

1-1-2003

Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques

Huseyin Polat
Syracuse University

Wenliang Du
Syracuse University, wedu@syr.edu

Follow this and additional works at: <http://surface.syr.edu/eecs>

 Part of the [Computer Sciences Commons](#)

Recommended Citation

Polat, Huseyin and Du, Wenliang, "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques" (2003).
Electrical Engineering and Computer Science. Paper 18.
<http://surface.syr.edu/eecs/18>

This Working Paper is brought to you for free and open access by the L.C. Smith College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

Privacy-Preserving Collaborative Filtering using Randomized Perturbation Techniques

Huseyin Polat and Wenliang Du
Systems Assurance Institute
Department of Electrical Engineering and Computer Science
Syracuse University, 121 Link Hall, Syracuse, NY 13244
Email: {hpolat,wedu}@ecs.syr.edu

Abstract

Collaborative Filtering (CF) techniques are becoming increasingly popular with the evolution of the Internet. E-commerce sites use CF systems to suggest products to customers based on like-minded customers' preferences. People use CF systems to cope with information overload. To conduct collaborative filtering, data from customers are needed. However, collecting high quality data from customers is not an easy task because many customers are so concerned about their privacy that they might decide to give false information. CF systems using these data might produce inaccurate recommendations.

We propose a randomized perturbation technique to protect users' privacy while still producing accurate recommendations. Although the randomized perturbation techniques add randomness to the original data to prevent the data collector from learning the private user data, our scheme can still provide recommendations with decent accuracy. We conducted several experiments to compare the recommendations on the randomized data with those on the original data. Using these experiment results, we analyzed how different parameters affect the accuracy. Our results show that the CF systems using the randomized perturbation techniques provide accurate recommendations while preserving the users' privacy.

Keywords: Privacy, Collaborative Filtering, Randomized Perturbation.

1 Introduction

With the amount of the information available for individuals growing steadily, information overload has become a major problem for users. To make the information to serve users better, information filtering and recommendation schemes become more and more important. Collaborative filtering (CF) or social filtering is a recent technique for such filtering and recommendation purposes.

The term "collaborative filtering" was first coined by the designers of Tapestry [7], a mail filtering software developed in the early nineties for the intranet at the Xerox Palo Alto Research Center. With the number of users accessing the Internet growing, CF techniques are becoming increasingly popular as part of online shopping sites. These sites incorporate recommendation systems that suggest products to users based on products that like-minded users have ordered before, or indicate as interesting. Collaborative filtering has many important applications [4, 5] in e-commerce, direct recommendations, and search engines. Users can get recommendations about many of their daily activities, including restaurants, bars, movies, books, news, music CDs, interesting sights to see, and things to do in a city with the help of collaborative filtering.

The goal of CF [13] is to predict the preferences of one user, referred to as the active user, based on the preferences of a group of other users. For example, given the active user’s ratings for several movies and a database of other users’ ratings, the system predicts how the active user would rate unseen movies. The key idea is that the active user will prefer those items that like-minded users prefer, or that dissimilar users do not. The fundamental assumption [8] is that if users A and B rate k items similarly, they share similar tastes, and hence will rate other items similarly.

1.1 Privacy-Preserving Collaborative Filtering Problem

Today’s CF systems have a number of disadvantages [4, 5]. The most important is that they are a serious threat to individual privacy. There is a great potential for individuals to share all kinds of information about places and things to do, see and buy, but the privacy risks are severe. Most online vendors collect buying information and preferences about their customers and make reasonable efforts to keep this data private. However, customer data is a valuable asset and it has been sold when some e-companies suffered bankruptcy.

The fragility of the digital marketplace has put many e-companies in bankruptcy [5]. The courts have supported the rights of liquidators to sell off data about their customers’ personal information as an asset. From Amazon.com’s policy “In the unlikely event that Amazon.com, Inc., or substantially all of its assets are acquired, customer information will of course be one of the transferred assets.” This policy is typical of many other companies and, in spite of Amazon’s language, the papers are filled with news of such “unlikely events” which lead to bankruptcies and transfer of customer information.

