving data update scheme is proposed for collaborative filtering based recommender systems. This scheme utilizes truncated SVD update algorithms [9, 38] and randomization techniques. It can provide privacy protection when incorporating new data into the original one in an efficient way. The scheme starts with the precomputed SVD of the original rating matrix. New rows/columns are then built into the existing factor matrices. Users' privacy is preserved by truncating the new matrix together with randomization and post-processing. It also takes into account the missing value imputation during the update process to provide high quality data for accurate recommendations. Results of the experiments conducted on the MovieLens dataset [64] and the Jester dataset [22] show that the proposed scheme can handle data growth efficiently and keep a low level of privacy loss. The prediction accuracy is still at a high level compared to most published results.

3.1 Problem Description

Assume the data owner has a user-item rating matrix, denoted by $R \in \mathbb{R}^{m \times n}$, where there are m users and n items. Differing from the click count matrix in Chapter 2, the entry r_{ij} in R here represents the rating left on item j by user i. The valid range of rating value varies from website to website. Some websites use the $1 \sim 5$ scale with

1 as the lowest rating (most disliked) and 5 as the highest rating (most favorated) while some others use the $-10 \sim 10$ scale with -10 as the lowest rating, 0 as neutral rating, and 10 as the highest rating.

The original rating matrix contains the real rating values left by users on items, which means it can be used to identify the shopping patterns of users. These patterns can reveal some of the users' privacy, so releasing the original rating data without any privacy protection will cause a privacy breach. One possible way to protect user privacy before releasing the rating matrix is to impute the matrix and then perturb it. In this procedure, imputation estimates the missing ratings as well as conceals the user preference on particular items; no missing value means there is no way to tell which items have been rated by users since all items are marked as rated. On the other hand, the perturbation distorts the ratings so that users' preferences on particular items are blurred.

When new users' transactions arrive, the new rows (each row contains the ratings left on items by the corresponding user), denoted by $T \in \mathbb{R}^{p \times n}$, should be appended to the original matrix R:

$$\begin{bmatrix} R \\ T \end{bmatrix} \to R'. \tag{3.1}$$

Similarly, when new items arrive, the new columns (each column contains the ratings left by users on the corresponding item), denoted by $G \in \mathbb{R}^{m \times q}$, should be appended to the original matrix R:

$$\begin{bmatrix} R & F \end{bmatrix} \to R''. \tag{3.2}$$

To protect users' privacy, the new rating data must be processed before it is released. $T_r \in \mathbb{R}^{p \times n}$ is adopted to denote the processed new rows and $G_r \in \mathbb{R}^{m \times q}$ is for processed new columns.

3.2 Privacy-Preserving Data Update Scheme

This section presents the data update scheme in collaborative filtering that could preserve the privacy during the update process. Users' privacy is protected in three aspects, 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 to some extent while leaving the distribution unchanged. On the other hand, the truncated SVD is a naturally ideal choice for data perturbation, since it captures the latent properties of a matrix and eliminates the useless noise. If given a well-chosen truncation rank, SVD can provide reasonable balance between data privacy and 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 R and further perturbed to protect users' privacy. In the following sections, the proposed scheme would be discussed in the row update and column update separately.

3.2.1 Row Update

In Eq. (3.1), T is added to R as a series of rows. The new matrix R' has a dimension of $(m+p) \times n$. Prior to the updates, it is assumed that the truncated rank-k SVD of R has been computed previously, as

$$R_k = U_k \cdot \Sigma_k \cdot V_k^T, \tag{3.3}$$

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 3.1, the user-item rating matrix is an incomplete matrix thus before it is factorized, the missing values must be imputed. Similar to [64], the column mean rating values are exploited 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 update the SVD.

For new rows T, before incorporating them into the existing matrix, an imputation step is performed. In this step, the empty entries should be filled with values that have knowledge from both the mean values of the existing matrix and the ratings in the new data. Eq. (3.4) calculates the new column mean.

$$\bar{r}'_j = \frac{m \times \bar{r}_j + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} r_{ij}}{m + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} 1} \tag{3.4}$$

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

The imputed matrices, $\hat{R}$ (with its factor matrices $\hat{U}_k, \hat{\Sigma}_k$ and $\hat{V}_k$) and $\hat{T}$ are then obtained. Now the problem space has been converted from Eq. (3.1) to Eq. (3.5):

$$\begin{bmatrix} \hat{R} \\ \hat{T} \end{bmatrix} \rightarrow \hat{R}' \tag{3.5}$$

After imputation, random noise drawn from Gaussian distribution is added to the new data $\hat{T}$, yielding $\dot{T}$. The update of the matrix follows the procedure in [72]. 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

$$\begin{aligned}\hat{R}' = \begin{bmatrix}\hat{R} \\ \hat{T}\end{bmatrix} \approx \begin{bmatrix}\hat{R}_k \\ \hat{T}\end{bmatrix} \approx \begin{bmatrix}\hat{R}_k \\ \dot{T}\end{bmatrix} \\ = \begin{bmatrix}\hat{U}_k & 0 \\ 0 & I_p\end{bmatrix} \begin{bmatrix}\hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T\end{bmatrix} \begin{bmatrix}\hat{V}_k & Q_T\end{bmatrix}^T\end{aligned} \tag{3.6}$$

The rank-k SVD is then computed on the middle matrix,

$$\begin{bmatrix}\hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T\end{bmatrix}_{(k+p)\times(k+p)} \approx U'_k \cdot \Sigma'_k \cdot V_k'^T \tag{3.7}$$

Since $(k+p)$ is typically small, the computation of the SVD should be very fast. Same as [75], the truncated rank-k SVD of $\hat{R}'$ instead of a complete one is computed,

$$\hat{R}'_k = \left(\begin{bmatrix}\hat{U}_k & 0 \\ 0 & I_p\end{bmatrix} \cdot U'_k\right) \cdot \Sigma'_k \cdot \left(\begin{bmatrix}\hat{V}_k & Q_T\end{bmatrix} \cdot V'_k\right)^T \tag{3.8}$$

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.

$$\Delta\hat{r}'_{k,ij} = \begin{cases} validMinValue & if \;\; \hat{r}'_{k,ij} < validMinValue \\ validMaxValue & if \;\; \hat{r}'_{k,ij} > validMaxValue \\ \hat{r}'_{k,ij} & otherwise \end{cases} \tag{3.9}$$

In Eq. (3.9), $\hat{r}'_{k,ij}$ is the (i,j)-th entry of $\hat{R}'_k$. $validMinValue$ and $validMaxValue$ depend on particular dataset. For the MovieLens dataset, $validMinValue = 0$ and $validMaxValue = 5$; for the 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 this scheme, it is assumed that the third party owns $\hat{R}_k$ so only ΔT ($\Delta T = \Delta\hat{R}'_k(m+1:m+p,:) \in \mathbb{R}^{p\times n}$)[1] is sent to it.

Algorithm 3.1 summarizes the SVD based row update.

Algorithm 3.1 Privacy-Preserving Row Update

```
Input:
    Precomputed 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 $T \in \mathbb{R}^{p\times n}$;
Output:
    SVD for the updated full matrix: $\hat{U}'_k, \Sigma'_k$ and $\hat{V}'_k$;
    Perturbed new data: $\Delta T$;
    Updated item mean vector: $\vec{r'}_{mean}$;

Impute the missing values in $T$ with Eq. (3.4) and update the item mean vector $\rightarrow \hat{T}, \vec{r'}_{mean}$;
Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{T} \rightarrow \dot{T}$;
Perform QR factorization on $\ddot{T} = (I_n - \hat{V}_k \cdot \hat{V}_k^T) \cdot \dot{T}^T \rightarrow Q_T \cdot S_T$;
Perform SVD on $\ddot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & 0 \\ \dot{T}\hat{V}_k & S_T^T \end{bmatrix} \rightarrow \ddot{\Sigma} \approx U'_k \cdot \Sigma'_k \cdot V'^T_k$;
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 $\hat{R}' \rightarrow \hat{R}'_k = \hat{U}'_k \cdot \Sigma'_k \cdot \hat{V}'^T_k$;
Process the invalid values by Eq. (3.9) $\rightarrow \Delta\hat{R}'_k$;
$\Delta\hat{R}'_k(m+1:m+p,:) \rightarrow \Delta T$;
Return $\hat{U}'_k, \Sigma'_k, \hat{V}'_k, \Delta T$ and $\vec{r'}_{mean}$.
```

3.2.2 Column Update

The column update is similar to the row update, however, there are several differences between them. It is worth mentioning that item means are used to impute the missing values in the raw user-item rating matrix. In the row update, the mean values change

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

when the new rows/users are added while in the column update, the mean values only depend on the new columns/items. With this property, it is not necessary to keep an item mean vector in the column update.

Like Eq. (3.5), column update has the following task:

$$\begin{bmatrix} \hat{R} & \hat{F} \end{bmatrix} \rightarrow \hat{R}'' \tag{3.10}$$

Algorithm 3.2 depicts the SVD based column update.

Algorithm 3.2 Privacy-Preserving Column Update

```
Input:
    Precomputed rank-$k$ SVD of $\hat{R}$: $\hat{U}_k, \hat{\Sigma}_k$ and $\hat{V}_k$;
    New data $F \in \mathbb{R}^{m \times q}$;
Output:
    SVD for the updated full matrix: $\hat{U}''_k, \Sigma''_k$ and $\hat{V}''_k$;
    Perturbed new data: $\Delta F$;

Impute the missing values in $F$ with corresponding item mean values $\rightarrow \hat{F}$;
Apply random noise $X(X \sim N(\mu, \sigma))$ to $\hat{F} \rightarrow \dot{F}$;
Perform QR factorization on $\ddot{F} = (I_m - \hat{U}_k \cdot \hat{U}_k^T) \cdot \dot{F} \rightarrow Q_F \cdot S_F$;
Perform SVD on $\dot{\Sigma} = \begin{bmatrix} \hat{\Sigma}_k & \hat{U}_k^T \cdot \dot{F} \\ 0 & R_F \end{bmatrix} \rightarrow \dot{\Sigma} \approx U''_k \cdot \Sigma''_k \cdot V''^T_k$;
Compute $\left( \begin{bmatrix} \hat{U}_k & Q_F \end{bmatrix} \cdot U''_k \right) \rightarrow \hat{U}''_k$
Compute $\left( \begin{bmatrix} \hat{V}_k & 0 \\ 0 & I_q \end{bmatrix} \cdot V''_k \right) \rightarrow \hat{V}''_k$
Compute the rank-$k$ approximation of $\hat{R}'' \rightarrow \hat{R}''_k = \hat{U}''_k \cdot \Sigma''_k \cdot \hat{V}''^T_k$;
Process the invalid values like Eq. (3.9) ($\hat{r}'_{k,ij}$ are now entries in $\hat{R}''_k$) $\rightarrow \Delta\hat{R}''_k$;
$\Delta\hat{R}''_k(:, n+1 : n+q) \rightarrow \Delta F$;
Return $\hat{U}''_k, \Sigma''_k, \hat{V}''_k$ and $\Delta F$.
```

The data owner should keep the updated factor matrices for the new user-item rating matrix ($\hat{U}'_k, \Sigma'_k$ and $\hat{V}'_k$ for the row update, $\hat{U}''_k, \Sigma''_k$ and $\hat{V}''_k$ for the column update) and the perturbed new data matrix (ΔT for the row update, ΔF for the column update). Moreover, the updated item mean $\vec{r'}_{mean}$ is also supposed to be held by the data owner if a row update 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. The truncated SVD update eliminates the factors that are trivial 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.

3.3 Experimental Study

3.3.1 Data Description

The experiments were conducted on the MovieLens [64] and Jester [22] datasets. The public MovieLens dataset has 3 subsets, 100K (100,000 ratings), 1M (1,000,000 ratings) and 10M (10,000,000 ratings). The first dataset, which is adopted in the experiments, has 943 users and 1,682 items. The 100,000 ratings, ranging from 1 to 5, were divided into two parts: the training set with 80,000 ratings and the test set with 20,000 ratings. Both sets are stored in matrices so they are very incomplete (93.7% of the entries in the training matrix are not observed).

The Jester datasets were from a web based joke recommendation system, which was developed by the University of California, Berkeley [22]. There are also three subsets, namely jester-data-1, jester-data-2 and jester-data-3, among which, the first one was chosen. It has 24,983 users and 100 jokes with 1,810,455 ratings ranging from -10 to 10. 80% of the ratings were randomly selected as the training set and the rest were used as the test set. Compared to MovieLens, the Jester dataset is not so incomplete (27.5% of the entries in the training matrix are not observed).

3.3.2 Prediction Model and Error Measurement

In Section 2.1.4, an SVD based CF model that predicts the ratings by utilizing users' latent factors is described. In this chapter, the same model is exploited to test the proposed data update scheme. Since the SVD can only work on complete matrices, the missing values in the rating matrices are treated as zeros if no pre-processing is performed. A typical way to impute missing values is using item means. For each column, the mean value is calculated from the existing ratings and all the missing values in this column are filled by the mean value.

Assume p'_{ij} is the predicted value computed by the SVD based CF model; to ensure the predicted ratings are in the valid range, the same boundary check like Eq. (3.9) is applied:

$$p_{ij} = \begin{cases} validMinValue & if \;\; p'_{ij} < validMinValue \\ validMaxValue & if \;\; p'_{ij} > validMaxValue \\ p'_{ij} & otherwise \end{cases} \tag{3.11}$$

When testing the prediction accuracy, the user factor matrix UF and the item factor matrix IF (see Eq. (2.5)) were first obtained from the training set; then for every rating in the test set, the corresponding predicted value was computed and the differences were measured. This was done for all the ratings in the test set and the MAE (mean absolute error) [10, 66] can be calculated as follows:

$$MAE = \frac{1}{|TestSet|} \sum_{r_{ij} \in TestSet} |r_{ij} - p_{ij}| \tag{3.12}$$

3.3.3 Privacy Measurement

When measuring the privacy, Definition 3.1 defines the privacy level.

Definition 3.1. ***Privacy level*** $\Pi(Y|X)$ *is a metric that indicates the extent to which a random variable* Y *could be estimated if given a random variable* X*.*

$$\Pi(Y|X) = 2^{h(Y|X)}, \tag{3.13}$$

where $h(Y|X)$ is the differential entropy of Y given X.

This privacy measure was proposed by Agrawal and Aggarwal [3] and was applied to measure the privacy in collaborative filtering by Polat and Du [59]. In addition to the privacy level, Agrawal and Aggarwal [3] also proposed the conditional privacy loss of Y after revealing X:

$$\mathcal{P}(Y|X) = 1 - \Pi(Y|X)/\Pi(Y) = 1 - 2^{h(Y|X)}/2^{h(Y)} \tag{3.14}$$

Similar to Polat and Du's work, the experiments measure privacy by $\Pi(Y|X)$ and $\mathcal{P}(Y|X)$.

3.3.4 Evaluation Strategy

The proposed scheme was tested in several aspects: the prediction accuracy in recommendation, the privacy protection level, how to split the new data in the update, when to recompute SVD, and the randomization degree with its effect in perturbations, etc.

To test when to recompute SVD, the data in the training set, which is viewed as a rating matrix, was split into two subsections with a particular ratio ρ_1. Assuming the first ρ_1 data has already been processed, the remaining data is then updated into it. For instance, when rows are split with $\rho_1 = 40\%$, the first 40% of the rows in the training set is treated as R in Eq. (3.1). The imputation would be done on this data without the knowledge from the remaining 60% of the data, yielding $\hat{R}$ in Eq. (3.5). Then a rank-k SVD and the item mean vector are computed on this matrix. The rank-k approximation of $\hat{R}$ is named the starting matrix. These data structures are utilized as the input of Algorithm 3.1. Results are expected to be different with the varying split ratio. If the result is too far from the predefined threshold or the results evolve more slowly or even start to degrade at some point, a recomputation is anticipated.

However, the remaining 60% of the rows in the training set are not simply updated in one round since data in real world applications usually grows incrementally. In the experiments, the 60% of the rows were added to the starting matrix in several rounds, depending on another split ratio ρ_2. For example, if $\rho_2 = 1/10$, the new data will be added to the starting matrix in 10 rounds. The final matrix, which equals starting matrix $+ \Delta T_1 + \cdots + \Delta T_{10}$, is the perturbed and updated matrix.

The algorithms were evaluated on both the MovieLens and Jester datasets by testing the time cost of the update, the prediction error, and the privacy measure on the final matrix.

3.3.5 Results and Discussion

3.3.5.1 Truncation Rank (k) in SVD

Due to the characteristics of SVD based CFs, the rank of the truncated matrix, k, must be chosen in advance. Most papers reported that $k = 13$ is an optimal choice for the MovieLens dataset and $k = 11$ for the Jester dataset. It was verified in the experiments by probing k in $\{2, 5, \ldots, 25, 50, 100\}$ and computing the corresponding MAE's [64]. The results on MovieLens are shown in Figure 3.1. Note that this experiment is unrelated to the update since the SVD based predictions were performed on the full imputed training data.

The curve shows the mean absolute errors with different k's and the lowest MAE (0.7769) was reached when $k = 13$. The same experiment on Jester also confirms that the lowest MAE (3.2871) was reached when $k = 11$. Accordingly, the truncation ranks for the MovieLens and Jester datasets are set to 13 and 11 in the following experiments, respectively.

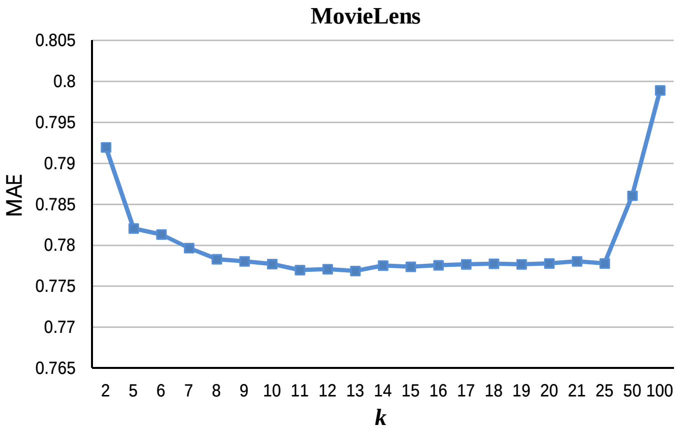


Figure 3.1: MAE variation with different rank-k

3.3.5.2 Split Ratio ρ_2

In this experiment, ρ_1 was fixed at 40%, meaning that the first 40% of the data in the training set is treated as the starting matrix, while the remaining 60% will be added to it. ρ_2 is set to 1/10, 1/9, 1/8, . . . ,1/2, and 1. The greater ρ_2 is, the fewer rounds will be needed in the update.

Figure 3.2 illustrates the time cost with the different split ratio ρ_2. The row update is represented by "Row", while "Column" refers to the column update. To eliminate randomization in both algorithms, μ and σ were set to zero. The results with randomization are reported in Section 3.3.5.4.

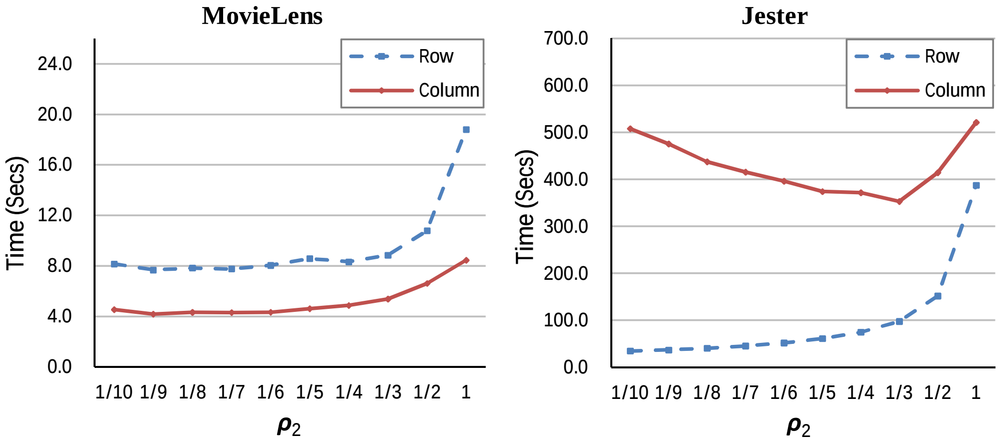


Figure 3.2: Time cost variation with split ratio ρ_2

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

Furthermore, the figure indicates that the time cost of the update depends on dimensionality of the rows and columns. For example, the MovieLens dataset has more columns (1,682 items) than rows (943 users) while the Jester dataset has fewer columns (100 items) than rows (24,983 users). Each step of both the row and column update algorithms shows that when the number of columns is greater than the number of rows, steps 1 and 3 in Algorithm 3.1 need more time than those in Algorithm 3.2 due to higher dimensionality and vice versa. Nevertheless, compared to the time cost for imputing the missing values and computing the SVD on raw training set, which is 44.9744 (imputation) + 2.2866 (SVD) = 47.261s for MovieLens and 1,592.8075 (imputation) + 3.7552 (SVD) = 1,596.5627s for Jester, the proposed scheme ran more efficiently in both row and column updates.

The mean absolute error charted in Figure 3.3 keeps stable with the different split ratio ρ_2 which implies that the quality of the updated data with respect to prediction accuracy is not affected notably by ρ_2. Similar results were obtained on the privacy measure, see Figure 3.4.

Based on the experimental results about split ratio ρ_2, it was fixed to 1/9 in both the row and column updates for the MovieLens data in the following experiments. The Jester data worked with $\rho_2 = 1/10$ in the row update and $\rho_2 = 1/3$ in the column update.

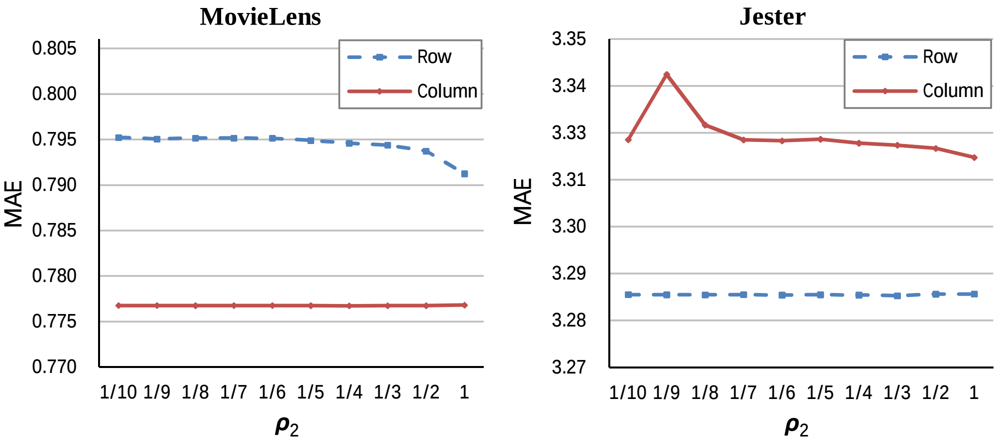


Figure 3.3: MAE variation with split ratio ρ_2

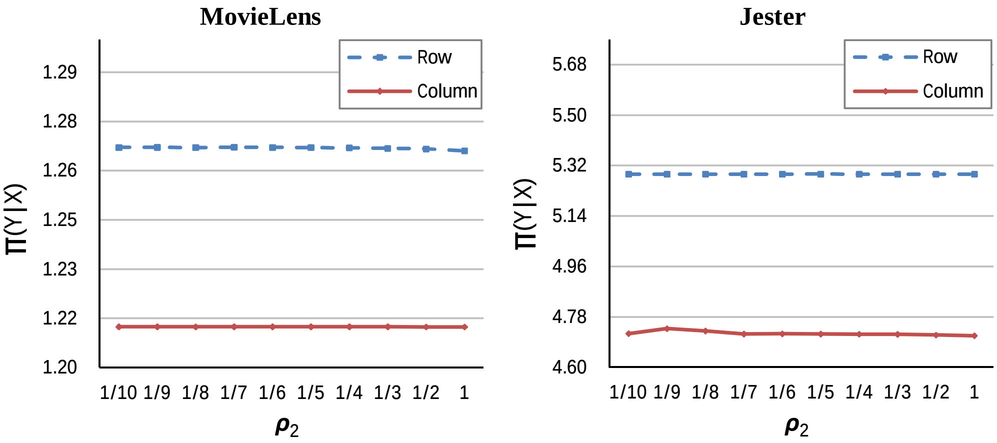


Figure 3.4: Privacy level variation with split ratio ρ_2

3.3.5.3 Split Ratio ρ_1

Because of the inherent properties of the SVD update algorithms, errors are generated in each run. The data owners should be aware of the correct time to recompute SVD for the whole data so that the quality of the data can be kept. This problem is studied by experimenting with the split ratio ρ_1.

The time cost for updating new data with varying ρ_1 is plotted in Figure 3.5. It is expected that updating fewer rows/columns takes less time. While different types of split ratios were tested, the relation between the time cost of the row and column updates did not change.

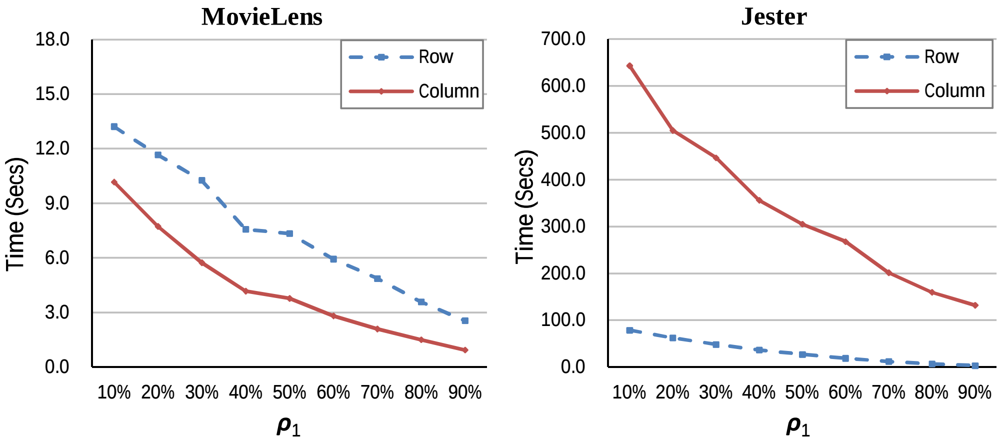


Figure 3.5: Time cost variation with split ratio ρ_1

Figure 3.6 shows the mean absolute error. The curve in the MovieLens data has a descending trend in the row update but keeps at a stable level in the column update. The case is different for the Jester data where the MAE is generally decreasing for the column update and keeps stable for the row update with rising split ratio ρ_1. It indicates that with fewer ratings in the starting matrix, the prediction model tends to less accurately profile users' preferences and thus leads to a lower prediction accuracy. In this case, the users in the MovieLens data affect more while the dominant factor of the Jester data is the items. It can be assumed that the total amount of information stored in a rating matrix is fixed, and every matrix entry contributes the same amount of information. Therefore, the fewer users (or items) we have, the more information each user (or item) can provide. In the MovieLens data, the row dimension is lower than the column dimension. In this scenario, users play a more important role than items because there are fewer users than items and each user contributes more than each item does. Therefore, with the increasing number of users, the MAE dropped. On the other hand, in the Jester data, the row dimension is higher than the item dimension so items are more critical and have a greater effect on errors.

As for the prediction errors, when $\rho = 40\%$, the MAE of the unperturbed training matrix is 0.7769 for MovieLens and 3.2871 for Jester. The update schemes reached

0.7951 for the row update and 0.7768 for the column update on the MovieLens dataset and 3.2870 for the row update and 3.3221 for the column update on the Jester data. The MAE's are still comparable to the published results. If using more sophisticated prediction models, the MAE could be lower.

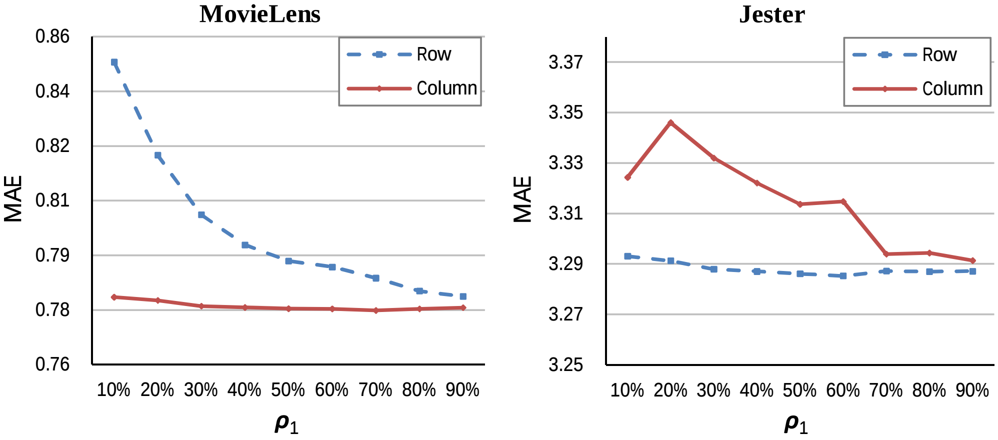


Figure 3.6: MAE variation with split ratio ρ_1

The privacy level with the varying split ratio is displayed in Figure 3.7. It is apparent that the privacy level decreases when the starting matrix holds more data. In this experiment, the privacy levels on both datasets were higher and changed faster in the row update than those in the column update. The results imply that the privacy with respect to users (rows) plays a dominant role in the update. This is reasonable because when people talk about the privacy, they mean the users' privacy and not the items'.

Corresponding to the privacy level, the privacy loss of the raw training data (Y) after revealing the perturbed and updated data (X) is presented in Figure 3.8.

With the growing split ratio, the privacy loss increases where the privacy level decreases. The curves in Figure 3.8 looks like an upside-down version of the curves in Figure 3.7 due to the relation between them, see Eq. (3.14).

Now it can be decided when to recompute the SVD for the whole data according to Figures 3.6 and 3.7. Since MAE's on both datasets drop more slowly after $\rho_1 \geq 50\%$

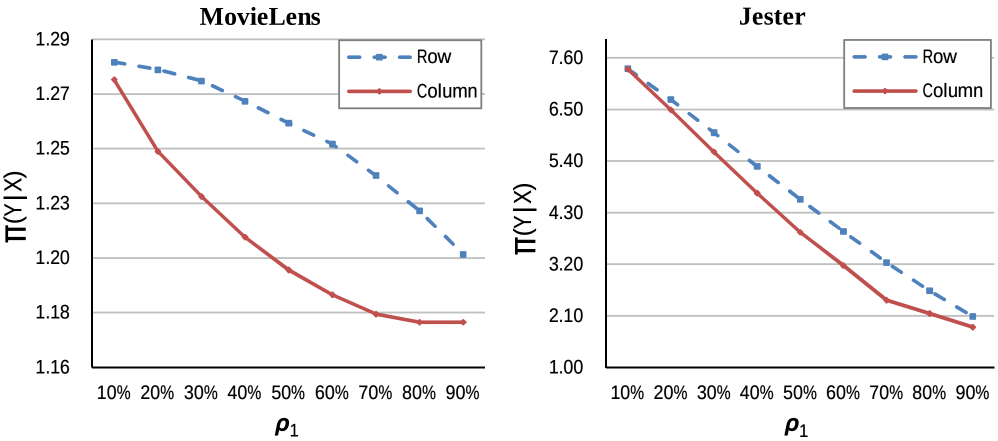


Figure 3.7: Privacy level variation with split ratio ρ_1

and there is no apparent variation of the slope for the privacy measure curves, the recomputation can be performed when ρ_1 reaches 50%.

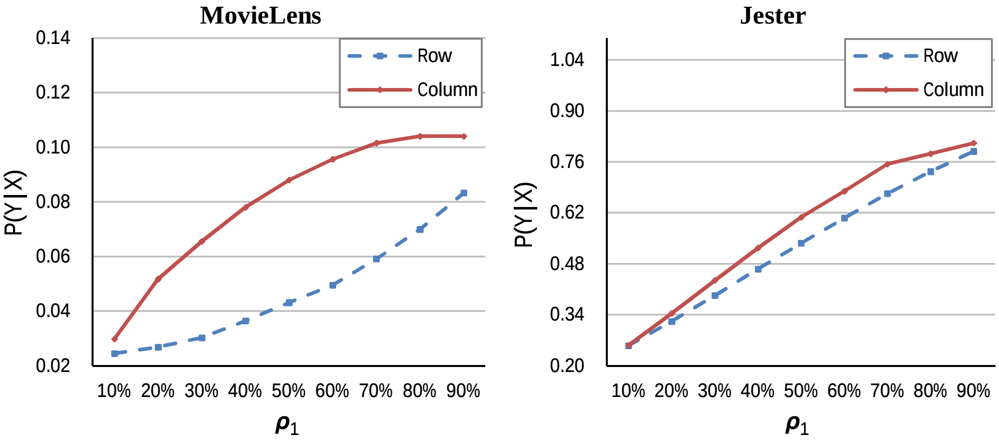


Figure 3.8: Privacy loss variation with split ratio ρ_1

3.3.5.4 Impact of Randomization in Data Updates

So far, randomization technique has not been applied to the proposed data update scheme. In this section, the impact of randomization (Gaussian noise with μ and σ as its parameters in Algorithms 3.1 and 3.2) is studied 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. μ is probed in {0, 1} and σ is probed in {0.1, 1} for both datasets. Table 3.1 collects the statistics of the test.

Table 3.1: Impact of randomization in data updates

		MovieLens Data			
		Row Update		Column Update	
μ	σ	MAE	$\Pi(Y\|X)$	MAE	$\Pi(Y\|X)$
0	0	0.7951	1.2671	0.7768	1.2124
0	0.1	0.7955	1.2792	0.7781	1.2558
0	1	0.8331	1.2927	0.8219	1.2869
1	0.1	1.0764	1.2808	0.9837	1.2583
1	1	1.0421	1.2926	0.9258	1.2839

		Jester Data			
		Row Update		Column Update	
μ	σ	MAE	$\Pi(Y\|X)$	MAE	$\Pi(Y\|X)$
0	0	3.2870	5.2894	3.3221	4.7178
0	0.1	3.2872	5.4717	3.3390	4.9811
0	1	3.3007	6.4436	3.3542	6.1920
1	0.1	3.3221	5.4706	3.3629	5.0179
1	1	3.3358	6.4410	3.3799	6.2577

In this table, the row and column updates with the randomization technique is compared to the non-randomized version. It is obvious that after applying the random noise to the new data before the update, both privacy metrics ($\Pi(Y|X)$ and $\mathcal{P}(Y|X)$) in all cases improved to a certain extent. Nevertheless, some utility of the data was lost which resulted in greater MAE's 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 affected the results to a greater extent than the standard deviation σ did. It is a good idea to determine μ first and tweak σ afterwards.

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

3.4 Summary

In this chapter, a privacy-preserving data update scheme for collaborative filtering purposes is presented. It is an incremental SVD based scheme with the randomization technique and could be utilized in updating incremental user-item matrices and preserving privacy at the same time. The scheme attempts to protect users' privacy in three aspects, missing value imputation, randomization based perturbation and SVD truncation. The experimental results on the MovieLens and Jester datasets show that the proposed scheme could update new data into the existing data very quickly. It can also provide high quality data for accurate recommendations while keeping the privacy.

4 Incorporating Auxiliary Information into Collaborative Filtering Data Update with Privacy Preservation

In Chapter 3, an SVD based data update scheme is presented. It incorporates the missing value imputation and randomization based perturbation techniques as well as a post-processing procedure into the incremental SVD to update the new data with privacy preservation. Nevertheless, the time complexity contains a cubic term with respect to the number of new rows or columns. This is a potentially expensive factor in the update process, especially when a large amount of new data comes in. It is expected that a better technique can be developed to improve the update process.

Furthermore, it is beneficial to utilize the auxiliary information of users and/or items that comes with the datasets, e.g., the MovieLens dataset [64] and the LibimSeTi Dating Agency dataset [11]. Typical auxiliary information includes user demographic data, item category data, etc. This information, if properly used, can improve the recommendation accuracy, especially when the original rating matrix contains a large number of missing values.

This chapter discusses an NMF based data update approach that solves these issues. The approach, named iAux-NMF is based on the incremental nonnegative matrix tri-factorization algorithms [20]. It starts with computing the weighted and constrained nonnegative matrix tri-factorization for the original incomplete rating matrix, utilizing both the rating matrix itself and the auxiliary information. The factor matrices of NMF are then used to approximate the original rating matrix with the missing values imputed. Meanwhile, the data is automatically perturbed due to the intrinsic properties of NMF [74]. For new data, iAux-NMF is performed to produce imputed and perturbed data. By doing so, even though the third party has this data on hand, it does not know which ratings it can trust or to what extent it can trust the ratings. Therefore, users' privacy is protected. Experimental results on the MovieLens and LibimSeTi datasets show that this approach could update the new data very fast with low levels of privacy loss and high levels of data utility.

4.1 Problem Description

In Chapter 3, the data update problem is discussed. In this chapter, the case is further extended. In addition to the incomplete user-item rating matrix $R \in \mathbb{R}^{m \times n}$, the data owner has two more matrices: a user feature matrix $F_U \in \mathbb{R}^{m \times k_U}$, and an item feature matrix $F_I \in \mathbb{R}^{n \times k_I}$, where there are m users, n items, k_U user features, and k_I item features.

The user feature matrix F_U and the item feature matrix F_I represent the auxiliary information of users and items, respectively. They are taken into account to help impute the missing entries in the rating matrix for better accuracy. The processed matrix $R_r \in \mathbb{R}^{m \times n}$ is the one that will be transferred to the third party.

When new users' ratings arrive, the new rows, denoted by $T \in \mathbb{R}^{p \times n}$, should be appended to the original matrix R (see Eq. (3.1)). Meanwhile, their auxiliary information is also available, and thus the updated feature matrix is

$$\begin{bmatrix} F_U \\ \Delta F_U \end{bmatrix} \to F'_U, \tag{4.1}$$

where $\Delta F_U \in \mathbb{R}^{p \times k_U}$.

Similarly, when new items become available, the new columns, denoted by $G \in \mathbb{R}^{m \times q}$, should be appended to the original matrix R[1],

$$\begin{bmatrix} R & G \end{bmatrix} \to R'' \tag{4.2}$$

and the updated item feature matrix is

[1] Eq. (3.2) is rewritten due to a different notation G

$$\begin{bmatrix} F_I \\ \Delta F_I \end{bmatrix} \rightarrow F_I', \tag{4.3}$$

where $\Delta F_I \in \mathbb{R}^{q \times k_I}$.

4.2 Using iAux-NMF for Privacy-Preserving Data Updates

This section introduces the iAux-NMF algorithm and its application in incremental data update with privacy preservation.

4.2.1 Aux-NMF

While iAux-NMF handles the incremental data update, it is necessary to present the non-incremental version, named Aux-NMF beforehand. This section is organized as follows: developing the objective function, deriving the update formulas, analyzing the convergences, and the detailed algorithms.

4.2.1.1 Objective Function

Nonnegative matrix factorization (NMF) [45] is a widely used dimension reduction method in many applications such as clustering [20, 37], text mining [79, 58], image processing and analysis [86, 63], data distortion based privacy preservation [32, 71], etc. NMF is also applied in collaborative filtering to make product recommendations [87, 13].

A conventional NMF is defined as follows [45],

$$R_{m \times n} \approx U_{m \times k} \cdot V_{n \times k}^T \tag{4.4}$$

The goal is to find a pair of orthogonal nonnegative matrices U and V that minimize the Frobenius norm $\|R - UV^T\|_F$. Thus the objective function for NMF is

$$min_{U\geq 0, V\geq 0} f(R, U, V) = \|R - UV^T\|_F^2 \tag{4.5}$$

This chapter proposes an NMF based matrix factorization technique that takes into account weights and constraints. It is expected to preserve the data privacy by imputing and perturbing the values during its update process.

As stated in the previous chapters, one of the significant distinctions between collaborative filtering data and other data is the missing value issue. The rating matrices are usually very incomplete so they cannot be directly fed to the matrix factorization algorithms, such as SVD and NMF. Those missing values should be imputed properly during the pre-processing step. Existing imputation methods include random value imputation, mean value imputation [64], expectation maximization (EM) imputation [17, 82], linear regression imputation [73], etc. Nevertheless, all of them require extra time to compute the missing values. In contrast, the weighted NMF (WNMF) [87] can work with incomplete matrices without a separate imputation procedure.

Given a weight matrix $W \in \mathbb{R}^{m\times n}$ that indicates the existence of values in the rating matrix R (see Eq. (4.7)), the objective function of WNMF is

$$min_{U\geq 0, V\geq 0} f(R, W, U, V) = \|W \circ (R - UV^T)\|_F^2 \tag{4.6}$$

where $\circ$ denotes the element-wise multiplication.

$$w_{ij} = \begin{cases} 1 \quad if \quad r_{ij} \neq 0 \\ 0 \quad if \quad r_{ij} = 0 \end{cases} \quad (w_{ij} \in W, r_{ij} \in R) \tag{4.7}$$

When WNMF converges, $\tilde{R} = UV^T$ is the matrix with all missing entries filled. Since the residual exists, $\tilde{R}$ is different from R, making it a perturbed version of R. As discussed in Chapter 1, users do not want their privacy, e.g., their ratings left on particular items and on which items they have rated, to be released to other people. In WNMF, both of them are protected.

In [19], Ding et al. showed the equivalency between NMF and the K-Means clustering algorithm. When given a matrix R with objects as rows and attributes as columns, the two matrices U and V produced by NMF on R describe the cluster information of the objects. Each column vector of U, u_i, can be regarded as a basis and each data point r_i is approximated by a linear combination of these k bases, weighted by the components of V [50], where k is the rank of factor matrices. Thus the objects are grouped into clusters in accordance with matrix U.

However, in some cases, the data matrix R can represent relationships between two types of objects, e.g., user-item rating matrices in collaborating filtering applications and term-document matrices in text mining applications. It is expected that both row (user/term) clusters and column (item/document) clusters can be obtained by performing NMF on R. Due to the intrinsic property of NMF, it is very difficult to find two matrices U and V that represent user clusters and item clusters respectively at the same time. Hence, an extra factor matrix is needed to absorb the different scales of R, U, and V for simultaneous row clustering and column clustering [20]. Eq. (4.8) gives the objective function of the nonnegative matrix tri-factorization (NMTF).

$$min_{U\geq 0,S\geq 0,V\geq 0}f(R,U,S,V) = \|R - USV^T\|_F^2 \tag{4.8}$$

where $U \in \mathbb{R}_+^{m\times k}$, $S \in \mathbb{R}_+^{k\times l}$, and $V \in \mathbb{R}_+^{n\times l}$ (U and V are orthogonal matrices).

The use of S brings in a large scale of freedom for U and V so that they can focus on row and column clustering and preserve the privacy during the factorization process. In this scheme, both U and V are cluster membership indicator matrices while S is the coefficient matrix. Note that objects corresponding to rows in R are clustered into k groups and objects corresponding to columns are clustered into l groups.

With the auxiliary information of users and items, NMTF can be converted to a supervised learning procedure by applying cluster constraints to the objective function (4.8), giving the equation

$$\begin{aligned} &min_{U\geq 0,S\geq 0,V\geq 0} f(R,U,S,V,C_U,C_I) = \\ &\alpha \cdot \|R - USV^T\|_F^2 + \beta \cdot \|U - C_U\|_F^2 + \gamma \cdot \|V - C_I\|_F^2 \end{aligned} \tag{4.9}$$

where α, β, and γ are coefficients that control the weight of each part. C_U and C_I are user cluster matrix and item cluster matrix, respectively. They are obtained by running the K-Means clustering algorithm on user feature matrix F_U and item feature matrix F_I as mentioned in Section 4.1.

Combining Eqs. (4.6) and (4.9), the objective function for the weighted and constrained nonnegative matrix tri-factorization is developed as

$$\begin{aligned} &min_{U\geq 0,S\geq 0,V\geq 0} f(R,W,U,S,V,C_U,C_I) = \\ &\alpha \cdot \|W \circ (R - USV^T)\|_F^2 + \beta \cdot \|U - C_U\|_F^2 + \gamma \cdot \|V - C_I\|_F^2. \end{aligned} \tag{4.10}$$

This matrix factorization is named Aux-NMF, indicating that it incorporates the user and item auxiliary information into the factorization.

4.2.1.2 Update Formulas

In this section, the update formulas are derived for Aux-NMF.

Let $L = f(R,W,U,S,V,C_U,C_I)$, $X = \|W \circ (R - USV^T)\|_F^2$, $Y = \|U - C_U\|_F^2$, and $Z = \|V - C_I\|_F^2$.