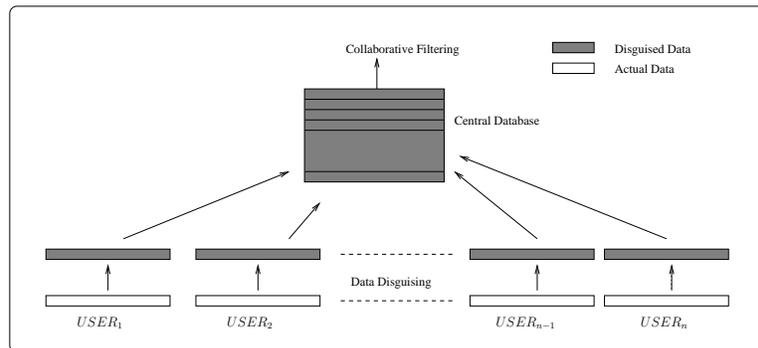


Figure 1: Privacy Preserving Collaborative Filtering

Some people might be willing to selectively divulge information if they can get benefit in return [19]. Examples of the benefit provided include discount of purchase, useful recommendations, and information filtering. However, a significant number of people are not willing to divulge their information because of privacy concerns, according to a survey conducted in 1999 [6]. The challenge is *how can users contribute their personal information for collaborative filtering purposes without compromising their privacy?*

One way to achieve privacy is to use anonymous techniques [1, 14, 18], which allow users to disclose their personal information without disclosing their identities. The biggest problem of using anonymous techniques is that there is no guarantee on the quality of the dataset. A malicious user (e.g. a competing company) could send a great deal of random information to the database and render the database useless, or a company could send a lot of made-up information to the database with the goal of making their products the most favorable ones. These potential attacks could all render the database useless. If the communication is really anonymous, it is difficult for the

database owner to control the quality of the data. It is important for the database owner to verify the identities of the data contributors in order to guarantee the quality.

Canny shows another way to conduct collaborative filtering without disclosing each user’s private data [4, 5]. The solution allows individuals both inside and outside the community to gain recommendations without learning each individual’s private data. In this scheme, many users need to participate in each recommendation.

1.2 Outline of Our Solution

We propose a new scheme in this paper to allow the privacy-preserving collaborative filtering. In our scheme (Fig. 1), each user first disguises his/her personal data, and then sends to a central place (the data collector), such that the data collector cannot derive the truthful information about a user’s private information. However, the data disguising scheme should still be able to allow the data collector to conduct collaborative filtering from the disguised data. We propose to use *Randomized Perturbation* techniques to disguise private data.

The basic idea of randomized perturbation is to perturb the data in such a way that the central place can only know the range of the data, and such range is broad enough to preserve users’ privacy. Although information from each individual user is scrambled, if the number of users is significantly large, the aggregate information of these users can be estimated with decent accuracy. Such property is useful for computations that are based on aggregate information. For those computations, we can still generate meaningful outcome without knowing the exact values of individual data items because the needed aggregate information can be estimated from the scrambled data. Since collaborative filtering is based on aggregate values of a dataset, rather than individual data items, we hypothesize that *by combining the randomized perturbation techniques with collaborative filtering algorithms, we can achieve a decent degree of accuracy for the privacy-preserving collaborative filtering.*

To verify this hypothesis, we implemented the randomized perturbation technique for a collaborative filtering algorithm, which was proposed by [10]. We then conducted a series of experiments to show how accurate our results are. We compared the predictions that are calculated based on original data with the predictions on randomized data. Our results show that if the number of users and items are significantly large, the predictions we have found on randomized data are very close to the predictions found on the original data.

2 Related Work

2.1 Privacy-Preserving Collaborative Filtering

Canny proposes two schemes for privacy-preserving collaborative filtering [4, 5]. In these schemes, users control all of their own private data; a community of users can compute a public “aggregate” of their data without disclosing individual users’ data. The aggregate allows personalized recommendations to be computed by members of the community, or by outsiders. Canny’s method reduces the collaborative filtering task to an iterative calculation of the aggregate requiring only addition of vectors of user data. Canny then uses homomorphic encryption to allow sums of encrypted vectors to be computed and decrypted without exposing individual data. His schemes are based on distributed computation of a certain aggregate of all users’ data. The aggregate is treated as public data. Each user constructs the aggregate and uses local computation to get personalized recommendations. Canny’s schemes can be implemented with untrusted servers, or with additional

infrastructures, as a fully peer-to-peer (P2P) system. The P2P architecture allows users to create and maintain their own recommender groups themselves.

While Canny’s work focuses on the peer-to-peer framework, in which users actively participate in the collaborative filtering process, our work focuses on another framework, in which users send their data to a central place and they do not participate in the CF process; only the central place needs to conduct the CF. Both frameworks have their applications. The peer-to-peer framework is more suitable in community-based CF systems while our framework is more suitable for systems that provide on-line CF services, such as Amazon, Yahoo travel, etc.

Randomized perturbation scheme was used by Agrawal and Srikant to solve privacy-preserving data mining problems [2]. The paper shows that it is possible to build decision tree classifiers if data is disguised using the randomized perturbation scheme. Our work solves a different but related problem.

2.2 Notations

We used the following notations in our study here:

v_{ij} is vote for user i on item j .

w_{ai} is similarity weight between active user a and user i .

\bar{v}_a is mean vote for active user a .

\bar{v}_i is mean vote for user i .

p_{aq} is prediction for active user a on item q .

σ_a is standard deviation of active user a ’s ratings.

σ_i is standard deviation of user i ’s ratings.

α is range of random numbers.

γ shows percentile of the data.

z_{ij} is z-score for user i on item j .

MAE is mean absolute error.

2.3 Collaborative Filtering Algorithms

There are two general classes of CF algorithms [3]. Memory-based algorithms operate over the entire user database to make predictions. Model-based CF algorithms use the user database to estimate or learn a model, which is then used for predictions. Cluster models and Bayesian network model are two of the model-based collaborative filtering algorithms. There is another classes of CF algorithms that are hybrid memory and model based algorithms [13].

The task in CF is to predict the votes of a particular user (the active user) based on a database of user votes from a sample or population of other users (the user database) [3]. The user database consists of a set of votes v_{ij} corresponding to the vote for user i on item j . The user database is usually a sparse matrix. The users do not rate all of the items. The missing ratings can be filled by using the item mean vote, the user mean vote, or the overall mean vote [5].

In memory-based collaborative filtering algorithms, we predict the votes of the active user (indicated with a subscript a) based on some partial information regarding the active user and a set of weights calculated from the user database [3].

Memory-based algorithms can be divided into different sub-categories in terms of the details of the weight calculation. Pearson correlation coefficient and vector similarity are two of the methods of weight calculation.

Pearson Correlation Coefficient This formulation first appeared in the published literature in the context of the GroupLens project [15], where the Pearson correlation coefficient was defined as the basis for the weight calculation. The correlation between active user a and user i is [3]:

$$w_{ai} = \frac{\sum_k (v_{ak} - \bar{v}_a)(v_{ik} - \bar{v}_i)}{\left[\sum_k (v_{ak} - \bar{v}_a)^2 \sum_k (v_{ik} - \bar{v}_i)^2 \right]^{1/2}} \quad (1)$$

where the summations over k are over the items for which both the active user a and the user i have voted. \bar{v}_a and \bar{v}_i are mean votes for active user a and user i , respectively. If I_i is the set of items on which user i has voted, then we can define the mean vote for user i as [3]:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{ij} \quad (2)$$

Vector Similarity In vector similarity, each user votes are represented as a vector. The weights can be calculated as [3]:

$$w_{ai} = \sum_k \frac{v_{ak}}{\left[\sum_{s \in I_a} (v_{as})^2 \right]^{1/2}} \frac{v_{ik}}{\left[\sum_{s \in I_i} (v_{is})^2 \right]^{1/2}} \quad (3)$$

where I_a is the set of items on which active user a has voted.