Take derivatives of X with respect to U, S, and V:

$$\frac{\partial X}{\partial U} = -2(W \circ R)VS^T + 2W \circ (USV^T)VS^T \tag{4.11}$$

$$\frac{\partial X}{\partial S} = -2U^T(W \circ R)V + 2U^T[W \circ (USV^T)]V \tag{4.12}$$

$$\frac{\partial X}{\partial V} = -2(W \circ R)^T US + 2[W \circ (USV^T)]^T US \tag{4.13}$$

Take derivatives of Y with respect to U, S, and V:

$$\frac{\partial Y}{\partial U} = 2U - 2C_U, \quad \frac{\partial Y}{\partial S} = \frac{\partial Y}{\partial V} = 0 \tag{4.14}$$

Take derivatives of Z with respect to U, S, and V:

$$\frac{\partial Z}{\partial U} = \frac{\partial Z}{\partial S} = 0, \quad \frac{\partial Z}{\partial V} = 2V - 2C_I \tag{4.15}$$

Using Eqs. (4.11) to (4.15), we get the derivatives of L:

$$\begin{aligned}\frac{\partial L}{\partial U} &= 2\alpha[W \circ (USV^T)]VS^T + 2\beta U \\ &- 2\alpha(W \circ R)VS^T - 2\beta C_U\end{aligned} \tag{4.16}$$

$$\begin{aligned}\frac{\partial L}{\partial V} &= 2\alpha[W \circ (USV^T)]^T US + 2\gamma V \\ &- 2\alpha(W \circ R)^T US - 2\gamma C_I\end{aligned} \tag{4.17}$$

$$\begin{aligned}\frac{\partial L}{\partial S} &= 2\alpha U^T[W \circ (USV^T)]V \\ &- 2\alpha U^T(W \circ R)V\end{aligned} \tag{4.18}$$

To obtain the update formulas, the Karush-Kuhn-Tucker (KKT) complementary condition [43] is applied to the nonnegativities of U, S, and V. We have

$$\{2\alpha[W \circ (USV^T)]VS^T + 2\beta U - 2\alpha(W \circ R)VS^T - 2\beta C_U\}_{ij}U_{ij} = 0 \tag{4.19}$$

$$\{2\alpha[W \circ (USV^T)]^T US + 2\gamma V - 2\alpha(W \circ R)^T US - 2\gamma C_I\}_{ij}V_{ij} = 0 \tag{4.20}$$

$$\{2\alpha U^T[W \circ (USV^T)]V - 2\alpha U^T(W \circ R)V\}_{ij}S_{ij} = 0 \tag{4.21}$$

They give rise to the corresponding update formulas:

$$U_{ij} = U_{ij} \cdot \frac{\{\alpha(W \circ R)VS^T + \beta C_U\}_{ij}}{\{\alpha[W \circ (USV^T)]VS^T + \beta U\}_{ij}} \tag{4.22}$$

$$V_{ij} = V_{ij} \cdot \frac{\{\alpha(W \circ R)^T US + \gamma C_I\}_{ij}}{\{\alpha[W \circ (USV^T)]^T US + \gamma V\}_{ij}} \tag{4.23}$$

$$S_{ij} = S_{ij} \cdot \frac{\{U^T(W \circ R)V\}_{ij}}{\{U^T[W \circ (USV^T)]V\}_{ij}} \tag{4.24}$$

Assume $k, l \ll \min(m, n)$, the time complexities of updating U, V, and S in each iteration are all $O(mn(k + l))$. Therefore, the time complexity of Aux-NMF in each iteration is $O(mn(k + l))$.

4.2.1.3 Convergence Analysis

This section follows [45] to prove that the objective function L is nonincreasing under the update formulas (4.22), (4.23), and (4.24).

Definition 4.1. *$H(u, u')$ is an auxiliary function for $F(u)$ if the conditions*

$$H(u, u') \geq F(u), \quad H(u, u) = F(u) \tag{4.25}$$

are satisfied.

Lemma 4.1. *If H is an auxiliary function for F, then F is nonincreasing under the* *update*

$$u^{t+1} = \operatorname*{argmin}_{u} H(u, u^t) \tag{4.26}$$

Lemma 4.1 can be easily proved since we have $F(u^{t+1}) = H(u^{t+1}, u^{t+1}) \leq H(u^{t+1}, u^t) \leq H(u^t, u^t) = F(u^t)$.

The convergences of the update formulas (4.22), (4.23), and (4.24) will be proved by their equivalence to Eq. (4.26), with proper auxiliary functions defined.

Let us rewrite the objective function L,

$$\begin{aligned} L = & tr\{\alpha(W \circ R)^T \cdot (W \circ R)\} + tr\{-2\alpha(W \circ R)^T \cdot [W \circ (USV^T)]\} \\ & + tr\{\alpha[W \circ (USV^T)]^T \cdot [W \circ (USV^T)]\} \\ & + tr(\beta U^T U) + tr(-2\beta U^T C_U) + tr(\beta C_U^T C_U) \\ & + tr(\gamma V^T V) + tr(-2\gamma V^T C_I) + tr(\gamma C_I^T C_I) \end{aligned} \tag{4.27}$$

where $tr(*)$ is the trace of a matrix.

Eliminating the irrelevant terms, we can define the following functions that are only related to U, V, and S, respectively.

$$\begin{aligned} L(U) = & tr\{-2\alpha(W \circ R)^T \cdot [W \circ (USV^T)] + \alpha[W \circ (USV^T)]^T \cdot [W \circ (USV^T)] \\ & + \beta U^T U - 2\beta U^T C_U\} \\ = & tr\{-2[\alpha(W \circ R)VS^T + \beta C_U]U^T + U^T[\alpha W \circ (USV^T)VS^T] + U^T(\beta U)\} \end{aligned} \tag{4.28}$$

$$\begin{aligned} L(V) = & tr\{-2\alpha(W \circ R)^T \cdot [W \circ (USV^T)] + \alpha[W \circ (USV^T)]^T \cdot [W \circ (USV^T)] \\ & + \gamma V^T V - 2\gamma V^T C_I\} \\ = & tr\{-2[\alpha(W \circ R)^T US + \gamma C_I]V^T + V^T[\alpha(W \circ (USV^T))^T US] + V^T(\gamma V)\} \end{aligned} \tag{4.29}$$

$$
\begin{aligned}
L(S) = & \, tr\{-2\alpha(W \circ R)^T \cdot [W \circ (USV^T)] + \alpha[W \circ (USV^T)]^T \cdot [W \circ (USV^T)]\} \\
& = tr\{[-2\alpha U^T(W \circ R)V]S^T + [\alpha U^T(W \circ (USV^T))V]S^T\}
\end{aligned}
\tag{4.30}
$$

Lemma 4.2. *For any matrices $X \in \mathbb{R}_+^{n\times n}$, $Y \in \mathbb{R}_+^{k\times k}$, $F \in \mathbb{R}_+^{n\times k}$, $F' \in \mathbb{R}_+^{n\times k}$, and X, Y are symmetric, the following inequality holds*

$$
\sum_{i=1}^{n}\sum_{j=1}^{k} \frac{(XF'Y)_{ij}F_{ij}^2}{F'_{ij}} \geq tr(F^TXFY) \tag{4.31}
$$

The proof of Lemma 4.2 is presented in [20]. This lemma is used to build an auxiliary function for $L(U)$. Since $L(V)$ and $L(S)$ are similar to $L(U)$, their convergences are not necessary to be discussed.

Lemma 4.3.

$$
\begin{aligned}
H(U,U') = & -2\sum_{ij}\{[\alpha(W \circ R)VS^T + \beta C_U]U^T\}_{ij} \\
& + \sum_{ij} \frac{\{\alpha W \circ (U'SV^T)VS^T + \beta U'\}_{ij}U_{ij}^2}{U'_{ij}}
\end{aligned}
\tag{4.32}
$$

is an auxiliary function of $L(U)$ and the global minimum of $H(U,U')$ can be achieved by

$$
U_{ij} = U'_{ij} \cdot \frac{\{\alpha(W \circ R)VS^T + \beta C_U\}_{ij}}{\{\alpha[W \circ (U'SV^T)]VS^T + \beta U'\}_{ij}} \tag{4.33}
$$

Proof. We need to prove two conditions as specified in Definition 4.1. It is apparent that $H(U,U) = L(U)$. According to Lemma 4.2, we have

$$
\begin{aligned}
& \sum_{ij} \frac{\{\alpha W \circ (U'SV^T)VS^T + \beta U'\}_{ij}U_{ij}^2}{U'_{ij}} \\
& = \sum_{ij} \frac{\{\alpha W \circ (U'SV^T)VS^T\}_{ij}U_{ij}^2}{U'_{ij}} + \sum_{ij} \frac{\{\beta U'\}_{ij}U_{ij}^2}{U'_{ij}} \\
& \geq tr\{U^T[\alpha W \circ (USV^T)VS^T]\} + tr\{U^T(\beta U)\}.
\end{aligned}
\tag{4.34}
$$

Therefore, $H(U,U') \geq L(U)$. Thus $H(U,U')$ is an auxiliary function of $L(U)$.

To find the global minimum of $H(U,U')$ with U' fixed, we can take the derivative of $H(U,U')$ with respect to U_{ij} and let it be zero:

$$\begin{aligned}\frac{\partial H(U,U')}{\partial U_{ij}} &= \{-2[\alpha(W \circ R)VS^T + \beta C_U]\}_{ij} \\ &+ 2\frac{\{\alpha W \circ (U'SV^T)VS^T + \beta U'\}_{ij}U_{ij}}{U'_{ij}} = 0\end{aligned} \tag{4.35}$$

Solving for U_{ij}, we have

$$U_{ij} = U'_{ij} \cdot \frac{\{\alpha(W \circ R)VS^T + \beta C_U\}_{ij}}{\{\alpha[W \circ (U'SV^T)]VS^T + \beta U'\}_{ij}} \tag{4.36}$$

Since $F(U^0) = H(U^0,U^0) \geq H(U^1,U^0) \geq F(U^1) \geq ...$, $F(U)$ is monotonically decreasing and updating U by Eq. (4.36) can reach global minimum. ■

Similarly, the convergences of update formulas (4.24) and (4.23) can be proved as well.

4.2.1.4 Detailed Algorithm

This section presents the specific algorithm for Aux-NMF in collaborating filtering which is the basis of the incremental Aux-NMF.

Algorithm 4.1 depicts the procedure of performing Aux-NMF on a rating matrix.

Though Aux-NMF will eventually converge to a local minimum, it may take hundreds or even thousands of iterations. In this algorithm, an extra stop criterion, the maximum iteration count, is set to terminate the program at a reasonable point. In collaborative filtering applications, this value varies from $10 \sim 100$ and can generally produce satisfactory results.

4.2.2 iAux-NMF

As discussed in Section 4.1, new data can be regarded as new rows or new columns in the matrix. They are imputed and perturbed by the incremental Aux-NMF (iAux-NMF) with the aid of $U, S, V, C_U, C_I, Centroids_U$, and $Centroids_I$ generated by Algorithm 4.1.

iAux-NMF is technically the same as Aux-NMF, but focuses on a series of new rows or new columns. Hence, in this section, the incremental case of Aux-NMF is discussed by row update and column update separately.

4.2.2.1 Row Update

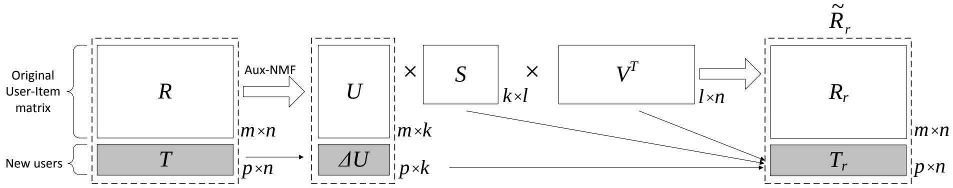


Figure 4.1: Updating new rows in iAux-NMF

In Eq. (3.1), it can be seen that $T \in \mathbb{R}^{p \times n}$ is added to R as a few rows. This process is illustrated in Figure 4.1. T should be imputed and perturbed before it can be released. Like Section 4.2.1.1, the objective function is developed by

$$\begin{aligned} &min_{\Delta U \geq 0} f(T, W_T, \Delta U, S, V, \Delta C_U) = \\ &\quad \alpha \cdot \|W_T \circ (T - \Delta U S V^T)\|_F^2 + \beta \cdot \|\Delta U - \Delta C_U\|_F^2 \end{aligned} \tag{4.37}$$

Accordingly, the update formula for this objective function is obtained as follows

$$\Delta U_{ij} = \Delta U_{ij} \cdot \frac{\{\alpha(W_T \circ T) V S^T + \beta \Delta C_U\}_{ij}}{\{\alpha[W_T \circ (\Delta U S V^T)] V S^T + \beta \Delta U\}_{ij}} \tag{4.38}$$

The convergence of Eq. (4.38) can be proved similarly as in Section 4.2.1.3. Since the row update only works on new rows, the time complexity of the algorithm in each iteration is $O(pn(l+k)+pkl)$. Assume $k, l \ll \min(p, n)$, the time complexity is then simplified to $O(pn(l+k))$.

Algorithm 4.2 illustrates the row update in iAux-NMF.

4.2.2.2 Column Update

The column update is almost identical to the row update. When the new data $G \in \mathbb{R}^{m \times q}$ arrives, it is updated by Algorithm 4.3. The time complexity for the column update is $O(qm(l+k))$.

The data owner should hold the updated factor matrices (U', S, and V') and the cluster information (the user/item cluster membership indicator matrices and the centroids) for future updates. Note that matrices S and V (S and U) are left unchanged in the row update (the column update), which does not indicate that they will never change. The experimental study will show when Aux-NMF should be recomputed to ensure the data utility and privacy.

4.3 Experimental Study

4.3.1 Data Description

In the experiments, the MovieLens [64], Sushi [35], and LibimSeTi [11] datasets were adopted as the test data. Table 4.1 collects the statistics of the datasets.

Table 4.1: Statistics of the data

Dataset	#users	#items	#ratings	Sparsity
MovieLens	943	1,682	100,000	93.7%
Sushi	5,000	100	50,000	90%
LibimSeTi	2,000	5,625	129,281	98.85%

The public MovieLens dataset has been described in Chapter 3. In addition to the rating data, user demographic information and item genre information are also available.

The Sushi dataset describes the user preferences on different kinds of sushi. There are 5,000 users and 100 sushi items. Each user has rated 10 items, with a rating ranging from 1 to 5. That is to say, there are 50,000 ratings in this dataset. To build the test set and the training set, for every user, 2 out of 10 ratings were randomly selected and were inserted into the test set (10,000 ratings) while the rest of ratings were used as the training set (40,000 ratings). Similar to MovieLens, the Sushi dataset comes with user demographic information as well as item group information and some attributes (e.g., the heaviness/oiliness in taste, how frequently the user eats the sushi etc.).

The LibimSeTi dating dataset was gathered by LibimSeTi.cz, an online dating website. It contains 17,359,346 anonymous ratings of 168,791 profiles made by 135,359 users as dumped on April 4, 2006. However, only the user's gender is provided with the data. Later sections will show how to resolve the problem with the lack of item information. Confined to the memory limit of the test computer, the experiments only used 2,000 users and 5,625 items[2] with 108,281 ratings in the training set and 21,000 ratings in the test set. Ratings are on a $1 \sim 10$ scale where 10 is best.

4.3.2 Data Pre-processing

The proposed algorithms require user and item feature matrices as the input. To build such feature matrices, the auxiliary information of users and items is pre-processed. In the MovieLens dataset, user demographic information includes user ID, age, gender, occupation, and zip code. Among them, age, gender, and occupation are utilized as features. For age, the numbers are categorized into 7 groups: 1-17, 18-24, 25-34, 35-44, 45-49, 50-55, >=56. For gender, there are two possible values: male and female. According to the statistics, there are 21 occupations: administrator, artist, doctor,

[2] User profiles are considered as items for this dataset.

and so on. Based on these possible values, a user feature matrix F_U was built with 30 features ($k_U = 30$). In other words, each user is represented as a row vector with 30 elements. An element is set to 1 if the corresponding feature value is true for this user and 0 otherwise. An example is, for a 48-year-old female user, who is an artist, the elements in the columns corresponding to female, 45-49, and artist should be set to 1. All other elements should be 0. Similar to the user feature matrix, the item feature matrix is built according to their genres. Movies in this dataset are attributed to 19 genres and hence the item feature matrix F_I has 19 features ($k_I = 19$) in it.

In the Sushi dataset, some of the user demographic information, e.g., gender and age, are used. In this case, user age has been divided into 6 groups by the data provider: 15-19, 20-29, 30-39, 40-49, 50-59, >=60. User gender consists of male and female, which is the same as MovieLens. Thus, the user feature matrix for this dataset has 5,000 rows and 8 columns. The item feature matrix, on the other hand, has 100 rows and 16 columns. The 16 features include 2 styles (maki and other), 2 major groups (seafood and other), and 12 minor groups (aomono (blue-skinned fish), akami (red-meat fish), shiromi (white-meat fish), tare (something like baste; for eel or sea eel), clam or shell, squid or octopus, shrimp or crab, roe, other seafood, egg, meat other than fish, and vegetables).

Different from the MoiveLens and Sushi datasets, the LibimSeTi dataset only provides user gender as its auxiliary information so it is directly used as the user cluster indicator matrix C_U. It is worth noting that in this dataset, there are three possible gender values: male, female, and unknown. To be consistent, the number of user clusters is set to 3.

4.3.3 Evaluation Strategy

For comparison purposes, the proposed approach and the SVD based data update approach presented in Chapter 3 were run on the datasets to measure the error of unknown value imputation and the privacy level of the perturbed data, as well as their time cost. The SVD based data update approach first uses the column mean to impute missing values in the new data and then performs the incremental SVD update on the imputed data. The machine that ran the experiments was equipped with Intel® Core™ i5-2405S processor, 8GB RAM and is installed with the UNIX operating system. The code was written and run in MATLAB.

When building the starting matrix R, the split ratio was used to decide how many ratings would be removed from the whole training data. For example, there are 1,000 users and 500 items with their ratings in the training data. If the split ratio is 40% and a row update will be done, the first 400 rows are considered as the starting matrix ($R \in \mathbb{R}^{400\times 500}$). The remaining 600 rows of the training matrix will be added to R in several rounds. Similarly, if a column update will be performed, the first 200 columns are considered as the starting matrix ($R \in \mathbb{R}^{1000\times 200}$) while the remaining 300 columns will be added to R in several rounds.

In each round, 100 rows/columns were added to the starting matrix. If the number of the rows/columns of new data is not divisible by 100, the last round will update the rest. Therefore, in this example, the remaining 600 rows will be added to R in 6 rounds with 100 rows each. It is worth mentioning that the Sushi data only has 100 items in total but the test of the column update was still expected on it so 10 items were added instead of 100 in each round.

The basic procedure of the experiments is as follows:

1. Perform Aux-NMF and SVD on R, producing the approximated matrix R_r (see Figure 4.1);

2. Append the new data to R_r by iAux-NMF and the SVD based data update algorithm which is proposed in Chapter 3, yielding the updated rating matrix $\tilde{R}_r$;

3. Measure the prediction error and the privacy level of the updated rating matrix $\tilde{R}_r$;

4. Compare and study the results.

In the experiments, the same measurements in Section 3.3.3 were adopted. However, unlike Chapter 3, there was no particular CF prediction model running on the released data $\tilde{R}_r$. Instead, the differences between the ratings in the test data and the released data were calculated.

Furthermore, the random variable Y in Definition 3.1 corresponds to only non-zero values in the training set while in Chapter 3, the same variable corresponds to all the values in the training set. X represents the perturbed values (at the same positions as those in the training set) in the released data.

4.3.4 Results and Discussion

This section presents and discusses the experimental results in two stages. Aux-NMF and SVD were first run on the whole training data to evaluate the performance of the non-incremental algorithm. Then the incremental algorithms were evaluated following the steps as specified in the previous section.

4.3.4.1 Test on Full Training Data

Some parameters of the proposed algorithms need to be determined in advance. Table 4.2 gives the parameter setup for Aux-NMF.

Table 4.2: Parameter setup for Aux-NMF

Dataset	α	β	γ	k	l	$MaxIter$
MovieLens	0.2	0	0.8	7	7	10
Sushi	0.4	0.6	0	7	5	10
LibimSeTi	1	0	0	3	10	10

For the MovieLens dataset, $\alpha = 0.2$, $\beta = 0$, and $\gamma = 0.8$, which means that the prediction relied mostly on the item cluster matrix, and then the rating matrix, whereas it eliminated the user cluster matrix. This combination was selected after probing many possible cases. Section 4.3.4.3 discusses how the parameters were chosen. It is highly possible that there exist better combinations. Both k and l were set to 7 because K-Means was prone to generate empty clusters with greater k and l, especially on the data with very few users or items. Note that if β or γ is a non-zero value, the user or item cluster matrix will be used and k or l is equal to the number of user clusters or item clusters. As long as β or γ is zero, the algorithm will eliminate the corresponding cluster matrix and k or l will be unrelated to the number of user clusters or item clusters.

For the Sushi dataset, $\alpha = 0.4$, $\beta = 0.6$, and $\gamma = 0$. The parameters indicate that the user cluster matrix plays the most critical role during the update process. In contrast, the rating matrix is the second important factor as it indicates the users' preferences on items. The item cluster matrix seemed trivial so it did not participate in the computation. k was set to 7 and l to 5 based on the same reason as mentioned in the previous paragraph.

For the LibimSeTi dataset, full weight was given to the rating matrix. The user and item cluster matrices received zero weight since they did not contribute anything to the positive results. As mentioned in the data description, users' auxiliary information only includes the genders with three possible values. So k was set to 3. In this case, l just denotes the column rank of V and was set to 10.

In SVD, since it cannot run on incomplete matrices, item mean was used to impute the missing values. The rank was set to 13 for MovieLens, 7 for Sushi, and 10 for LibimSeTi. Table 4.3 lists the results on the three datasets.

Table 4.3: Results on MovieLens dataset

Dataset	Method	MAE	$\Pi(Y\|X)$	Time Cost
MovieLens	Aux-NMF	0.7481	1.2948	0.9902s
	SVD	0.7769	1.2899	34.1341s
Sushi	Aux-NMF	0.9016	1.4588	0.5350s
	SVD	0.9492	1.4420	5.4175s
LibimSeTi	Aux-NMF	1.2311	1.0715	5.7962s
	SVD	1.2154	1.0537	390.2246s

In this table, the time cost of SVD includes the imputation time while the time cost of Aux-NMF includes the clustering time. For instance, on the MovieLens dataset, the imputation took 32.2918 seconds and SVD itself took 1.8423 seconds, for a total of 34.1341 seconds; the clustering time took 0.0212 seconds and Aux-NMF itself took 0.9690 seconds, for a total of 0.9902 seconds. One can see that Aux-NMF outperformed SVD in all aspects on all three datasets. It is apparent that the former is significantly more efficient than the latter. It saved 97% time on MovieLens, 90% time on Sushi, and 98% time on LibimSeTi. This is mainly because the SVD based algorithm needs to impute missing values, which is time consuming. However for Aux-NMF, it can directly work on incomplete matrices, though it needs to cluster beforehand, which is fast in general.

It is interesting to investigate the results of running the K-Means algorithm on the final matrix generated by Aux-NMF and the matrix generated by SVD. As shown in Figure 4.2(a), the MovieLens users with ratings produced by Aux-NMF were clustered into 7 groups with clear boundaries. The result is different for SVD - most users were grouped together and thus the clusters cannot be distinguished from each others. In both figures, the axes denote the ratings left by users on items. The results indicate that the ratings generated by Aux-NMF distributed more normally than those that were produced by SVD. Remember that the goal is to provide good imputation accuracy as well as high privacy level. In addition, the data should look like real world values. To this purpose, the ratings should be distributed normally, e.g., people may leave more 3 stars on a 1 ∼ 5 scale than 1 star and 5 stars. In this regard, Aux-NMF generated more reasonable data than SVD did.

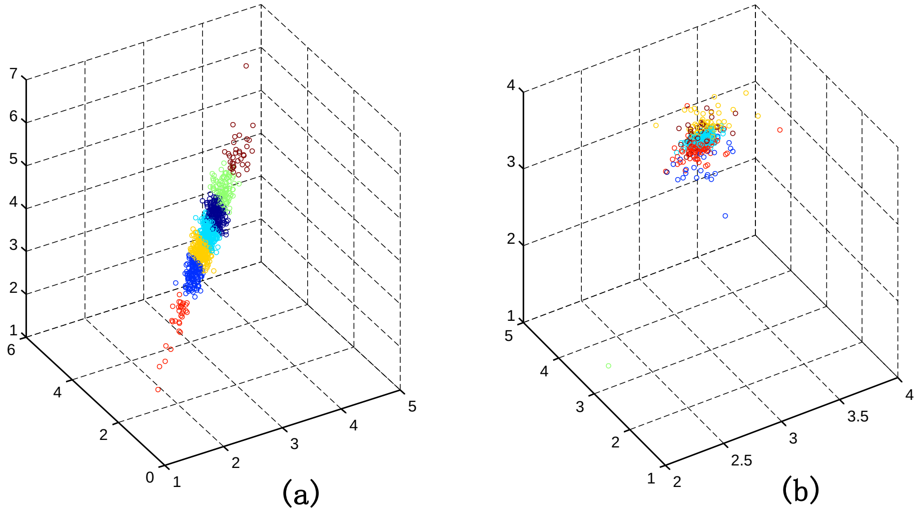


Figure 4.2: Clustering results on ratings predicted by Aux-NMF (a) and SVD (b) on MovieLens dataset

4.3.4.2 The Incremental Case

In the previous section, the experiments examined Aux-NMF on three datasets in terms of MAE, the privacy level, as well as the time cost. This section presents the same measurements on iAux-NMF.

Figure 4.3 shows the time cost for updating new rows and columns by iAux-NMF and the SVD based data update algorithm (SVDU).“RowN” and “ColumnN” are used to represent the row and column updates in iAux-NMF. Similarly, “RowS” and “ColumnS” are for the row and column updates in SVDU. The same parameter setup listed in Table 4.2 was adopted.

It can be seen that iAux-NMF outperformed SVDU in both row and column updates. As pointed out in Section 4.2.2, the time complexity of the row update in iAux-NMF is $O(pn(l+k))$ and the column update has a time complexity of $O(qm(l+k))$. As a reference, the time complexities of the row and column updates in SVDU are $O(k^3+(m+n)k^2+(m+n)kp+p^3)$ and $O(k^3+(m+n)k^2+(m+n)kq+q^3)$, respectively. When the rating matrix has high dimensionality, the time cost difference can be significant. For example, the LibimSeTi dataset has both more users and more items than MovieLens so the improvement of iAux-NMF over SVDU plotted in Figure 4.3(c) was greater than Figure 4.3(a). However, the Sushi data is a bit special as the time difference between the two methods in row updates was very small, though iAux-NMF was still faster. In Section 4.3.4.1, the time cost of both methods was split into two pieces. For SVDU, the cost consists of the imputation time and the SVD computation time. For Aux-NMF, the cost consists of the clustering time and the Aux-NMF computation time[3]. By tracking the time cost of each stage, it was found that the imputation in SVDU took considerably shorter time in the row update than the column update on this dataset but the time cost of Aux-NMF in the row update and the column update did not differ remarkably. Essentially, the faster imputation in the row update can be attributed to the small number of items. Since SVDU uses the column mean to impute the missing values, if there are only a few items, the mean value calculation will be fast.

However, with the substantial improvement in time cost, iAux-NMF should not produce a significantly higher imputation error than SVDU.

Figure 4.4 shows the mean absolute errors of the prediction. When the split ratio was greater than 20%, iAux-NMF achieved lower errors than SVDU on the MovieLens and Sushi datasets. The average improvement on MovieLens was 9.79% for row update and 9.76% for column update. The Sushi dataset had a little less average

[3] Before running the algorithms, the parameters need to be determined. The time cost for this part is discussed in Section 4.3.4.3.

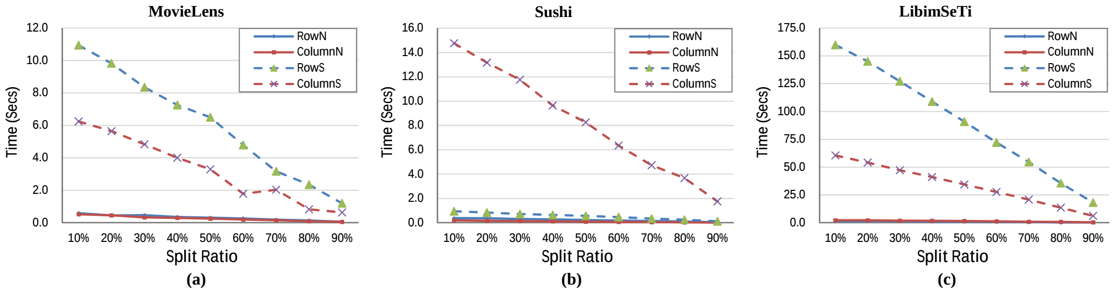


Figure 4.3: Time cost variation with split ratio

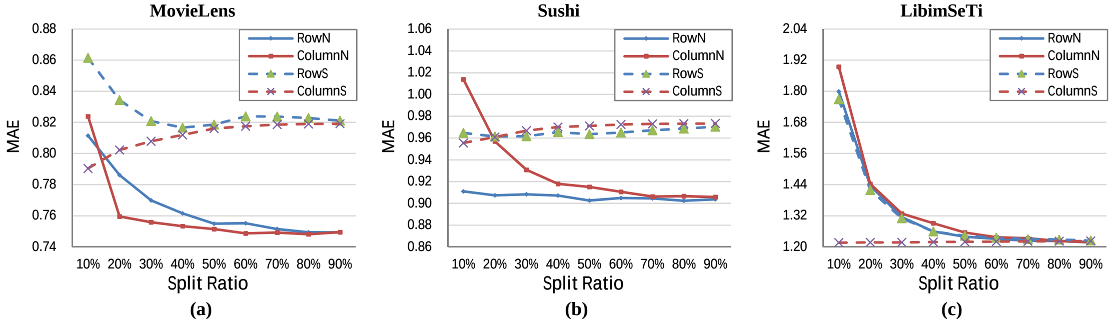


Figure 4.4: MAE variation with split ratio

improvement than MoiveLens but still noticeable. Nevertheless, both of them had greater errors by iAux-NMF than by SVD when the split ratio was less than 20%. This is because the centroids chosen by the K-Means algorithm did not distribute over the data that was adequately large enough to reflect the complete set. With not optimally selected centroids, K-Means cannot produce reasonable clustering results which further affect Aux-NMF and iAux-NMF, so the errors will be large. Unlike MovieLens and Sushi, the LibimSeTi dataset received different results. In this case, iAux-NMF still performed better than SVDU but the gap tended to be smaller as the split ratio increased. The results imply that the auxiliary information is important to iAux-NMF as it is used as the constraints in the update process. On the contrary, SVDU does not need it. This can explain why SVDU performed better than iAux-NMF on LibimSeTi since no auxiliary information was used in the update process.

In Section 4.2.2.2, the Aux-NMF recomputation issue is addressed. As presented in Figure 4.4, the MAE's of both the row and column updates on the MovieLens dataset decreased more slowly at 70% and leveled off after this point. Similarly, but more interestingly, the MAE of the row update on the Sushi dataset began to increase at 70%. Therefore, a recomputation can be performed at 70% for these two datasets. For the LibimSeTi dataset, the MAE's did not stop decreasing so the recomputation was not immediately necessary.

In addition to MAE, it is expected to investigate the privacy metrics presented in Section 4.3.3. The privacy levels with varying split ratios are plotted in Figure 4.5. The curve shows that the privacy levels of the data produced by iAux-NMF were higher and more stable than SVDU while the latter had decreasing trends with greater split ratios.

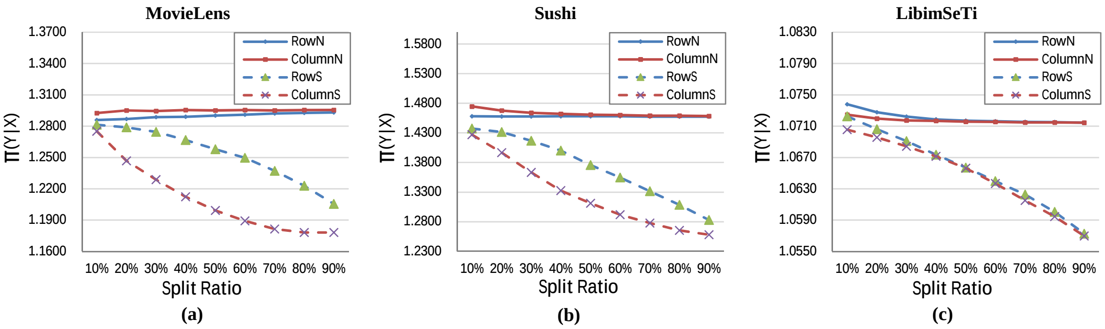


Figure 4.5: Privacy level variation with split ratio

As a summary, the iAux-NMF data update algorithm performed more efficiently than SVDU while maintaining nearly the same data utility and privacy, if not better.

4.3.4.3 Parameter Study

In iAux-NMF, three parameters, α, β, and γ need to be set. In this section, several parameter combinations are compared. Note that the split ratio was kept at 40% and the initial random matrices in Algorithms 4.2 and 4.3 were pre-generated to eliminate the effect of randomness in the experiments. The same parameter setup in Table 4.2 is adopted because it is the best combination obtained by probing multiple cases. The pseudocode in Algorithm 4.4 shows the procedure to find out the parameters that produce the lowest MAE's. The step is set to 0.1 when incrementing the parameters. Since there is a constraint $\alpha+\beta+\gamma=1$, the total number of parameter combinations is 66. It took 806.28 seconds to run a full test on the MovieLens dataset, 1,116.9 seconds on Sushi, and 11,517.87 seconds on LibimSeTi. The times are relatively long when compared with the times of running the incremental algorithms. However, the parameters only need to be determined offline once so this process does not affect the online performance.

Table 4.4 lists some representative combinations with their results on the MovieLens dataset. The best combinations are in bold font. It can be seen that if the updates simply rely on the rating matrix, the results are only a little worse than those that take into account the auxiliary information. In contrast, if only the auxiliary information is considered, the MAE is unacceptable, though the privacy level is the highest. It is clear that between user features and item features, the latter makes a positive contribution to the results while the former seems trivial. Nevertheless, the weight of the rating matrix can be lowered but should not be ignored. The Sushi dataset (Table 4.5) has a similar conclusion but it is the user features that played a more dominant role.

As shown in Table 4.6, the rating matrix of the LibimSeTi dataset is the only information used in the computation. This indicates that even the dataset comes with users' genders, they do not help in the proposed model. This is reasonable as the gender is not a necessary factor for people to determine their ratings, e.g., a female can rate another female with a fairly high rating. It is worth noting that since there is no item feature available in this dataset, γ was set to zero all the time.

Therefore, it can be concluded that the rating matrix should always be utilized while the auxiliary information makes contributions to the improved results as well.

Table 4.4: Parameter probe on MovieLens dataset

Parameters	Update	MAE	$\Pi(Y\|X)$
$\alpha = 1, \beta = 0, \gamma = 0$	Row	0.7643	1.2913
	Column	0.7538	1.2964
$\alpha = 0.5, \beta = 0.5, \gamma = 0$	Row	0.7643	1.2913
	Column	0.7539	1.2963
$\alpha = 0.5, \beta = 0, \gamma = 0.5$	Row	0.7624	1.2909
	Column	0.7534	1.2958
$\alpha = 0, \beta = 0.5, \gamma = 0.5$	Row	0.9235	1.3149
	Column	0.9164	1.3150
$\boldsymbol{\alpha = 0.2, \beta = 0, \gamma = 0.8}$	Row	0.7616	1.2890
$\boldsymbol{\alpha = 0.4, \beta = 0, \gamma = 0.6}$	Column	0.7533	1.2955

Table 4.5: Parameter probe on Sushi dataset

Parameters	Update	MAE	$\Pi(Y\|X)$
$\alpha = 1, \beta = 0, \gamma = 0$	Row	0.9083	1.4578
	Column	0.9221	1.4613
$\alpha = 0.5, \beta = 0.5, \gamma = 0$	Row	0.9073	1.4580
	Column	0.9201	1.4614
$\alpha = 0.5, \beta = 0, \gamma = 0.5$	Row	0.9085	1.4580
	Column	0.9221	1.4614
$\alpha = 0, \beta = 0.5, \gamma = 0.5$	Row	1.0468	1.4851
	Column	1.0371	1.4849
$\boldsymbol{\alpha = 0.4, \beta = 0.6, \gamma = 0}$	Row	0.9071	1.4580
$\boldsymbol{\alpha = 0.2, \beta = 0.8, \gamma = 0}$	Column	0.9180	1.4620

Table 4.6: Parameter probe on LibimSeTi dataset

Parameters	Update	MAE	$\Pi(Y\|X)$
$\boldsymbol{\alpha = 1, \beta = 0, \gamma = 0}$	Row	1.2589	1.0719
	Column	1.2911	1.0717
$\alpha = 0.5, \beta = 0.5, \gamma = 0$	Row	1.3378	1.0713
	Column	1.3926	1.0709
$\alpha = 0, \beta = 1, \gamma = 0$	Row	5.4017	1.0782
	Column	5.4017	1.0782

4.4 Summary

This chapter proposes an NMF based privacy-preserving data update approach for collaborative filtering purposes. This approach utilizes the auxiliary information to build the cluster membership indicator matrices for users and items. These matrices are regarded as the additional constraints in updating the weighted nonnegative matrix tri-factorization. The proposed approach, named iAux-NMF, can incorporate the incremental data into existing data quite efficiently while maintaining the high data utility and privacy. Furthermore, the inevitable missing value imputation issues in collaborative filtering is solved in a subtle manner by this approach without using any particular imputation methods. Experiments conducted on three different datasets demonstrate the superiority of iAux-NMF over the previously discussed SVD based data update method in the incremental data update.

Algorithm 4.1 Aux-NMF

```
Input:
    User-Item rating matrix: $R \in \mathbb{R}^{m \times n}$;
    User feature matrix: $F_U \in \mathbb{R}^{m \times k_U}$;
    Item feature matrix: $F_I \in \mathbb{R}^{n \times k_I}$;
    Column dimension of U: $k$;
    Column dimension of V: $l$;
    Coefficients in objective function: $\alpha$, $\beta$, and $\gamma$;
    Number of maximum iterations: $MaxIter$.
Output:
    Factor matrices: $U \in \mathbb{R}^{m \times k}$, $S \in \mathbb{R}^{k \times l}$, $V \in \mathbb{R}^{n \times l}$;
    User cluster membership indicator matrix: $C_U \in \mathbb{R}^{m \times k}$;
    Item cluster membership indicator matrix: $C_I \in \mathbb{R}^{n \times l}$;
    User cluster centroids: $Centroids_U$;
    Item cluster centroids: $Centroids_I$;

Cluster users into $k$ groups according to $F_U$ by K-Means algorithm $\rightarrow$ $C_U$, $Centroids_U$;
Cluster items into $l$ groups according to $F_I$ by K-Means algorithm $\rightarrow$ $C_I$, $Centroids_I$;
Initialize $U$, $S$, and $V$ with random values;
Build weight matrix $W$ by Eq. (4.7);
Set $iteration = 1$ and $stop = false$;
while ($iteration < MaxIter$ $and$ ($stop == false$) do
    $U_{ij} \leftarrow U_{ij} \cdot \frac{\{\alpha(W \circ R)VS^T + \beta C_U\}_{ij}}{\{\alpha[W \circ (USV^T)]VS^T + \beta U\}_{ij}}$;
    $V_{ij} \leftarrow V_{ij} \cdot \frac{\{\alpha(W \circ R)^T US + \gamma C_I\}_{ij}}{\{\alpha[W \circ (USV^T)]^T US + \gamma V\}_{ij}}$;
    $S_{ij} \leftarrow S_{ij} \cdot \frac{\{U^T(W \circ R)V\}_{ij}}{\{U^T[W \circ (USV^T)]V\}_{ij}}$;
    $L \leftarrow \alpha \cdot \|W \circ (R - USV^T)\|_F^2 + \beta \cdot \|U - C_U\|_F^2 + \gamma \cdot \|V - C_I\|_F^2$;
    if ($L$ increases in this iteration) then
        $stop = true$;
        Restore $U$, $S$, and $V$ to their values in last iteration.
    end if
    $iteration = iteration + 1$;
end while
Return $U, S, V, C_U, C_I, Centroids_U$, and $Centroids_I$.
```

Algorithm 4.2 iAux-NMF for Row Update

```
Input:
    New rating data: $T \in \mathbb{R}^{p\times n}$;
    New user feature matrix: $\Delta F_U \in \mathbb{R}^{p\times k_U}$;
    Coefficients in objective function: $\alpha$, $\beta$, and $\gamma$;
    Factor matrices: $U \in \mathbb{R}^{m\times k}$, $S \in \mathbb{R}^{k\times l}$, $V \in \mathbb{R}^{n\times l}$;
    User cluster membership indicator matrix: $C_U \in \mathbb{R}^{m\times k}$;
    User cluster centroids: $Centroids_U$;
    Number of maximum iterations: $MaxIter$.
Output:
    Updated factor matrix: $U' \in \mathbb{R}^{(m+p)\times k}$;
    Updated user cluster membership indicator matrix: $C'_U \in \mathbb{R}^{(m+p)\times k}$;
    Updated user cluster centroids: $Centroids'_U$;
    Imputed and perturbed new data: $T_r \in \mathbb{R}^{p\times n}$;

Cluster new users into $k$ groups based on $\Delta F_U$ and $Centroids_U$ by K-Means algorithm $\rightarrow \Delta C_U$, $Centroids'_U$;
Initialize $\Delta U \in \mathbb{R}^{p\times k}$ with random values;
Build weight matrix $W_T$ by Eq. (4.7);
Set $iteration = 1$ and $stop = false$;
while ($iteration < MaxIter$) and ($stop == false$) do
    $\Delta U_{ij} \leftarrow \Delta U_{ij} \cdot \frac{\{\alpha(W_T \circ T)VS^T + \beta\Delta C_U\}_{ij}}{\{\alpha[W_T \circ (\Delta USV^T)]VS^T + \beta\Delta U\}_{ij}}$
    $L \leftarrow \alpha \cdot \|W_T \circ (T - \Delta USV^T)\|_F^2 + \beta \cdot \|\Delta U - \Delta C_U\|_F^2$;
    if ($L$ increases in this iteration) then
        $stop = true$;
        Restore $U'$ to its value in last iteration.
    end if
    $iteration = iteration + 1$;
end while
Append $\Delta C_U$ to $C_U \rightarrow C'_U$;
Append $\Delta U$ to $U \rightarrow U'$;
Calculate $\Delta USV^T \rightarrow T_r$;
Return $U', C'_U, Centroids'_U$, and $T_r$.
```

Algorithm 4.3 iAux-NMF for Column Update

```
Input:
    New rating data: $G \in \mathbb{R}^{m \times q}$;
    New item feature matrix: $\Delta F_I \in \mathbb{R}^{q \times k_I}$;
    Coefficients in objective function: $\alpha$, $\beta$, and $\gamma$;
    Factor matrices: $U \in \mathbb{R}^{m \times k}$, $S \in \mathbb{R}^{k \times l}$, $V \in \mathbb{R}^{n \times l}$;
    Item cluster indicator membership matrix: $C_I \in \mathbb{R}^{n \times l}$;
    Item cluster centroids: $Centroids_I$;
    Number of maximum iterations: $MaxIter$.
Output:
    Updated factor matrix: $V' \in \mathbb{R}^{(n+q) \times l}$;
    Updated item cluster membership indicator matrix: $C'_I \in \mathbb{R}^{(n+q) \times l}$;
    Updated item cluster centroids: $Centroids'_I$;
    Imputed and perturbed new data: $G_r \in \mathbb{R}^{m \times q}$;

Cluster new items into $l$ groups based on $\Delta F_I$ and $Centroids_I$ by K-Means algorithm $\rightarrow \Delta C_I$, $Centroids'_I$;
Initialize $\Delta V \in \mathbb{R}^{q \times l}$ with random values;
Build weight matrix $W_G$ by Eq. (4.7);
Set $iteration = 1$ and $stop = false$;
while ($iteration < MaxIter$) and ($stop == false$) do
    $\Delta V_{ij} \leftarrow \Delta V_{ij} \cdot \frac{\{\alpha(W_G \circ G)^T US + \gamma \Delta C_I\}_{ij}}{\{\alpha[W_G \circ (US\Delta V^T)]^T US + \gamma \Delta V\}_{ij}}$
    $L \leftarrow \alpha \cdot \|W_G \circ (G - US\Delta V^T)\|_F^2 + \gamma \cdot \|\Delta V - \Delta C_I\|_F^2$;
    if ($L$ increases in this iteration) then
        $stop = true$;
        Restore $V'$ to its value in last iteration.
    end if
    $iteration = iteration + 1$;
end while
Append $\Delta C_I$ to $C_I \rightarrow C'_I$;
Append $\Delta V$ to $V \rightarrow V'$;
Calculate $US\Delta V^T \rightarrow G_r$;
Return $V', C'_I, Centroids'_V$, and $G_r$.
```

Algorithm 4.4 Pseudocode for Parameter Probing

```
for $\alpha = 0 : 0.1 : 1$ do
    for $\beta = 0 : 0.1 : 1 - \alpha$ do
        $\gamma = 1 - \alpha - \beta$.
        Run Aux-NMF and iAux-NMF on a dataset with parameter $\alpha$, $\beta$, and $\gamma$,
            saving the MAE's as well as $\alpha$, $\beta$, and $\gamma$ to the corresponding variables.
    end for
end for
Find out the lowest MAE and return the associated parameters.
```

5 Automated Dimension Determination for NMF Based Incremental Collaborative Filtering

In the previous chapter, an NMF based incremental data update scheme, named iAux-NMF, is introduced. It is well known that in NMF, the dimensions of the factor matrices have to be determined in advance. Moreover, data is growing fast; thus, in some cases, the dimensions need to be changed to reduce the approximation error. The recommender systems should be capable of updating new data in a timely manner without sacrificing the prediction accuracy.

This chapter proposes an NMF based data update approach with automated dimension determination for collaborative filtering purposes. The approach can determine the dimensions of the factor matrices and update them automatically. It exploits the nearest neighborhood based clustering algorithm to cluster users and items according to their auxiliary information, and uses the clusters as the constraints in NMF. The dimensions of the factor matrices are associated with the cluster quantities. When new data becomes available, the incremental clustering algorithm determines whether to increase the number of clusters or merge the existing clusters. Experiments on three different datasets (MovieLens, Sushi and LibimSeTi) were conducted to examine the proposed approach. The results show that this approach can update the data quickly and provide satisfactory prediction accuracy.

5.1 Using iCluster-NMF for Collaborative Filtering Data Updates

In this section, the improved version of iAux-NMF, named iCluster-NMF algorithm, is introduced. The new algorithm utilizes the same objective function as well as the update formulas of iAux-NMF but with different clustering mechanisms.

5.1.1 Clustering the Auxiliary Information

In Eq. (4.10), the cluster membership indicator matrices are used as the constraints to perform the supervised learning. This requires the auxiliary information to be clustered beforehand. In [68], Su et al. proposed a nearest neighborhood based incremental clustering algorithm that can directly work on categorical data. Thus, their algorithm was modified and integrated into iCluster-NMF as the fundamental clustering technique.