GroupLens introduced an automated collaborative filtering system using a neighborhood-based algorithm [12, 15]. GroupLens provided personalized predictions for Usenet news articles. The original GroupLens system used Pearson correlations to weight user similarity, used all available correlated neighbors, and computed a final prediction by performing a weighted average of deviations from the neighbor's mean:

$$p_{aq} = \bar{v}_a + \frac{\sum_{i=1}^n (v_{iq} - \bar{v}_i) \cdot w_{ai}}{\sum_{i=1}^n w_{ai}} \quad (4)$$

p_{aq} represents the prediction for the active user a on item q . n is the number of neighbors and w_{ai} is the similarity weight between the active user a and the user i as defined by the Pearson correlation coefficient.

An extension of the GroupLens algorithm, which we use in our study in here was proposed by [10]. Herlocker et al. compare the performance of different normalization techniques such as the bias-from-mean, the z-scores, and the non-normalized rating. The z-scores perform significantly better than the non-normalized rating approach. The mean and the standard deviation of the z-scores are 0 and 1, respectively. If the v_{ij} is user i 's vote on item j , \bar{v}_i is the mean vote of the user i , and σ_i is the standard deviation for the user i , then the z-scores (z_{ij}) can be defined as:

$$z_{ij} = \frac{(v_{ij} - \bar{v}_i)}{\sigma_i} \quad (5)$$

Herlocker et al. [10] account for the differences in spread between users' rating distributions by converting ratings to z-scores, and compute a weighted average of the z-scores:

$$p_{aq} = \bar{v}_a + \sigma_a \cdot \frac{\sum_{i=1}^n \frac{(v_{iq} - \bar{v}_i)}{\sigma_i} \cdot w_{ai}}{\sum_{i=1}^n w_{ai}} \quad w_{ai} = \frac{\sum_k (v_{ak} - \bar{v}_a) \cdot (v_{ik} - \bar{v}_i)}{\sigma_a \cdot \sigma_i} \quad (6)$$

where k is the item set both the active user a and the user i have rated. σ_a and σ_i are standard deviations of the active user a 's ratings and the user i 's ratings, respectively.

There are also other memory-based collaborative filtering algorithms. The Ringo Music Recommender [17] and the Bellcore Video Recommender [11] expanded upon the original GroupLens algorithm. Ringo used 4 as the mean vote for all users and limited membership when computing the weights while Bellcore Video Recommender selected the best neighbors to create a prediction. A constant time collaborative filtering algorithm was proposed by [8], which is based on principal component analysis (PCA).

3 Privacy Preserving CF using Randomized Perturbation

3.1 Randomized Perturbation Techniques

There are several ways to hide numbers or information. To hide a number a , a simple way is to add a random number r to it. Although we cannot do anything to a since it is disguised, we can conduct certain computations if we are interested in the aggregate data, rather than each individual data.

The basic idea of randomized perturbation is to perturb the data in such a way that certain computations can be done while preserving users' privacy. Although information from each individual user is scrambled, if the number of users is significantly large, the aggregate information of these users can be estimated with decent accuracy. Such property is useful for computations that are based on aggregate information. Scalar product and sum are among such computations and used in collaborative filtering algorithms. For those computations, we can still generate meaningful outcome without knowing the exact values of individual data items because the needed aggregate information can be estimated from the scrambled data.

Scalar Product Let A and B be the original vectors, where $A = (a_1, \dots, a_n)$ and $B = (b_1, \dots, b_n)$. A is disguised by $R = (r_1, \dots, r_n)$, and B is disguised by $V = (v_1, \dots, v_n)$, where r_i 's and v_i 's are uniformly distributed in domain $[-\alpha, \alpha]$. Let $A' = A + R$ and $B' = B + V$ be the disguised data that are known, we now show how the scalar product of A and B can be estimated from A' and B' :

$$A' \cdot B' = \sum_{i=1}^n (a_i b_i + a_i v_i + r_i b_i + r_i v_i)$$

Because R and B are independent, we have $\sum_{i=1}^n r_i b_i \approx 0$; similarly we have $\sum_{i=1}^n a_i v_i \approx 0$, and $\sum_{i=1}^n r_i v_i \approx 0$. Therefore, we have

$$\sum_{i=1}^n (a_i + r_i)(b_i + v_i) = \sum_{i=1}^n (a_i b_i + a_i v_i + r_i b_i + r_i v_i) \approx \sum_{i=1}^n a_i b_i \quad (7)$$