Algorithm 5.1 depicts the steps to build the initial clusters for the existing feature matrices F_U and F_I. It is worth mentioning that since this algorithm takes categorical data as the input, for each attribute, all possible values are stored in one column. For example, a user vector (a row in F_U) contains 3 attributes (columns): gender, age, and occupation. Each column has a different number of possible values, e.g., gender has two possible values: male and female. The same format applies to F_I.

5.1.2 Detailed Algorithm

The whole procedure of performing Cluster-NMF, the non-incremental version of iCluster-NMF, on a rating matrix is illustrated in Algorithm 5.2 which is generally the same as Algorithm 4.1.

5.1.3 iCluster-NMF

When new rows/columns are available, they are imputed by iCluster-NMF with the aid of U, S, V, C_U, and C_I generated by Algorithm 5.2.

Technically, iCluster-NMF is identical to Cluster-NMF, but focuses on a series of new rows or columns. Meanwhile, according to Algorithm 5.3, when new feature data ΔF_U and ΔF_I arrive, they need to be clustered into existing clusters, otherwise new clusters are created. Eq. (4.10) indicates the relationship between the dimensions of

U and C_U, V and C_I. This means that once the clusters are updated, NMF must be completely recomputed.

5.2 Experimental Study

The experiments in this chapter tested iAux-NMF and iCluster-NMF on the same datasets that were used in Chapter 4 for their time cost and prediction errors.

5.2.1 Data Pre-processing

Because iAux-NMF and iCluster-NMF require different feature data formats (numerical vs categorical), the data fed to them should be processed in different ways. While in Section 4.3.2, data pre-processing for iAux-NMF is already described, the data that would be used by iCluster-NMF has to be processed in another way. Nevertheless, the way to build the data for iAux-NMF is also discussed as a reference.

For the MovieLens dataset, since iCluster-NMF directly works on categorical data, the user feature matrix F_U was built with 3 attributes ($k_U = 3$). They correspond to gender (2 possible values), age (7 possible values), and occupation (21 possible values), respectively. In contrast, for iAux-NMF, the categories were converted to numbers since the K-Means algorithm can only work on numerical data. The user feature matrix F_U was built with 30 attributes ($k_U = 30$); each user was represented as a row vector with 30 elements. An element will be set to 1 if the corresponding attribute value is true for this user and 0 otherwise. Similar with the user feature matrix, the item feature matrix was built in terms of their genres. Movies in this dataset were attributed to 19 genres and hence the item feature matrix F_I has 6 attributes for iCluster-NMF ($k_I = 6$ as a single movie could have up to 6 genres) and 19 attributes for iAux-NMF ($k_I = 19$).

In the Sushi dataset, eight of the users' demographic attributes are used: gender, age, and city in which the user has lived the longest until age 15 (plus region and east/west). Additionally, the city (plus region and east/west) in which the user currently lives is also used. As described in Section 4.3.2, users' age has been categorized into six groups while users' gender has two possible values. There are 48 cities (Tokyo, Hiroshima, Osaka, etc.), 12 regions (Hokkaido, Tohoku, Hokuriku, etc.) and 2 possible east/west values (either the eastern or western part of Japan). Thus, the user feature matrix for iCluster-NMF on this dataset has 5,000 rows and 8 columns. Nevertheless, since there are too many possible values $(2+6+(48+12+2)\times 2 = 132$ values) for all attributes, only gender and age are used to build the user feature matrix for iAux-NMF. This makes the matrix have 5,000 rows and 8 columns (2 genders plus 6 age groups). The item feature matrix, on the other hand, has 100 rows and 3 columns for iCluster-NMF (16 columns for iAux-NMF) since there are 2 styles, 2 major groups, and 12 minor groups.

Since the LibimSeTi dataset only provides the user gender information, it is simply used as the user cluster indicator matrix C_U. To be consistent, the number of user clusters is set to 1 for iCluster-NMF and 3 for iAux-NMF.

5.2.2 Evaluation Strategy

To evaluate the algorithms, the error of unknown value prediction and the time cost are measured. Besides iCluster-NMF and iAux-NMF, a naive Cluster-NMF is exploited as the benchmark in the experiments for comparisons. The general idea of the naive Cluster-NMF is quite close to iCluster-NMF. The only difference is the way of updating the clusters. In iCluster-NMF, it uses Alogrthim 5.1 to build the initial clusters which are then updated by Algorithm 5.3. In contrast, the naive Cluster-NMF does not use incremental clustering but simply uses the idea of Alogrthim 5.1 to cluster the existing objects to the fixed number of clusters and re-cluster them (to the fixed number of clusters as well) when new data is available. In other words,

F'_U in Eq. (4.1) and F'_I in Eq. (4.3) are re-clustered every time there is an update on the data. This will significantly lower the performance of the algorithm but it theoretically produces the most accurate result among all.

In order to demonstrate how much improvement the proposed algorithms have achieved, they were compared to two SVD based collaborative filtering algorithms and the performance was evaluated. In [9], Brand proposed a recommender system that leveraged the probabilistic imputation to fill the missing values in the incomplete rating matrix and then used the incremental SVD to update the imputed rating matrix. This makes SVD work seamlessly for CF purposes. In the experiments, this algorithm is denoted as iSVD. The SVD based method that is proposed in Chapter 3 is similar to [9] but has additional processing steps to ensure privacy protection. Additionally, it uses the mean value imputation instead of the probabilistic imputation to remove missing values. It is denoted as pSVD. Note that neither of them considers auxiliary information so only the rating matrix is used.

The experiments measure the prediction errors and the time cost on three proposed algorithms as well as iSVD and pSVD. The prediction error is measured by MAE defined in Eq. (3.12).

The basic procedure of the experiments is as follows:

1. Perform Algorithm 5.1 and Algorithm 5.2 on R;
2. Append the new data to R by iCluster-NMF, iAux-NMF, and naive Cluster-NMF (nCluster-NMF for short), yielding the updated rating matrix $\tilde{R}_r$;
3. Measure the prediction error and the time cost of the updated rating matrix $\tilde{R}_r$;
4. Compare and study the results.

5.2.3 Results and Discussion

5.2.3.1 Parameter Setup

The parameters that have to be determined by iAux-NMF are listed in Table 4.2 in Chapter 4.

The iCluster-NMF and nCluster-NMF are in general the same as iAux-NMF but with different clustering approaches and NMF recomputation strategies. In Algorithm 5.1, the maximum number of clusters $maxK$, the initial radius threshold s, and the radius decreasing step d_s must be determined in advance. Table 5.1 gives the parameter setup for iCluster-NMF and nCluster-NMF. It is worth noting that the LibimSeTi dataset has $maxK = 3$ for user clusters and $maxK = 1$ for item clusters.

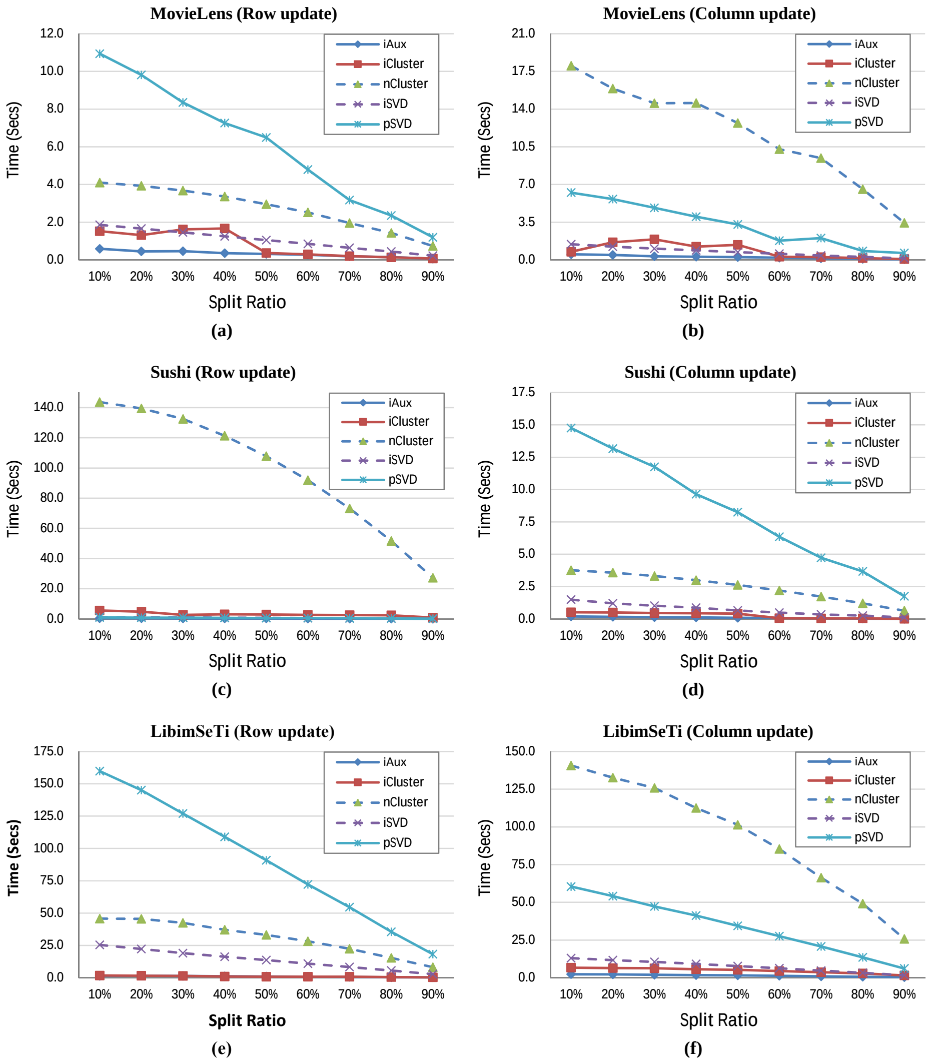
Table 5.1: Parameter setup for iCluster-NMF and nCluster-NMF

Dataset	$maxK$	s	d_s
MovieLens	10	1	0.1
Sushi	10	1	0.1
LibimSeTi	3/1	1	0.1

As far as iSVD and pSVD, the only parameter involved is the rank of the singular matrix. To determine this value, both algorithms were run for multiple times with different ranks. The numbers that achieved the optimal outcomes were selected. The best ranks for the MovieLens, the Sushi, and the LibimSeti datasets are 13, 7, and 10, respectively.

5.2.3.2 Experimental Results

Figure 5.1 shows the time cost for updating new rows and columns by iAux-NMF, iCluster-NMF, nCluster-NMF, as well as iSVD and pSVD. In most cases, nCluster-NMF and pSVD took significantly longer time than others. This is because nCluster-NMF was used to probe all possible cluster quantities to find out the choices that achieve the best MAE's. That is to say, it tries to cluster users into k groups and items into l groups, where $k, l = \{1, 2, ..., 10\}$, which results in 100 combinations. In addition, nCluster-NMF needs to re-cluster the whole data every time the new portion arrives. This requires even more time. As for pSVD, since it uses the mean value of each column to impute all missing values in that column, when a large amount of data is involved in the update (e.g. the row update on MovieLens and the column update on Sushi), the time cost can be high. The performance of iSVD is not as sensitive as pSVD to the data size but it also suffers from high matrix dimensionality, as shown in Figure 5.1(e).



(a)

(b)

(c)

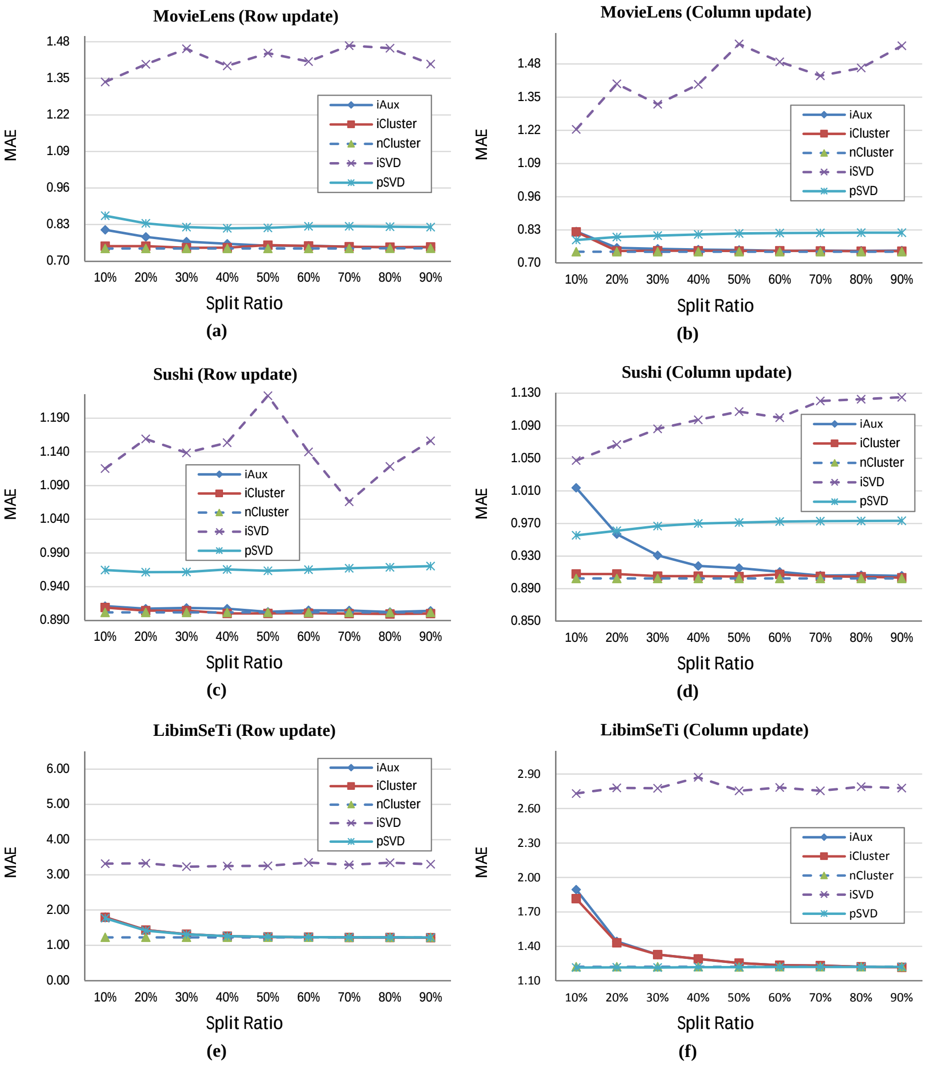
(d)

(e)

(f)

Figure 5.1: Time cost variation with split ratio

Comparing iAux-NMF and iCluster-NMF, it can be seen that their time costs were close in the process, though the former was slightly faster than the latter. This is because iCluster-NMF not only updates the clusters' content as iAux-NMF does, but also combines existing clusters or creates new clusters when necessary. The cluster update itself does not cost more time but since the number of clusters changes in some cases, the NMF has to be recomputed, which requires additional time.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.2: MAE variation with split ratio

As a reference, Table 5.2 lists the optimal number of clusters on the Sushi dataset. Note that the split ratio determines how many rows or columns should be present in

Table 5.2: Optimal number of clusters on Sushi dataset

Split Ratio		10%	20%	30%	40%	50%	60%	70%	80%	90%
iCluster (Row)	#UserClusters	10	10	10	10	10	10	9	10	8
	#ItemClusters	10	10	10	10	10	10	10	10	10
iCluster (Col)	#UserClusters	5	5	5	5	5	5	5	5	5
	#ItemClusters	5	5	5	5	4	4	7	9	10
nCluster (Row)	#UserClusters	7	7	7	7	7	7	7	7	7
	#ItemClusters	7	7	7	7	7	7	7	7	7
nCluster (Col)	#UserClusters	6	6	6	6	6	6	6	6	6
	#ItemClusters	10	10	10	10	10	10	10	10	10

the starting matrix. iCluster-NMF first runs Algorithm 5.1 on R to find the optimal number of clusters for users and items. Then they will be updated when new data is added to R. The numbers shown in this table are the final cluster quantities. When the rows were being updated, the model kept the columns unchanged and vice versa. This is why the number of item clusters remained the same when performing the row update and the number of user clusters remained the same when performing the column update. From the table, one can see that the best combinations obtained by nClsuter-NMF were 7 user clusters / 7 item clusters for the row update and 6 user clusters / 10 item clusters for the column update. Although the numbers are different from the ones obtained by iCluster-NMF, their MAE's are nearly the same.

The mean absolute errors of the predictions are plotted in Figure 5.2. iSVD performed worst on all datasets while nCluster-NMF reached the best results in most cases. Due to the way that nCluster-NMF works, the MAE's were consistently at the same level. They did not change significantly with varying split ratios. The only exception was the row update on the Sushi dataset, where iCluster-NMF achieved lower MAE than nCluster-NMF when the split ratio became higher. This to some extent means that updating the number of clusters in iCluster-NMF benefited the lower global prediction error. The figures show that iCluster-NMF outperformed iAux-NMF on all three datasets. It is interesting to look at the errors of pSVD, which were very close to iCluster-NMF on LibimSeTi but were worse on other datasets.

Remember that LibimSeTi only provides user gender information. In other words, the proposed models did not receive any extra helpful information from this dataset. Thus, its prediction accuracy was almost identical to pSVD's, which does not utilize auxiliary information at all.

The promising results can be attributed to not only the incremental clustering but also the recomputations of NMF. On one hand, clusters are updated when the new data comes in. This strategy ensures that the cluster membership indicator matrices C_U and C_I in Eq. (4.10) always maintain up-to-date relationships between either rows or columns. This, in turn, benefits the NMF update. On the other hand, due to the accumulated error in the incremental updates, NMF needs to be recomputed to maintain the accuracy. It is not convenient for the data owner to determine when to perform recomputations and update the dimensions of the factor matrices. In this situation, iCluster-NMF recomputes NMF when the number of clusters changes. It also explains why the MAE's of iAux-NMF and iCluster-NMF tend to be close when the split ratios become higher —since iAux-NMF does not recompute NMF, the more data it starts with, the less accumulated update error it has. Nevertheless, with more data available, the error will inevitably become larger.

Essentially, the iCluster-NMF data update algorithm produced higher prediction accuracy while costing just slightly more time, if not the same as iAux-NMF did. More importantly, the former does not require the data owner to determine the number of user and item clusters and can recompute NMF when necessary.Once useful auxiliary information became available, both algorithms outperformed the incremental SVD based algorithms with respect to the prediction accuracy.

5.3 Summary

In this chapter, an improved NMF based data update approach, named iCluster-NMF, is proposed. It can automatically determine the dimensions for NMF during the data update process. iCluster-NMF integrates the incremental clustering technique into the NMF based data update algorithm. This approach utilizes the auxiliary information to build the cluster membership indicator matrices of users and items. These matrices are regarded as the constraints in updating the weighted nonnegative matrix tri-factorization. iCluster-NMF does not require the data owner to determine when to recompute the NMF and the dimensions of the factor matrices. Instead, it sets the dimensions of the factor matrices according to the clustering result on users and items and updates them automatically. Experiments conducted on three different datasets demonstrate the high accuracy and performance of iCluster-NMF.

Algorithm 5.1 Initial Cluster Builder

```
Input:
    Object feature matrix: $D \in \mathbb{R}^{m \times f}$, where there are $m$ objects and $f$ attributes;
    Maximum number of clusters: $maxK$;
    Initial radius threshold: $s$;
    Radius decreasing step: $d_s$;
    Empty cluster collection: $CS$;
    Initial cluster feature: $CF$.
Output:
    Updated radius threshold: $s'$;
    Updated cluster collection: $CS'$;
    Updated cluster feature: $CF'$;

 1: Set $CS'$ to empty and $maxScore$ to 0;
 2: for $numK = 1$ to $maxK$ do
 3:   Reset $D$, $s$, $CS$, and $CF$;
 4:   while $D$ is not empty do
 5:     Read a new object $O$ from $D$;
 6:     if $CS$ is empty then
 7:       Create a cluster with $O$ and place it into $CS$;
 8:     else
 9:       Calculate the distance between $O$ and each cluster in $CS$ and find out the
          smallest distance $minDis_{oc}$;
10:       if $minDis_{oc} < s$ then
11:         Insert $O$ into the nearest cluster and update $CF$;
12:       else
13:         Create a cluster with $O$ and place it into $CS$;
14:       end if
15:     end if
16:     if $|CS| > numK$ then
17:       Calculate the distance between any two clusters and merge the two clusters
          with the minimum distance $minDis_{cc}$;
18:       if $minDis_{cc} > s$ then $s = minDis_{cc}$;
19:     end if
20:   end while
21:   if $|CS| < numK$ then $s = s - d_s$; Goto 3;
22:   Calculate the inter-cluster distance and inner-cluster distance to obtain the
      clustering score $lScore$.
23:   if $lScore > mScore$ then $mScore = lScore$; $CS' = CS$; $CF' = CF$; $s' = s$;
24: end for
```

Algorithm 5.2 Cluster-NMF

```
Input:
    User-Item rating matrix: $R \in \mathbb{R}^{m \times n}$;
    User feature matrix: $F_U \in \mathbb{R}^{m \times k_U}$;
    Item feature matrix: $F_I \in \mathbb{R}^{n \times k_I}$;
    Coefficients in objective function: $\alpha$, $\beta$, and $\gamma$;
    Number of maximum iterations: $MaxIter$.
Output:
    Factor matrices: $U \in \mathbb{R}^{m \times k}$, $S \in \mathbb{R}^{k \times l}$, $V \in \mathbb{R}^{n \times l}$;
    User cluster membership indicator matrix: $C_U \in \mathbb{R}^{m \times k}$;
    Item cluster membership indicator matrix: $C_I \in \mathbb{R}^{n \times l}$;

Cluster users based on $F_U$ by Algorithm 5.1 $\rightarrow C_U$;
Cluster items based on $F_I$ by Algorithm 5.1 $\rightarrow C_I$;
Initialize $U$, $S$, and $V$ with random values;
Build weight matrix $W$ by Eq. (4.7);
Set $iteration = 1$ and $stop = false$;
while $(iteration <= MaxIter)$ $and$ $(stop == false)$ do
    $U_{ij} \leftarrow U_{ij} \cdot \frac{\{\alpha(W \circ R)VS^T + \beta C_U\}_{ij}}{\{\alpha[W \circ (USV^T)]VS^T + \beta U\}_{ij}}$;
    $V_{ij} \leftarrow V_{ij} \cdot \frac{\{\alpha(W \circ R)^T US + \gamma C_I\}_{ij}}{\{\alpha[W \circ (USV^T)]^T US + \gamma V\}_{ij}}$;
    $S_{ij} \leftarrow S_{ij} \cdot \frac{\{U^T (W \circ R) V\}_{ij}}{\{U^T [W \circ (USV^T)] V\}_{ij}}$;
    $L \leftarrow \alpha \cdot \|W \circ (R - USV^T)\|_F^2 + \beta \cdot \|U - C_U\|_F^2 + \gamma \cdot \|V - C_I\|_F^2$;
    if ($L$ increases in this iteration) then
        $stop = true$;
        Restore $U$, $S$, and $V$ to their values in last iteration.
    end if
    $iteration = iteration + 1$;
end while
```

Algorithm 5.3 Incremental clustering algorithm

```
Input:
    Object feature matrix: $\Delta D \in \mathbb{R}^{m \times f}$, where there are $m$ objects and $f$ attributes;
    Maximum number of clusters: $maxK$;
    Radius threshold: $s'$;
    Cluster collection: $CS'$;
    Cluster feature: $CF'$.
Output:
    Updated radius threshold: $s''$;
    Updated cluster collection: $CS''$;
    Updated cluster feature: $CF''$;

while $\Delta D$ is not empty do
    Read a new object $O$ from $\Delta D$;
    Calculate the distance between $O$ and each cluster in $CS'$ and find out the smallest distance $minDis_{oc}$;
    if $minDis_{oc} < s'$ then
        Insert $O$ into the nearest cluster and update $CF'$;
    else
        Create a cluster with $O$ and place it into $CS'$;
    end if
    if $|CS'| > maxK$ then
        Calculate the distance between any two clusters and merge the two clusters with the minimum distance $minDis_{cc}$;
        if $minDis_{cc} > s'$ then $s'' = minDis_{cc}$;
    end if
end while
$CF'' = CF'$; $CS'' = CS'$; $s'' = s'$;
```

6 Trust-aware Privacy-Preserving Recommender System

It has been discussed in the previous chapters that almost all online shopping websites store users' sensitive information on their central servers, such as users' purchase history as well as their ratings and comments on products. When writing reviews, some websites allow users to choose publishing or saving them as drafts. If the reviews are not sensitive, they can be simply published by their authors. However, if users just want to save those comments for personal purposes and do not hope to share with others, they can keep the drafts unpublished. By doing so, people can ensure that these comments are only visible to themselves. With that said, people's published reviews are exposed to the public so anyone can read them freely. Though these comments might not be considered as customers' privacy, malicious users are able to identify customers' private preferences on particular products by cheating recommender systems. In a typical attack model, the attackers create fake user profiles containing real customers' public comments and then obtain recommendations from the system [48]. Due to the connections (either implicit or explicit) between users' public preferences and private preferences, if a recommender system fails to distinguish the real customers from the malicious users, it would be highly possible that the private preferences of the real customers can be exposed.

On some product review websites, such as Epinions.com and Ciao.com, in addition to leaving reviews on items, users can also tag trust votes based on their opinions on others' reviews [30]. In other words, every user maintains a list of the users that he trusts. Now an assumption is added to the aforementioned attack model - attackers are not easily trusted by other people because they only duplicate real customers' reviews and ratings and create the fake profiles in a relatively short time. This assumption is made because before a user tags his trusted users, he must have read many of their reviews in the past. If a newly created account has very similar or the same product reviews and ratings as his trusted users, people might suspect that this account is fake so they would not trust it.

This chapter proposes a privacy-preserving recommendation framework, named TrustRS, which preserves users' private preferences when providing accurate recommendations. The framework uses NMF based CF techniques that work on both users' rating information and their trustworthiness to provide personalized recommendations. The procedure is divided into two stages: unknown rating predictions and unrelated entries filtering. The raw ratings and trustworthiness information participate in the weighted nonnegative matrix tri-factorization [20] to approximate the original rating matrix. User and item groups are then established according to the factor matrices. The framework utilizes the group information to filter out unrelated entries so that a majority of real users' items of interest are concealed from the attackers. The experiments examined TrustRS on the Epinions [52] and Ciao datasets [69] in two aspects: (1) unknown rating prediction accuracy, and (2) user's privacy preservation level. The results show that the proposed framework can tell the real customers from the attackers to a great extent without compromising the prediction accuracy.

6.1 Problem Description

Assume the data owner has two matrices: a user-item rating matrix, denoted by $R \in \mathbb{R}^{m \times n}$, and a user-user trust matrix, denoted by $T \in \mathbb{R}^{m \times m}$, where there are m users and n items. An entry r_{ij} in R represents the rating left on item j by user i. The trust matrix T indicates the trust relationships among users. Possible values of the entry t_{ik} in T are 1, 0, and -1. $t_{ik} = 1$ means user i is trusted by user k, while $t_{ik} = -1$ means user i is distrusted by user k. If the trust relationship is unknown, then $t_{ik} = 0$.

As discussed in the previous chapters, since most users rate only a few items and most items are rated by a small amount of users, R is incomplete. The main task of recommender systems is to predict the unknown ratings in R, which are further used to make recommendations. The collaborative filtering based recommender systems predict ratings by using relevant users' preferences. In this scheme, if two users have very similar preferences, they are very likely to receive close or same recommendations. Malicious users can take this advantage to obtain real users' private interests, which are considered sensitive information. Definition 6.1 gives the formal description of the attack model.

Definition 6.1. *An* ***Attack model*** *is a 4-tuple:* $Attack(u, a) = \{R_u, R_a, L_u, L_a\}$*, where* u *is a real user and* a *is the attacker;* R_u *and* R_a *are their rating vectors;* L_u *and* L_a *are their recommendation lists*[1]*.*

To attack a recommender system, the attacker makes a fake profile with R_a *according to the real users' preferences* R_u *and receives the recommendations from the system. The closer* L_u *and* L_a *are, the more successful of an attack is achieved.*

As mentioned earlier, an attacker can easily collect real users' public preferences. However, it is not easy for the attackers to be trusted by the same people who trust the attackees. Therefore, in the proposed attack model, it is assumed that the attackers only possess attackees' partial ratings but are not in other people's trust lists.

The purpose of the proposed recommendation framework is to distinguish the real users from the attackers by differing their recommendation lists without degrading the accuracy of the unknown rating predictions.

[1] The recommendation list in this framework contains all relevant items to the active user. They might receive either low or high ratings from this user.

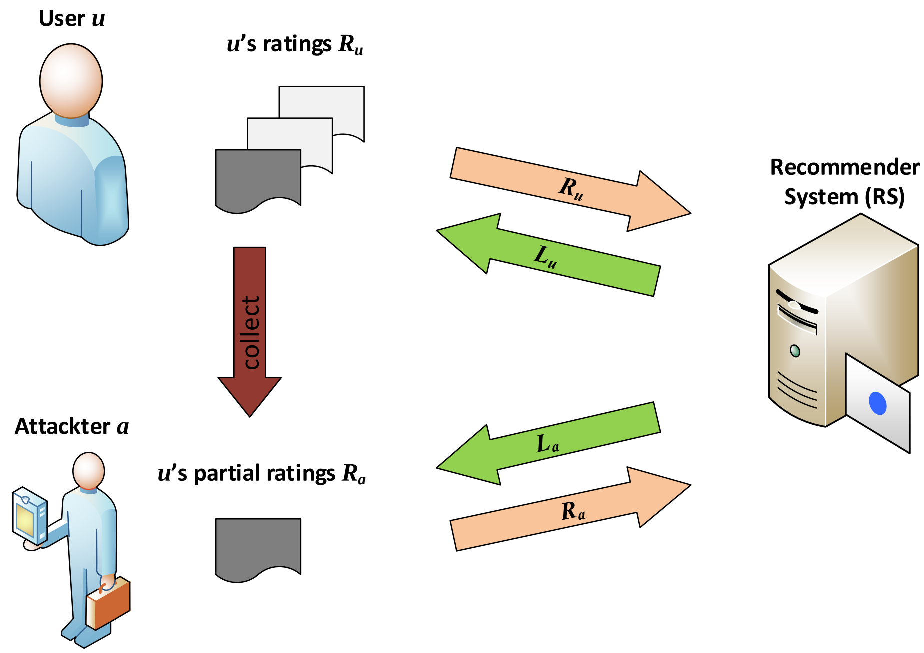


Figure 6.1: An attack model

6.2 Trust-aware Privacy-Preserving Recommendation Framework

This section introduces the proposed recommendation framework —TrustRS in two phases: unknown rating predictions and unrelated entries filtering.

6.2.1 Unknown Rating Predictions

In Chapter 4, the model uses the weighted and constrained nonnegative matrix tri-factorization for privacy-preserving data updates. In TrustRS, users' trustworthiness information is incorporated into a similar matrix factorization to approximate the original rating matrix. The proposed approach will be presented as follows: developing the objective function, deriving the update formulas, and analyzing the convergences.

6.2.1.1 Objective Function

Recall that in Chapter 4, a conventional NMF is defined in Eq. (4.4) and the goal is to find a pair of orthogonal nonnegative matrices U and V that minimize the Frobenius norm $\|R - UV^T\|_F$. Thus the objective function for NMF is

$$min_{U \geq 0, V \geq 0} f(R, U, V) = \|R - UV^T\|_F^2. \tag{6.1}$$

It is also discussed that R must not contain missing values as NMF is not suitable for such matrices. If all missing values in R are replaced with zeros and NMF is performed on the imputed rating matrix, it will lead to very biased results. Moreover, the trustworthiness is expected to be incorporated into the matrix factorization, and the users as well as items are expected to be grouped in terms of the factor matrices. Therefore, additional constraints need to be applied to NMF. To do so, the trust-incorporated NMF as illustrated in Eq. (6.2) is proposed.

$$\begin{aligned} & min_{U \geq 0, S \geq 0, V \geq 0, B \geq 0} f(R, W_R, T, W_T, U, S, V, B) = \alpha \cdot \|W_R \circ (R - USV^T)\|_F^2 \\ & + \beta \cdot \|W_T \circ (T - UB^T)\|_F^2 + \gamma \cdot (\|U\|_F^2 + \|S\|_F^2 + \|V\|_F^2 + \|B\|_F^2). \end{aligned} \tag{6.2}$$

where $U \in \mathbb{R}_+^{m \times k}$, $S \in \mathbb{R}_+^{k \times l}$, $V \in \mathbb{R}_+^{n \times l}$, and $B \in \mathbb{R}_+^{m \times k}$. U, V, B are orthogonal matrices. $\circ$ denotes the element-wise multiplication.

In this equation, R is a rating matrix and T is a trust matrix. Since both R and T are incomplete, two weight matrices W_R and W_T are introduced to resolve the missing value issues [87]. They indicate the locations of the observed entries in R and T as

$$w_{Rij} = \begin{cases} 1 \quad if \quad r_{ij} \neq 0 \\ 0 \quad if \quad r_{ij} = 0 \end{cases} (w_{Rij} \in W_R, r_{ij} \in R) \tag{6.3}$$

$$w_{Tij} = \begin{cases} 1 \quad if \quad t_{ij} \neq 0 \\ 0 \quad if \quad t_{ij} = 0 \end{cases} \quad (w_{Tij} \in W_T, t_{ij} \in T) \tag{6.4}$$

While the first two terms focus on optimizing R and T, the third term of this equation is adopted to avoid overfitting. Coefficients α, β, and γ are used to control the weight of each term.

6.2.1.2 Update Formulas

This section follows the derivation procedure in Section 4.2.1.1 to derive the update formulas to minimize the objective function in Eq. (6.2).

Let $L = f(R, W_R, T, W_T, U, S, V, B)$, and take the derivatives of L with respect to U, S, V, and B:

$$\begin{aligned} \frac{\partial L}{\partial U} &= 2\alpha W_R \circ (USV^T)VS^T - 2\alpha(W_R \circ R)VS^T \\ &+ 2\beta[W_T \circ (UB^T)]B - 2\beta(W_T \circ T)B + 2\gamma U \end{aligned} \tag{6.5}$$

$$\frac{\partial L}{\partial S} = 2\alpha U^T[W_R \circ (USV^T)]V - 2\alpha U^T(W_R \circ R)V + 2\gamma S \tag{6.6}$$

$$\frac{\partial L}{\partial V} = 2\alpha[W_R \circ (USV^T)]^T US - 2\alpha(W_R \circ R)^T US + 2\gamma V \tag{6.7}$$

$$\frac{\partial L}{\partial B} = 2\beta[W_T \circ (UB^T)]^T U - 2\beta(W_T \circ T)^T U + 2\gamma B \tag{6.8}$$

The corresponding update formulas are:

$$U_{ij} = U_{ij} \cdot \frac{\{\alpha(W_R \circ R)VS^T + \beta(W_T \circ T)B\}_{ij}}{\{\alpha[W_R \circ (USV^T)]VS^T + \beta[W_T \circ (UB^T)]B + \gamma U\}_{ij}} \tag{6.9}$$

$$S_{ij} = S_{ij} \cdot \frac{\{\alpha U^T (W_R \circ R) V\}_{ij}}{\{\alpha U^T [W_R \circ (USV^T)] V + \gamma S\}_{ij}} \tag{6.10}$$

$$V_{ij} = V_{ij} \cdot \frac{\{\alpha (W_R \circ R)^T US\}_{ij}}{\{\alpha [W_R \circ (USV^T)]^T US + \gamma V\}_{ij}} \tag{6.11}$$

$$B_{ij} = B_{ij} \cdot \frac{\{\beta (W_T \circ T)^T U\}_{ij}}{\{\beta [W_T \circ (UB^T)]^T U + \gamma B\}_{ij}} \tag{6.12}$$

Assuming $k, l \ll \min(m, n)$, the time complexities of each update is $O(mn(k + l) + (m^2 + n^2)k)$.

6.2.1.3 Convergence Analysis

The proof of convergence of the derived update formulas is similar to Section 4.2.1.3. The convergences of the update formulas (6.9), (6.10), (6.11), and (6.12) will be proved by showing that they are equivalent to Eq. (4.26) in Chapter 4, with proper auxiliary functions defined.

Let us rewrite the objective function L,

$$\begin{aligned} L = {} & tr\{\alpha (W_R \circ R)^T \cdot (W_R \circ R)\} + tr\{-2\alpha (W_R \circ R)^T \cdot [W_R \circ (USV^T)]\} \\ & + tr\{\alpha [W_R \circ (USV^T)]^T \cdot [W_R \circ (USV^T)]\} + tr[\beta (W_T \circ T)^T (W_T \circ T)] \\ & + tr\{-2\beta (W_T \circ T)^T \cdot [W_T \circ (UB^T)]\} \\ & + tr\{\beta [W_T \circ (UB^T)]^T \cdot [W_T \circ (UB^T)]\} \\ & + tr(\gamma U^T U) + tr(\gamma S^T S) + tr(\gamma V^T V) + tr(\gamma B^T B) \end{aligned} \tag{6.13}$$

where $tr(*)$ is the trace of a matrix.

Eliminating the irrelevant terms, we define the following functions that are only related to U, V, S, and B, respectively.

$$
\begin{aligned}
&L(U) = \\
&\quad tr\{-2\alpha(W_R \circ R)^T \cdot [W_R \circ (USV^T)] + \alpha[W_R \circ (USV^T)]^T \cdot [W_R \circ (USV^T)] \\
&- 2\beta(W_T \circ T)^T \cdot [W_T \circ (UB^T)] + \beta[W_T \circ (UB^T)]^T \cdot [W_T \circ (UB^T)] + \gamma U^T U\} \\
&= tr\{-2[\alpha(W_R \circ R)VS^T + \beta(W_T \circ T)B]U^T + U^T[\gamma U + \alpha W_R \circ (USV^T)VS^T \\
&+ \beta[W_T \circ (UB^T)]B]\}
\end{aligned}
\tag{6.14}
$$

$$
\begin{aligned}
&L(S) = tr\{-2\alpha(W_R \circ R)^T \cdot [W_R \circ (USV^T)] + \gamma S^T S \\
&+ \alpha[W_R \circ (USV^T)]^T \cdot [W_R \circ (USV^T)]\} \\
&= tr\{-2[\alpha U^T(W_R \circ R)V]S^T + S^T[\gamma S + \alpha U^T[W_R \circ (USV^T)]V]\}
\end{aligned}
\tag{6.15}
$$

$$
\begin{aligned}
&L(V) = tr\{-2\alpha(W_R \circ R)^T \cdot [W_R \circ (USV^T)] + \gamma V^T V \\
&+ \alpha[W_R \circ (USV^T)]^T \cdot [W_R \circ (USV^T)]\} \\
&= tr\{-2[\alpha(W_R \circ R)^T US]V^T + V^T[\gamma V + \alpha[W_R \circ (USV^T)]^T US]\}
\end{aligned}
\tag{6.16}
$$

$$
\begin{aligned}
&L(B) = tr\{-2\beta(W_T \circ T)^T \cdot [W_T \circ (UB^T)] + \gamma B^T B \\
&\quad + \beta[W_T \circ (UB^T)]^T \cdot [W_T \circ (UB^T)]\} \\
&\quad = tr\{-2[\beta(W_T \circ T)^T U]B^T + B^T[\gamma B + \beta[W_T \circ (UB^T)]^T U]\}
\end{aligned}
\tag{6.17}
$$

Lemma 4.2 is used to build an auxiliary function for $L(U)$. Since $L(U)$ is similar to $L(V)$ and $L(S)$, their convergences will not be discussed.

Lemma 6.1.

$$\begin{aligned} H(U,U') = &-2\sum_{ij}\{[\alpha(W_R\circ R)VS^T+\beta(W_T\circ T)B]U^T\}_{ij} \\ &+\sum_{ij}\frac{\{\gamma U'+\alpha W_R\circ(U'SV^T)VS^T+\beta[W_T\circ(U'B^T)]B\}_{ij}U_{ij}^2}{U'_{ij}} \end{aligned} \tag{6.18}$$

is an auxiliary function of $L(U)$ *and the global minimum of* $H(U,U')$ *can be achieved by*

$$U_{ij} = U'_{ij}\cdot\frac{\{\alpha(W_R\circ R)VS^T+\beta(W_T\circ T)B\}_{ij}}{\{\gamma U+\alpha W_R\circ(USV^T)VS^T+\beta[W_T\circ(UB^T)]B\}_{ij}} \tag{6.19}$$

Proof. We need to prove two conditions as specified in Definition 4.1 in Chapter 4. It is apparent that $H(U,U)=L(U)$. According to Lemma 4.2, we have

$$\begin{aligned} &\sum_{ij}\frac{\{\gamma U'+\alpha W_R\circ(U'SV^T)VS^T+\beta[W_T\circ(U'B^T)]B\}_{ij}U_{ij}^2}{U'_{ij}} \\ &=\sum_{ij}\frac{\{\gamma U'\}_{ij}U_{ij}^2}{U'_{ij}}+\sum_{ij}\frac{\{\alpha W_R\circ(U'SV^T)VS^T\}_{ij}U_{ij}^2}{U'_{ij}} \\ &\quad+\sum_{ij}\frac{\{\beta[W_T\circ(U'B^T)]B\}_{ij}U_{ij}^2}{U'_{ij}} \\ &\geq tr\{\gamma U\}+tr\{\alpha W_R\circ(USV^T)VS^T\}+tr\{\beta[W_T\circ(UB^T)]B\}. \end{aligned} \tag{6.20}$$

Therefore, $H(U,U')\geq L(U)$ and $H(U,U')$ is an auxiliary function of $L(U)$.

To find the global minimum of $H(U,U')$ with U' fixed, we can take the derivative of $H(U,U')$ with respect to U_{ij} and let it be zero:

$$\begin{aligned} \frac{\partial H(U,U')}{\partial U_{ij}} &= \{-2[\alpha(W_R\circ R)VS^T+\beta(W_T\circ T)B]\}_{ij} \\ &+2\frac{\{\gamma U+\alpha W_R\circ(USV^T)VS^T+\beta[W_T\circ(UB^T)]B\}_{ij}U_{ij}}{U'_{ij}} \\ &=0 \end{aligned} \tag{6.21}$$

Solving for U_{ij}, we have

$$U_{ij} = U'_{ij} \cdot \frac{\{\alpha(W_R \circ R)VS^T + \beta(W_T \circ T)B\}_{ij}}{\{\gamma U + \alpha W_R \circ (USV^T)VS^T + \beta[W_T \circ (UB^T)]B\}_{ij}} \tag{6.22}$$

Since $F(U^0) = H(U^0, U^0) \geq H(U^1, U^0) \geq F(U^1) \geq ...$, $F(U)$ is monotonically decreasing and updating U by Eq. (6.22) can reach global minimum. ■

Similarly, the convergences of update formulas (6.10), (6.11), and (6.12) can be proved as well.

6.2.2 Unrelated Entries Filtering

In the previous section, an NMF model that considers rating as well as trustworthiness is developed. By updating the formulas, the objective function's global minimum can be achieved. Thus $\hat{R} = USV^T$ is the approximated original rating matrix with all unknown entries filled. To make personalized and privacy-preserving recommendations, TrustRS filters unrelated ratings before it generates the recommendation lists.

To this purpose, users and items are grouped based on U and V in Eq. (6.2) so that only relevant items would be presented to the users. The following definitions define the user group and the interest group that would be used next.

Definition 6.1. *A* ***user group*** $g_{Up} = \{T_{Up}, D_{Up}\}$*, where* T_{Up} *is the set of users that belong to this group;* D_{Up} *contains the membership degree of each user in* T_{Up}*.*

Definition 6.2. *An* ***interest group*** $g_{Iq} = \{T_{Iq}, M_{Iq}, g'_{Uq}\}$*, where* T_{Iq} *is the set of items that belong to this group;* M_{Iq} *contains the average rating of each item in* T_{Iq}*;* g'_{Uq} *is the group of users (with their membership degrees) who have rated at least one item in* T_{Iq}*.*

Interest groups, denoted by $G_I = \{g_{I1}, g_{I2}, ..., g_{Il}\}$, can be established in terms of matrix V according to the equivalence between NMF and the K-Means clustering algorithm [19]. For a given $V \in \mathbb{R}_{+}^{n \times l}$, items are clustered into l groups by Eq. (6.23).

$$c_i = \underset{1 \leq j \leq l}{\operatorname{argmax}} V_{ij} \tag{6.23}$$

where c_i is the index of the group that item i belongs to.

For each item group T_{Iq}, the average rating of every item is calculated and the related users are also identified. In this case, a single user may belong to multiple groups. This way of user grouping is called the interest based grouping. The membership degree of user u to interest group g can be calculated by Eq. (6.24).

$$d_{ug} = (1 - dis_{ug}) \cdot wa_{ug} \cdot wb_{ug} \tag{6.24}$$

where dis_{ug} is the normalized distance between user u's ratings and the corresponding average ratings in M_{Iq} of interest group g; wa_{ug} is the fraction of user u's ratings left on items in g; wb_{ug} is the fraction of items in g that are rated by user u. wa and wb are used to weight $(1 - dis_{ug})$ so the greater they are, the higher d_{ug} will be.

In addition to the interest based grouping, users are also clustered by matrix U like the way that items are clustered by matrix V. Note that $U \in \mathbb{R}_{+}^{m \times k}$ is updated by taking into account both rating matrix R and trust matrix T. Hence the groups obtained in this way, denoted by $G_T = \{g_{U1}, g_{U2}, ..., g_{Uk}\}$, reflect users' rating patterns as well as their trustworthiness. This grouping strategy is named the trust based grouping.

Algorithm 6.1 illustrates the procedures to recommend items to a user u. The general idea of this algorithm is to recommend user u the items that are rated by u's neighbors. The corresponding ratings are predicted by the NMF model proposed in the previous section.

Algorithm 6.1 Rating filtering algorithm

```
Input:
    Raw rating matrix: $R$;
    Imputed rating matrix: $\hat{R} \in \mathbb{R}^{m \times n}$;
    User groups: $G_T = \{g_{U1}, g_{U2}, ..., g_{Uk}\}$;
    Interest groups: $G_I = \{g_{I1}, g_{I2}, ..., g_{Il}\}$;
    User index: $u$;
    Membership degree threshold: $m_u$;
Output:
    Recommendation list: $L_u$;

$max_{degree} = 0$, $g_{I_{closest}} = \varnothing$, $NL_I, NL_T = \varnothing$;
for $i = 1$ to $l$ do
    if $d_{ui} \geq m_u$ and $d_{ui} > max_{degree}$ then
        $max_{degree} = d_{ui}$;
        $g_{I_{closest}} = g_{Ii}$;
    end if
end for
Add users in $g_{I_{closest}}$ with membership degree $\geq m_u$ to $u$'s interest neighbor list $NL_I$;
Search in $G_T$ and identify the user group $g_{U_u}$ that $u$ belongs to;
Add all users in $g_{U_u}$ to $u$'s trust neighbor list $NL_T$;
Take the intersection of $NL_I$ and $NL_T$. The result is saved in $NL_u$;
Search in $R$ to find items that are rated by users in $NL_u$. Save them in $IL_u$;
$L_u \leftarrow (j, \hat{R}_{uj})$, where $j \in IL_u$.
```

6.3 Experimental Study

6.3.1 Data Description

The experiments adopted the Epinions [52] and Ciao [69] datasets to examine TrustRS. Both provide rating and trustworthiness information. Table 6.1 collects the statistics of the datasets.

Table 6.1: Statistics of the data

Dataset	#users	#items	#ratings	#trusts
Ciao	2,056	1,458	53,312	36,432
Epinions	7,260	2,440	172,497	49,248

The raw Epinions dataset has 22,166 users and 296,227 items with 912,441 ratings and 355,217 trust values. Due to the memory limit, 7,260 users (who have rated at least 10 items) and 2,440 items with 172,497 ratings and 49,248 trust values were selected for testing. The ratings were divided into two subsets: a training set and a test set. To build the test set, users with more than 100 ratings and 20 trust values were chosen. For each of these users, 20 ratings were randomly selected and added into the test set. The rest of the ratings were added into the training set. Consequently, there are 171,577 ratings in the training set and 920 ratings in the test set.

Similarly, only partial ratings were selected from the Ciao dataset for the experiments. In this case, there are 2,056 users and 1,458 items with 53,312 ratings and 36,432 trust values. The test set contains 2,960 ratings from users who have at least 60 ratings and 20 trust values. The remaining ratings form the training set.

It is worth noting that both datasets provide the helpfulness of the ratings. For example, a user left 5 stars on an item and another user thought the helpfulness of this rating was 7 out of 10. This additional "rating of a rating" makes users' feedback more reliable. Therefore, in the experiments, all the ratings were preprocessed by considering their helpfulness. In this example, the user's rating becomes $original\ rating \times helpfulness\ degree = 5 \times (7/10) = 3.5$ as opposed to 5 accordingly.

For each dataset, some of the test users were chosen and 30% of their ratings were used to create the attackers[2]. For instance, if a test user has 100 ratings in the training set and 20 ratings in the test set, 30 of his training ratings are used to create an attacker profile. The test ratings of this attacker are identical to the real user's, meaning that the attacker has 20 ratings in the test set as well. Moreover, attackers are not trusted by anyone, so they do not have trust values.

[2] In this chapter's experiments, the active users include both test users and attackers.

6.3.2 Evaluation Strategy

The experiments tested TrustRS in two aspects: unknown rating predictions and user privacy preservation. The error measurement of the experiments was MAE.

To obtain the privacy preservation level,the recall rates derived from Eq. (6.25) were calculated for both the real users and the corresponding attackers. The experiments then measured how much the recall rate is reduced. More specifically, higher recall rates for the real users and lower recall rates for the attackers are expected because fewer items that are related to the real users would be recommended to the attackers. Therefore the higher privacy preservation level is achieved.

$$Recall_u = \frac{|L_u \cap T_u|}{|T_u|} \tag{6.25}$$

where L_u is the set of items that are recommended to user u; T_u is the set of items that appear in the test set and are rated by user u.

The proposed recommender system was compared against three existing algorithms that are discussed in Section 1.2: the SVD based CF with random perturbation [60] (referred to as RandSVD), the NMF based CF with random perturbation [47] (referred to as RandNMF), and the naive Baysian classifier based PPCF with pre-processing [8] (referred to as PRNBC).

Since RandSVD and RandNMF focus on perturbing rating matrix instead of refining the recommendation list, no ratings are filtered out, and the recall rates will always be 1. Thus the experiments only compared them with TrustRS in unknown rating predictions. PRNBC, in contrast, has a filtering step so both prediction accuracy and privacy preservation level were investigated on this algorithm.

6.3.3 Results and Discussion

In this section, the experimental results with respect to privacy preservation level and the prediction accuracy are studied.

6.3.3.1 Privacy Preservation

As mentioned in Section 1.2, PRNBC applies a pre-processing step to the naive Baysian classifier based PPCF to filter out the less important neighbors. By doing so, the online performance of PPCF is improved. A side effect is that the number of recommended items can be controlled – greater number of neighbors result in more recommended items. This facilitates the comparisons because different neighbor sizes can be probed for PRNBC to obtain the real users' recall rates that are very close to the recall rates produced by TrustRS. Then how much the recall rate is reduced from the real users to the attackers on both methods can be measured.

Note that the authors in [8] did not study MAE of PRNBC but converted the integer ratings to binaries. Whereas in this chapter's experiments, it is expected to examine both MAE's and recall rates so this conversion is not carried out. However, more classes (e.g., 5 classes in a 1-5 rating system, and 2 classes in a binary rating system) require significantly more computation time. To compensate for it, the one-group scheme was used as opposed to the multi-group scheme.

Table 6.2: Parameter setup for TrustRS

Dataset	α	β	γ	k	l	m_u
Ciao	0.1	0.3	0.6	20	13	0.0005
Epinions	0.1	0.7	0.2	20	20	0.0017

Table 6.2 lists the parameter setup for TrustRS. α, β, and γ are the coefficients that control the weight of each term in Eq. (6.2). k and l are utilized as the dimensions of the factor matrices as well as the number of user/item clusters. m_u is the membership degree threshold required by Algorithm 6.1.

The parameters were determined by probing different combinations. The experiments first fixed k, l, and m_u to test α, β, and γ in $\{0.1, 0.2, ..., 0.9\}$ under the constraint $\alpha + \beta + \gamma = 1$. Then k and l were probed in $\{1, 2, ..., 20\}$ with everything else unchanged. m_u was obtained last. The parameters were selected by taking into account both MAE's and the reductions of the recall rates.

In this table, β is greater than α on both datasets. It means that the trust matrices play a more important role than the rating matrices. Two k's and one l are 20, which is the largest number that were probed so it is very likely that better results might exist if giving larger numbers. Nonetheless, the more clusters that are produced, the longer computation time will be required.

The privacy protection levels, measured by the recall rate reduction percentages, are illustrated in Table 6.3. "RecallU" represents the average recall rate for real users, and "RecallA" represents that of the attackers. It can be seen that TrustRS has greatly reduced the recall rates from the real users to the attackers. For example, the Ciao dataset has 2,960 ratings in its test set. TrustRS recommended $2,960 \times 0.9297 \approx 2,751$ items to the real users but only $2,960 \times 0.4595 \approx 1,360$ items to the attackers. It protected half of the real users' relevant items from being exposed to the attackers. PRNBC, on the contrary, merely reduced the recall rates. The reduction percentages of TrustRS on the Epinions dataset are not as high as those on Ciao, but it still outperformed PRNBC to a great extent.

Table 6.3: Privacy protection level

Dataset	Method	RecallU	RecallA	%Reduced
Ciao	PRNBC	0.9295	0.9228	0.007%
	TrustRS	0.9297	0.4595	50.581%
Epinions	PRNBC	0.8886	0.8815	0.008%
	TrustRS	0.8870	0.5935	33.088%

The recall rates with varying membership degree threshold m_u were also recorded. Since a user might belong to multiple interest groups, this value affects the results in two aspects: (1) it determines which groups are considered as the active users' interest groups; (2) it controls the number of similar users that would be recognized as the active users' interest neighbors. A smaller m_u causes higher recall rates and vice versa. With that said, m_u should not be extremely small because the filtering step will fail to distinguish the real users from the attackers thus no privacy can be preserved. Figure 6.2 plots the recall rates on two datasets, in which $m_u \in \{0.0001, 0.0002, ..., 0.01\}$. With the increasing m_u, both recall rates decreased but they tended to be closer when m_u was very small or very large.

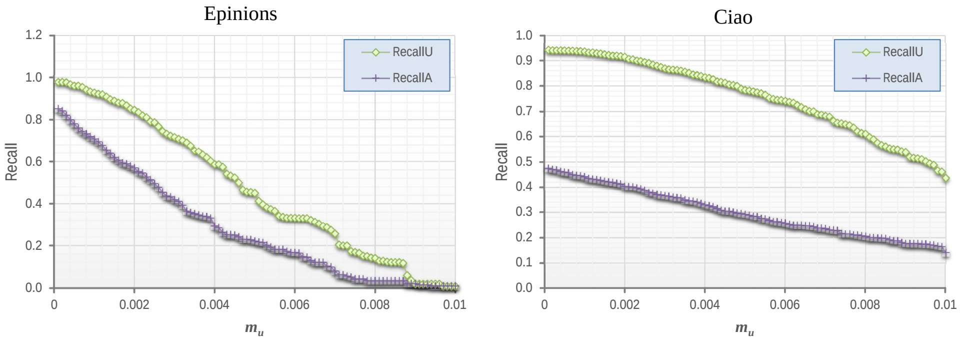


Figure 6.2: Recall rates with varying m_u in TrustRS

It is depicted in Algorithm 6.1 that TrustRS filters the unrelated items based on user groups G_T and interest groups G_I. While m_u directly affects the selection of a user's interest neighbors, there is no extra constraint on the formation of his trust neighbors. That is to say, all users in the same group as an active user are saved in his trust neighbor list NL_T. In Eq. (6.2), U is optimized for both R and T. Therefore, the user groups established from U are highly related to customers' rating patterns and trustworthiness, which are consequential to identifying malicious users. To investigate the impact of NL_T on the final recall rates, each active user's list was sorted according to the membership degrees of the users in the list, and the top n neighbors were kept for filtering.

Figure 6.3 charts the recall rates with multiple n's. The curves in this figure indicate that when just a few trust neighbors were used, TrustRS was not able to tell the real users from the attackers. With more neighbors' participation, the recall rates increasingly differed but kept stable after some points. The results imply the importance of the trustworthiness.

As a reference, the recall rates for PRNBC with different numbers of neighbors were also studied in Figure 6.4. When more neighbors were retained, the recall rates increased subsequently. Nevertheless, the algorithm failed to tell the real users from the attackers as their recall rates were almost identical.

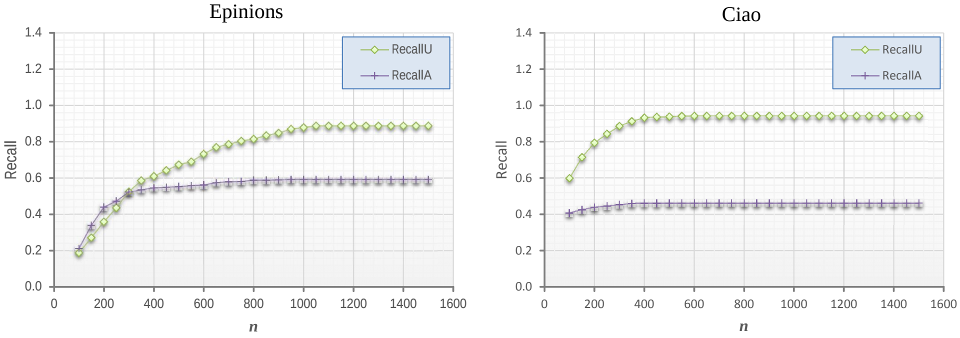


Figure 6.3: Recall rates with varying *#neighbors* in TrustRS

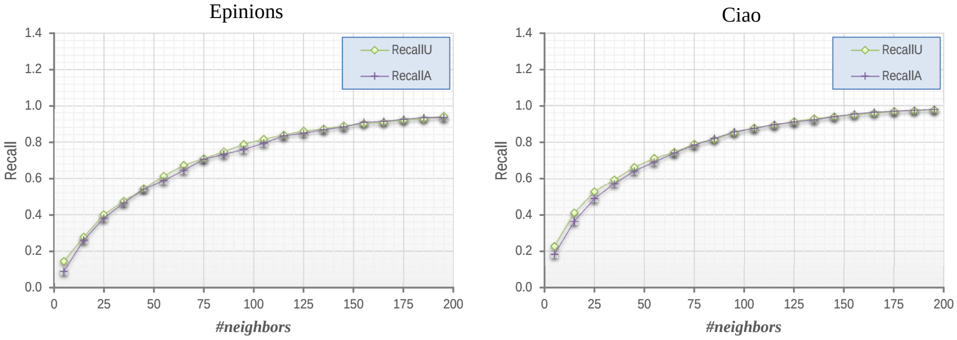


Figure 6.4: Recall rates with varying *#neighbors* in PRNBC

6.3.3.2 Prediction Accuracy

The prediction accuracy was judged in terms of MAE on all the ratings in the test set. Thus the second step of TrustRS, which is unrelated entries filtering, was ignored. The experiments used the same parameter setup presented in the previous section.

For both RandSVD and RandNMF, the experiments calculated the z-scores for each user and replaced the values in the original rating matrix R with z-scores. The missing entries were imputed by the users' mean ratings. The noise drawn from the normal distribution was added to the imputed matrix for perturbation purposes.

Table 6.4: Prediction accuracy

Dataset	Method	MAE	
Ciao	RandSVD	0.4124	$rank = 4$
	RandNMF	0.4162	$rank = 1$
	PRNBC	1.7851	-
	TrustRS	0.2725	-
Epinions	RandSVD	0.5362	$rank = 16$
	RandNMF	0.5480	$rank = 1$
	PRNBC	1.0202	-
	TrustRS	0.4084	-

The MAE's of all four methods are shown in Table 6.4. It is worth noting that the rank of RandSVD and RandNMF were probed in $\{1, 2, ..., 20\}$ and the lowest MAE's were taken. As one can see, TrustRS produced the lowest prediction errors on both datasets. RandSVD and RandNMF had very close values which were not significantly worse than TrustRS. This result is considered as normal owing to the inherent connections between SVD and NMF: they both focus on reducing the dimensionality of rating matrices and finding the latent factors. TrustRS is based on NMF but it uses weight matrices to avoid biased missing value imputation and incorporates trustworthiness to improve the prediction accuracy. Thus TrustRS achieved lower MAE's than other two matrix factorization based algorithms. PRNBC, the naive Bayesian classifier based recommendation algorithm, performed worst and the prediction errors were remarkably higher than others. It is attributed to the randomized response techniques (RRT) used in the one-group scheme [34]. Because in this scheme, each user's ratings are placed in the same group. According to the RRT, they either remain unchanged or are completely reversed (e.g., change 1 to 5, 2 to 4). It perturbs the ratings to a certain degree and protects the privacy, but the prediction errors would increase as well.

It can be concluded that the proposed recommendation framework TrustRS outperformed three existing privacy-preserving recommendation algorithms with regard to unknown rating prediction accuracy and privacy preservation level. The results demonstrate that the trustworthiness information makes a substantial contribution to correctly filtering out unrelated ratings. The membership degree threshold needs to be carefully determined as it directly affects the recall rates for both the real users and the attackers.

6.4 Summary

This chapter studies an attack model that aims at finding customers' potentially interested items by cheating recommender systems. A weighted nonnegative matrix tri-factorization based privacy-preserving recommendation framework, named TrustRS, is proposed to neutralize this type of attack. TrustRS utilizes customers' trustworthiness to filter out unrelated ratings so that their privacy can be preserved. Experiments conducted on two popular rating datasets show that TrustRS can preserve users' privacy to a great extent without compromising the prediction accuracy.

7 Conclusions and Future Work

This dissertation presents the research work in privacy-preserving algorithms for the collaborative filtering based recommender systems. This work involves the empirical study of classical recommendation algorithms, singular value decomposition based privacy-preserving data updates, incorporating auxiliary information into the NMF based collaborative filtering, the incremental clustering based data updates, and a trust-aware privacy-preserving recommender system. This chapter summarizes the dissertation work and proposes some future research topics.

7.1 Research Accomplishments

Over the past 20 years, the Internet has served as the major technology connecting our world. Economists have discovered the great potential that lies in that piece of technology. They have tried and are still trying to find suitable ways to make it as easy and pleasant as possible to spend money while surfing the Internet. Almost every shop now has an online presence that makes it possible to search, compare, and buy specific items or groups of items without the necessity of personal presence. To sell their products better, most online shopping websites provide recommendations to the customers who have visited their websites in the past. It is well known that recommender systems have achieved great success in providing product recommendations for online shopping. With recommender systems, customers can find their desired merchandise in a timely manner. It not only facilitates customers' purchases, but also promotes the sales. While recommender systems can predict customers' preferences accurately, they suffer from privacy leakage in many aspects.

This dissertation addresses several topics in collaborative filtering based recommender systems. In general, it can be divided into three parts: (1) Empirical study on classical recommendation algorithms; (2) Matrix factorization based recommender systems that preserve user privacy in data sharing; (3) A trust-aware recommender system that can tell the normal users from the attackers who attempt to cheat recommender systems.

Empirical Study on Classical Recommendation Algorithms

The empirical study in Chapter 2 presents several classical recommendation algorithms and studies the experiments on the clicking history data from a retargeting company. The predictions produced by different models have varied accuracies. It demonstrates that when selecting recommendation algorithms, people should carefully examine the data to identify what kinds of algorithms (e.g., user/item correlation based models, latent factor based models, and genetic algorithm based models) might be suitable. It also suggests that multiple methods can be combined to provide optimal predictions.

SVD based Privacy-Preserving Data Updates

Chapter 3 proposes an SVD based privacy-preserving data update scheme that makes use of the incremental SVD update technique to update the fast growing collaborative filtering data and preserve user privacy during data sharing. It protects the privacy by performing the truncated SVD on the original rating matrix with randomization and post-processing techniques. It also takes into account the missing value imputation during the update process to provide high quality data for accurate recommendations. Results of the experiments conducted on the MovieLens and Jester datasets show that the proposed scheme can handle data growth efficiently and keep a low level of privacy loss. The prediction accuracy is still at a satisfactory level compared to most published results.

Incorporating Auxiliary Inforamtion into NMF based Data Update Scheme

Performance analysis shows that the time complexity of the scheme proposed in Chapter 3 contains a cubic term with respect to the number of new rows or columns. It is a potentially expensive factor in the update process, especially when a large amount of new data comes in. Therefore, a better technique is needed to improve the update process so that it can be done faster. Chapter 4 uses NMF instead of SVD as the fundamental matrix factorization technique to reduce the dimensionality of the rating matrix and impute the missing values. By selecting reasonable iteration counts and the factor matrix dimensions in NMF updates, the scheme can be very efficient. Furthermore, the auxiliary information of users and items, e.g., user demographic information and item category information, is considered additional constraints in the NMF objective function. This behavior brings new knowledge into the update process and the experimental results indicate that it has improved both prediction accuracy and the privacy level.

Automated Dimension Determination for NMF based Incremental Collaborative Filtering

As a prerequisite, the dimensionality of the factor matrices in NMF has to be determined in advance. Moreover, data is growing fast. Thus in some cases, the dimensions need to be changed to reduce the approximation error. Recommender systems should be capable of updating new data in a timely manner without sacrificing the prediction accuracy. However, the data update scheme proposed in Chapter 4 does not consider these issues. In Chapter 5, an incremental nearest neighborhood based clustering algorithm is exploited to automatically determine and update the dimensions of the factor matrices in NMF updates. Experiments on three different datasets were conducted to examine the proposed approach. The results show that this approach can update the data quickly and provide encouraging prediction accuracy without needing predetermined matrix dimensions.

Trust-aware Privacy-Preserving Recommender System

In Chapters 3, 4, and 5, the privacy issues that are studied occur when two data owners are sharing/transferring their customer preference information. In real world scenarios, there are also quite a few other privacy issues on the Internet. Chapter 6 addresses an attack model in which an attacker holds some of the real customers' ratings and attempts to obtain their private preferences by cheating recommender systems. Due to the connections between users' public preferences and private preferences, if a recommender system fails to distinguish the real customers from the malicious users, it would be highly possible that the real customers' private preferences can be exposed. To neutralize this problem, a trust-aware privacy-preserving recommender system is proposed in this chapter. The system makes use of the trustworthiness information in online social networks to detect attackers and makes reasonably differed recommendations to the normal users and the attackers. The results demonstrate that this recommender system can distinguish between the real customers and the attackers to a great extent without compromising the prediction accuracy.

7.2 Suggestions for Future Work

In the future, it would be interesting to investigate more details of the collaborative filtering problem together with the privacy issue. In general, the following three topics would be studied:

(1) Utilization of the temporal information;

(2) Handling data growth with privacy preservation in distributed scenarios;

(3) Large scale recommender systems.

Privacy-Preserving Data Updates with Temperal CF

"There is nothing permanent except change.", said by Heraclitus (540-480BC), the Greek philosopher. In other words, the only thing that does not change is change itself. This is the case for people's shopping habits. For example, a person who was initially interested in digital SLR cameras a few years ago is now interested in cars. People cannot predict what will be attractive to this person in the near future. Therefore, his product preferences vary from time to time. The conventional collaborative filtering techniques may not work properly for him because most of the techniques assume that users' shopping patterns do not change. To make better recommendations, the corresponding temporal information should be fully utilized as it reveals the time-evolving trends of both user shopping patterns and item popularity.

While there are some temporal CF algorithms proposed in the past[21, 41, 78, 61, 36, 54, 1], they did not address the privacy issues that arise in this case. As it is stated in Chapter 3, users' privacy includes the exact ratings of a user left on particular items and on which items that this user has rated. When time factor is considered, this privacy information is extended to a further step: at what time this user rated which item with what rating. This motivation needs a conversion on the problem space from 2-D to 3-D which requires a tensor structure to process the data. In the future, nonnegative tensor factorization (NTF) would be used to handle the additional dimension – the temporal information, so that the approximated rating tensors are capable of providing more accurate predictions.

Similar to SVD and NMF that are used to protect user privacy in the accomplished works, NTF also has to preserve the privacy information. Some regular techniques would be used, e.g., applying random noise, manipulating the dimensionality of the factor matrices (which works like the truncated SVD), as well as perturbing the overall distribution of the samples while preserving the local distribution. Additionally, various kinds of information like trustworthiness and friendship, would also be considered.

The NTF model would be extended to the incremental case, as shown in Figure 7.1. In this scenario, once new items or users arrive in different time slots, the time factor matrix must be updated.

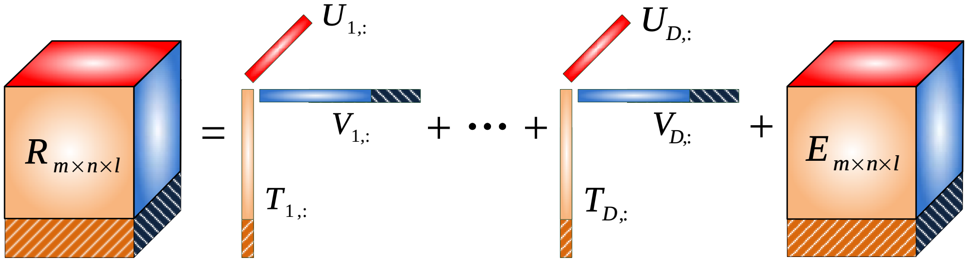


Figure 7.1: Incremental nonnegative tensor factorization

The Distributed Scenario

Collaborative filtering has proved to be one of the most effective techniques in recommender systems. An inevitable issue is, as data grows, the computational complexity of conventional centralized CF algorithms increases dramatically. It requires not only more memory but also faster processors to handle the large scale data. It is unrealistic to extend the memory to the unlimited size and the processor to unlimited fast speed on a single computer. Thus the distributed CF was proposed to resolve the problem.

Tveit[70] first proposed a distributed CF algorithm for the Peer-to-Peer mobile recommender systems. After that, the P2P based collaborative filtering techniques have been extensively studied, e.g., [81], [5], [24], [77], etc. Zhang et al. [85] studied the similar problem but in the cloud computing environment.

In future work, a distributed CF model that has one central server and several worker servers would be designed. This distributed model is inspired by the MapReduce[16] framework which was proposed by Google for processing large datasets. Figure 7.2 shows how MapReduce works.

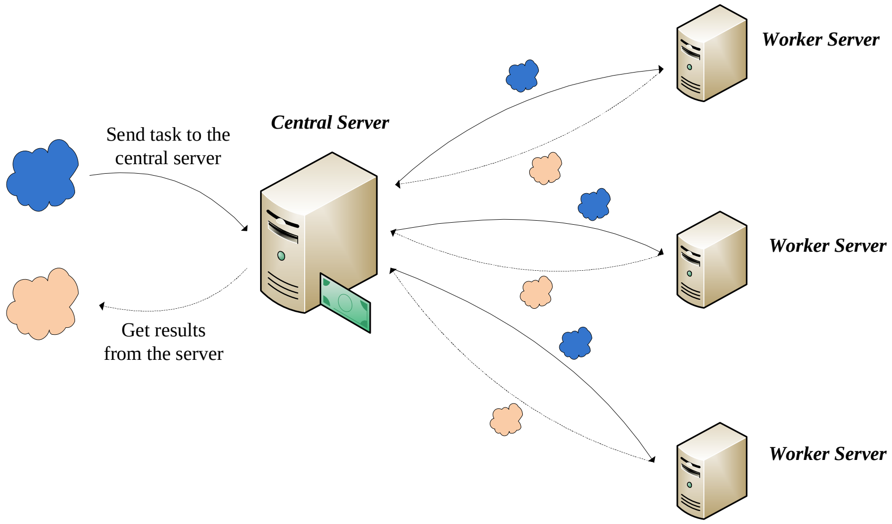


Figure 7.2: The MapReduce framework

However, the problems that need to be solved are: (1) how to manage the cluster membership indicator matrices in the MapReduce operations; (2) how to handle the partition of matrices as there are more than two matrices in the multiplication; (3) how to address the data growth issue.

To overcome the above three problems, insightful investigations of the update formulas in iAux-NMF are expected so that a good strategy to partition the matrices can be developed in order to adapt the update process to the MapReduce model.

Large Scale Recommender Systems

With the increasing popularity of online applications, the problem of managing fast growing data has become one of the major research topics in data science. Although the accomplished works in this dissertation have addressed the data update issues, the algorithms still need to be verified on huge datasets. Conventional CF models focus on the complete rating data to find the underlying correlations. However, if the size of the data is extremely large, the time cost would be unacceptable. A feasible solution is to sample the data. In other words, the models need to extract information that is manageable in size and can maximally represent the complete set. The subsequent CF algorithms can be performed only on the sample data to maintain the online performance.

In the future, the sampling techniques that can be used on large collaborative filtering data would be studied. It is expected that efficient and effective algorithms can be proposed to identify and obtain the representative features of the original rating and auxiliary data. These features are then fed to the existing collaborative filtering algorithms for real time fast recommendations. Nevertheless, the computation ability of single servers is limited so the sampling models should be adapted to the distributed scenarios. By doing so, the overall performance can be guaranteed.

Bibliography

[1] E. Acar, D. M. Dunlavy, T. G. Kolda and M. Mrup, Scalable Tensor Factorizations with Missing Data, In *Proceedings of the 2010 SIAM International Conference on Data Mining*, 701-712 (2010)

[2] G. Adomavicius and A. Tuzhilin, Toward the Next Generation of Recommender Syetems: A Survey of the State-of-the-Art and Possible Extensions, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, 6:734-749 (2005)

[3] D. Agrawal and C. C. Aggarwal, On the Design and Quantification of Privacy Preserving Data Mining Algorithms, In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 247-255 (2001)

[4] J. Bennet and S. Lanning, The Netflix Prize, In *Proceedings of KDD Cup and Workshop 2007* (2007)

[5] S. Berkovsky, Y. Eytani and L. Manevitz, Efficient CF in Content-addressable Spaces, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 21, 2:265-289 (2007)

[6] S. Berkovsky, Y. Eytani, T. Kuflik and F. Ricci, Enhancing Privacy and Preserving Accuracy of a Distributed Collaborative Filtering, In *Proceedings of the 2007 ACM Conference on Recommender Systems*, 9-16 (2007)

[7] M. Berry, Large-scale Sparse Singular Value Computations, *North-Holland*, ISBN 0-444-19451-7 (1976)

[8] A. Bilge and H. Polat, Improving Privacy-Preserving NBC-based Recommendations by Preprocessing, In *Proceedings of 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 1, 143-147 (2010)

[9] M. Brand, Fast Low-Rank Modifications of the Thin Singular Value Decomposition, *Linear Algebra and its Applications*, Vol. 415, 1:20-30 (2006)

[10] J. Breese, D. Heckerman and C. Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, *Technical Report, MSR-TR-98-12*, Microsoft Research, Microsoft Corporation (1998)

[11] L. Brozovsky and V. Petricek, Recommender System for Online Dating Service, In *Proceedings of Znalosti 2007 Conference*, Ostrava, Czech Republic (2007)

[12] J. Canny, Collaborative Filtering with Privacy, In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 45-57 (2002)

[13] G. Chen, F. Wang and C. Zhang, Collaborative Filtering Using Orthogonal Nonnegative Matrix Tri-factorization, In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, 303-308 (2007)

[14] R. Chen, M. Xie, L. Lakshmanan, Thwarting Passive Privacy Attacks in Collaborative Filtering, *Database Systems for Advanced Applications*, Vol. 8422, 218-233 (2014)

[15] P. Cremonesi, Y. Koren and R. Turrin, Performance of Recommender Algorithms on Top-N Recommendation Tasks, In *Proceedings of the 4th ACM Conference on Recommender Systems*, 39-46 (2010)

[16] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 137-149 (2004)

[17] A. Dempster, N. Laird and D. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, 1:1-38 (1977)

[18] M. Deshpande and G. Karypis, Item-Based Top-N Recommendation Algorithms, *ACM Transactions on Information Systems*, Vol. 22, 1:143-177 (2004)

[19] C. Ding, X. He and H. Simon, On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering, In *Proceedings of the 2005 SIAM International Conference on Data Mining*, 606-610 (2005)

[20] C. Ding, T. Li, W. Peng and H. Park, Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering, In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 126-135 (2006)

[21] Y. Ding and X. Li, Time Weight Collaborative Filtering, In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, 485-492 (2005)

[22] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, Eigentaste: A Constant Time Collaborative Filtering Algorithm, *Information Retrieval*, Vol. 4, 2:115-132 (2001)

[23] D. Goldberg, D. Nichols, B. Oki and D. Terry, Using Collaborative Filtering to Weave an Information Tapestry, *Communications of the ACM*, 35:61-70 (1992)

[24] P. Han, B. Xie, F. Yang and R. Shen, A Scalable P2P Recommender System Based on Distributed Collaborative Filtering, *Expert Systems with Applications*, Vol. 27, 2:203-210 (2004)

[25] J. Herlocker, J. Konstan, A. Borchers and J. Riedl, An Algorithmic Framework for Performing Collaborative Filtering, In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 230-237 (1999)

[26] C. Hsu, H. Chung and H. Huang, Mining Skewed and Sparse Transaction Data for Personalized Shopping Recommendation, *Machine Learning*, Vol. 57, 35-59 (2004)

[27] Z. Huang, A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining, *Research Issues on Data Mining and Knowledge Discovery*, 1-8 (1997)

[28] Z. Huang, S. Deng and X. Xu, Improving K-modes Algorithm Considering Frequencies of Attribute Values in Mode, *Computational Intelligence and Security - Lecture Notes in Computer Science*, Vol. 3801, 157-162 (2005)

[29] Z. Huang, D. Zeng and H. Chen, A Link Analysis Approach to Recommendation under Sparse Data, In *Proceedings of the 10th Americas Conference on Information Systems*, 1997-2005 (2004)

[30] J. Huang, F. Nie, H. Huang, Y. Lei and C. Ding, Social Trust Prediction Using Rank-k Matrix Recovery, In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2647-2653 (2013)

[31] M. Jamali and M. Ester, A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks, In *Proceedings of the 4th ACM Conference on Recommender Systems*, 135-142 (2010)

[32] S. Kabir, A. Youssef and A. Elhakeem, On Data Distortion for Privacy Preserving Data Mining, In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, 308-311 (2007)

[33] C. Kaleli and H. Polat, Privacy-Preserving Trust-based Recommendations on Vertically Distributed Data, In *Proceedings of the 5th IEEE International Conference on Semantic Computing*, 376-379 (2011)

[34] C. Kaleli and H. Polat, Providing Private Recommendations Using Naive Bayesian Classifier, *Advances in Intelligent Web Mastering - Advances in Soft Computing*, Vol. 43, 168-173 (2007)

[35] T. Kamishima and S. Akaho, Efficient Clustering for Orders, In *Proceedings of the 2nd International Workshop on Mining Complex Data*, 274-278 (2006)

[36] A. Karatzoglou, X. Amatriain, L. Baltrunas and N. Oliver, Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering, In *Proceedings of the 4th ACM Conference on Recommender Systems*, 79-86 (2010)

[37] J. Kim and H. Park, Sparse Nonnegative Matrix Factorization for Clustering, *Technical Report*, Georgia Institute of Technology, (2008)

[38] O. Koch and C. Lubich, Dynamical Low-Rank Approximation, *SIAM Journal on Matrix Analysis and Applications*, Vol. 29, 2:434-454 (2007)

[39] T. G. Kolda and B. W. Bader, Tensor Decompositions and Applications, *SIAM Review*, Vol. 51, 3:455-500 (2009)

[40] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon and J. Riedl, GroupLens: Applying Collaborative Filtering to Usenet News, *Communications of the ACM*, 40:77-87 (1997)

[41] Y. Koren, Collaborative Filtering with Temporal Dynamics, In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 447-456 (2009)

[42] Y. Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 436-434 (2008)

[43] H. Kuhn and A. Tucker, Nonlinear Programming, In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, 481-492 (1951)

[44] G. Lawler, Introduction to Stochastic Processes, Second Edition, *Chapman & Hall*, ISBN(13) 978-1584886518 (2006)

[45] D. Lee and H. Seung, Algorithms for Non-negative Matrix Factorization, *Advances in Neural Information Processing Systems*, Vol. 13, 556-562 (2001)

[46] M. Li, B. Dias, I. Jarman, W. EI-Deredy and P. Lisboa, Grocery Shopping Recommendations Based on Basket-Sensitive Random Walk, In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1215-1223 (2009)

[47] T. Li, C. Gao and J. Du, A NMF-based Privacy-Preserving Recommendation Algorithm, In *Proceedings of the 1st International Conference on Information Science and Engineering*, 754-757 (2009)

[48] D. Li, Q. Lv, H. Xia, L. Shang, T. Lu and N. Gu, Pistis: A Privacy-Preserving Content Recommender System for Online Social Communities, In *Proceedings of 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 79-86 (2011)

[49] C. Liu, H. Yang, J. Fan, L. He and Y. Wang, Distributed Nonnegative Matrix Factorization for Web-Scale Dyadic Data Analysis on MapReduce, In *Proceedings of the 19th International World Wide Web Conference*, 681-690 (2010)

[50] H. Liu and Z. Wu, Non-Negative Matrix Factorization with Constraints, In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 506-511 (2010)

[51] J. MacQueen, Some Methods for Classification and Analysis of Multivariate Observations, In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, 281-297 (1967)

[52] P. Massa and P. Avesani, Trust-aware Bootstrapping of Recommender Systems, *ECAI 2006 Workshop on Recommender Systems*, 29-33 (2006)

[53] 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*, 627-636 (2009)

[54] N. Metropolis and S. Ulam, The Monte Carlo Method, *Journal of the American Statistical Association*, Vol. 44, 247:335-341 (1949)

[55] M. Papagelis and D. Plexousakis, Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Agents, *Engineering Applications of Artificial Intelligence*, Vol. 18, 7:781-789 (2005)

[56] S. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste, Naive Filterbots for Robust Cold-Start Recommendations, In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 699-705 (2006)

[57] A. Paterek, Improving Regularized Singular Value Decomposition for Collaborative Filtering, In *Proceedings of KDD Cup and Workshop*, 39-42 (2007)

[58] V. Pauca, F. Shahnaz, M. Berry and R. Plemmons, Text Mining Using Nonnegative Matrix Factorizations, In *Proceedings of 2004 SIAM Interational Conference on Data Mining*, Vol. 54, 452-456 (2004)

[59] H. Polat and W. Du, Privacy-Preserving Collaborative Filtering, *International Journal of Electronic Commerce*, Vol. 9, 4:9-35 (2005)

[60] H. Polat and W. Du, SVD-based Collaborative Filtering with Privacy, In *Proceedings of the 2005 ACM Symposium on Applied Computing*, 791-795 (2005)

[61] S. Rendle, L. B. Marinho, A. Nanopoulos and L. Schmidt-Thieme, Learning Optimal Ranking with Tensor Factorization for Tag Recommendation, In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 727-736 (2009)

[62] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, GroupLens: An Open Architecture for Collaborative Filtering of Netnews, In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 175-186 (1994)

[63] R. Sandler and M. Lindenbaum, Nonnegative Matrix Factorization with Earth Mover's Distance Metric for Image Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, 8:1590-1602 (2011)

[64] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, Application of Dimensionality Reduction in Recommender Systems – A Case Study, In *Proceedings of ACM WebKDD Workshop* (2000)

[65] S. Shang, Y. Huiy, P. Huiz, P. Cuff and S. Kulkarni, Privacy Preserving Recommendation System Based on Groups, http://arxiv.org/abs/1305.0540, (2013)

[66] 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*, 210-217 (1995)

[67] G. Stewart, Perturbation Theory for the Singular Value Decomposition, *Technical Report, CS-TR-2539*, Computer Science Department, University of Maryland (1990)

[68] X. Su, Y. Lan, R. Wan and Y. Qin, A Fast Incremental Clustering Algorithm, In *Proceedings of the 2009 International Symposium on Information Processing*, 175-178 (2009)

[69] J. Tang, H. Gao and H. Liu, mTrust: Discerning Multi-faceted Trust in a Connected World, In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, 93-102 (2012)

[70] A. Tveit, Peer-to-peer Based Recommendations for Mobile Commerce, In *Proceedings of the 1st International Workshop on Mobile Commerce*, 26-29 (2001)

[71] N. Thapa, L. Liu, P. Lin, J. Wang and J. Zhang, Constrained Nonnegative Matrix Factorization for Data Privacy, In *Proceedings of the 7th International Conference on Data Mining*, 88-93 (2011)

[72] J. Tougas and R. Spiteri, Updating the Partial Singular Value Decomposition in Latent Semantic Indexing, *Computational Statistics & Data Analysis*, Vol. 52, 174-183 (2007)

[73] S. Vucetic and Z. Obradovic, Collaborative Filtering Using a Regression-based Approach, *Knowledge and Information Systems*, Vol. 7, 1:1-22 (2005)

[74] J. Wang, W. Zhong and C. Zhang, NNMF-Based Factorization Techniques for High-Accuracy Privacy Protection on Non-negative-valued Datasets, In *Proceedings of the 6th IEEE International Conference on Data Mining Workshops*, 513-517 (2006)

[75] 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*, 606-611 (2008)

[76] S. Warner, Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias, *Journal of the American Statistical Association*, Vol. 60, 309:63-69 (1965)

[77] B. Xie, P. Han, F. Yang, R. Shen, H. Zeng, Z. Chen, DCFLA: A Distributed Collaborative-filtering Neighbor-locating Algorithm, *Information Sciences*, Vol. 177, 1349-1363 (2007)

[78] L. Xiong, X. Chen, T. Huang, J. Schneidery and J. G. Carbonellz, Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization, In *Proceedings of SIAM Data Mining*, 211-222 (2010)

[79] W. Xu, X. Liu and Y. Gong, Document Clustering Based On Non-negative Matrix Factorization, In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 267-273 (2003)

[80] B. Yang, Y. Lei, D. Liu and J. Liu, Social Collaborative Filtering by Trust, In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2747-2753 (2013)

[81] F. Yuan, J. Liu, C. Yin and Y. Zhang, A Distributed Recommendation Mechanism Based on Collaborative Filtering in Unstructured P2P Networks, *Journal of Computational Information Systems*, Vol. 4, 3:1111-1118 (2008)

[82] K. Yu, S. Zhu, J. Lafferty and Y. Gong, Fast Nonparametric Matrix Factorization for Large-scale Collaborative Filterings, In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 211-218 (2009)

[83] J. Zhan, C. Hsieh, I. Wang, T. Hsu, C. Liau and D. Wang, Privacy-Preserving Collaborative Recommender Systems, *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, Vol. 40, 4:472-476 (2010)

[84] S. Zhang, J. Ford and F. Makedon, A Privacy-Preserving Collaborative Filtering Scheme with Two-way Communication, In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 316-323 (2006)

[85] Y. Zhang, H. Liu and S. Li, A Distributed Collaborative Filtering Recommendation Mechanism for Mobile Commerce Based on Cloud Computing, *Journal of Information & Computational Science*, Vol. 8, 16:3883-3891 (2011)

[86] J. Zhang, Image Fusion Based on Nonnegative Matrix Factorization, In *Proceedings of 2004 International Conference on Image Processing*, Vol.2, 973-976 (2004)

[87] S. Zhang, W. Wang, J. Ford and F. Makedon, Learning from Incomplete Ratings Using Non-negative Matrix Factorization, In *Proceedings of the 6th SIAM International Conference on Data Mining*, 548-552 (2006)

Vita

Personal Data:

Name: Xiwei Wang

Place of Birth: Wuhu, Anhui, China

Educational Background:

- M.Eng. in Computer Software and Theory, University of Science and Technology of China, Hefei, China, 2009.
- B.Sc. in Information and Computing Science, Hefei University of Technology, Hefei, China, 2006.

Professional Experience:

- Teaching Assistant, 08/2010 - 05/2015. Department of Computer Science, University of Kentucky, Lexington, KY, USA.
- Research Assistant, 08/2009 - 05/2010. Department of Computer Science, University of Kentucky, Lexington, KY, USA.
- Teaching Assistant, 09/2007 - 02/2008. School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
- Research Assistant, 09/2006 - 06/2009. School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Awards:

- Verizon Communication Graduate Fellowship, University of Kentucky, August 2014.

- Student travel grant from IEEE International Conference on Information Reuse and Integration, July 2014.

- Student travel support from Graduate School Fellowship, University of Kentucky, July 2014.

- Third prize in 27th Annual Eastern Kentucky University Symposium in Mathematical, Statistical and Computer Sciences, April 2014.

- ACM Outstanding Teaching Assistant, University of Kentucky, April 2014.

- Student travel support from Graduate School Fellowship, University of Kentucky, May 2012.

- First category in International Teaching Assistant screening test, University of Kentucky, January 2011.

- First prize in 24th Annual Eastern Kentucky University Symposium in Mathematical, Statistical and Computer Sciences, April 2010.

Publications:

- **Xiwei Wang**, Jun Zhang and Yin Wang, Trust-aware Privacy Preserving Recommender System. Under revision for *Information Processing & Management*, (2015).

- **Xiwei Wang**, Jun Zhang and Ruxin Dai, Automated Dimension Determination for NMF-based Incremental Collaborative Filtering. Under review by *EAI Endorsed Transactions on Collaborative Computing*, (2014).

- **Xiwei Wang** and Jun Zhang. Using Incremental Clustering Technique in Collaborative Filtering Data Update. In *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration*, 420-427 (2014).

- **Xiwei Wang**, Jun Zhang, Pengpeng Lin, Nirmal Thapa, Yin Wang and Jie Wang, Incorporating Auxiliary Information in Collaborative Filtering Data Update with Privacy Preservation. *International Journal of Advanced Computer Science and Applications*, Vol. 5, 4:224-235 (2014).

- **Xiwei Wang** and Jun Zhang, Handling the Data Growth with Privacy Preservation in Collaborative Filtering. *IAENG Transactions on Engineering Technologies*, Vol. 229, 231-243 (2013).

- **Xiwei Wang** and Jun Zhang, SVD-based Privacy Preserving Data Updating in Collaborative Filtering. In *Proceedings of the World Congress on Engineering 2012*, Vol. I, 377-384 (2012).

- **Xiwei Wang**, Erik von der Osten, Xuzi Zhou, Hui Lin, and Jinze Liu, A Case Study of Recommendation Algorithms. In *Proceedings of the 2011 International Conference on Computational and Information Sciences*, 410-417 (2011).

- **Xiwei Wang**, Haifeng You, Xufa Wang, Interest-based Trust Propagation in Blog Community. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery*, Vol. 7, 555-560 (2009).

- **Xiwei Wang** and Xufa Wang, An Approach of Blogshpere Construction by Using Bloggers' Interests. *Journal of Chinese Computer Systems*, Vol. 30, 12:2424-2427 (2009). (In Chinese)

- Kiho Lim and **Xiwei Wang**, NMF-based Privacy Preservation in Vehicular Communication. Accepted by the *IEEE SoutheastCon 2015 Conference*, (2015).

- Ruxin Dai, Yin Wang and **Xiwei Wang**, Effects of Different High Order Compact Computations for Solving Boundary Layer Problems on Nonuniform Grids. *Journal of Computational Intelligence and Electronic Systems*, Vol. 3, 3:200-211 (2014).

- Pengpeng Lin, Jun Zhang, **Xiwei Wang** and Art Shindhelm, Simultaneous Pattern and Data Hiding in Supervised Learning. In *Proceedings of the 13th IEEE International Conference on Information Reuse and Integration*, 385-392 (2012).

- Haifeng You, **Xiwei Wang**, Xiang Xu and Xufa Wang, A Multi-user Interactive Genetic Algorithm and Its Application in Group Design. *Journal of University of Science and Technology of China*, Vol. 40, 4:425-430 (2010). (In Chinese)