Sum Let A be the original vector with n values, where $A = (a_1, \dots, a_n)$. A is disguised by $R = (r_1, \dots, r_n)$, where r_i 's are uniformly distributed in domain $[-\alpha, \alpha]$. Let $A' = A + R$ be the disguised data that is known. Since r_i 's are uniformly distributed in domain $[-\alpha, \alpha]$, the contribution of the sum of the random values to the actual sum of the values of vector A is close to zero. In the long run, the relative error will converge to zero. Therefore, we have

$$\sum_{i=1}^n (a_i + r_i) = \sum_{i=1}^n a_i + \sum_{i=1}^n r_i \approx \sum_{i=1}^n a_i \quad (8)$$

Next, we will show how we use these two approximation techniques to conduct privacy-preserving collaborative filtering.

3.2 CF with Privacy using Randomized Perturbation Techniques

Our goal of collaborative filtering using randomized perturbation is to achieve privacy and produce recommendations with high accuracy. However, achieving privacy and producing accurate recommendations are two conflicting goals. Users might send false data instead of their actual data to achieve perfect privacy. But producing accurate recommendations is impossible from this false data. On the other hand, if users send their actual data to the server, finding high quality recommendations is possible but the users' privacy is not preserved. We propose a technique to achieve a good balance between the privacy and the accuracy.

Without privacy concerns, every user sends his/her ratings to the server, which creates a central database containing ratings from all users. To get a recommendation, an active user sends his/her known ratings and a query (for which item he/she is looking for prediction) to the server, and the server can calculate the p_{aq} (predicted vote for active user a on item q) using the Eq. (6).

With the privacy concerns, the server should not know the true data of each user including the active users. We use the randomized perturbation technique to achieve data disguise. In our approach, users add a random number to each of their actual ratings that they want to disguise, and send the results to the server. The server should not be able to find the true values of the ratings because of the random numbers. To show how our approach works, we use the z-score notation to simplify the Eq. (6):

$$z_{iq} = \frac{v_{iq} - \bar{v}_i}{\sigma_i} \quad (9)$$

Therefore, from Eq. (6) we get:

$$p_{aq} = \bar{v}_a + \sigma_a \cdot p'_{aq} = \bar{v}_a + \sigma_a \cdot \frac{\sum_{i=1}^n w_{ai} \cdot z_{iq}}{\sum_{i=1}^n w_{ai}}, \quad w_{ai} = \sum_k z_{ak} \cdot z_{ik} \quad (10)$$

where

$$p'_{aq} = \frac{\sum_{i=1}^n \left[\sum_k z_{ak} \cdot z_{ik} \right] \cdot z_{iq}}{\sum_{i=1}^n \sum_k z_{ak} \cdot z_{ik}} = \frac{\sum_k z_{ak} \cdot \left[\sum_{i=1}^n z_{ik} \cdot z_{iq} \right]}{\sum_k z_{ak} \cdot \left[\sum_{i=1}^n z_{ik} \right]} \quad (11)$$

Since the active user and the other users have not rated all items, the counter, k , is different from user to user. Only those items that have been rated by both the active user a and the user i are involved in computations. The entries for those items that have not been rated are zero.

Notice that the nominator part consists of a scalar product between vector $Z_k = (z_{1k}, \dots, z_{nk})$ and vector $Z_q = (z_{1q}, \dots, z_{nq})$. If the server can compute the scalar products for all k 's, it can send the results of the scalar products to the active user who can easily compute the nominator part. The denominator part is even simpler. All the server needs to do is to send the result of $\sum_{i=1}^n z_{ik}$ for each k to the active user.

Based on the above observations, we develop the following scheme, such that each user does not need to send the actual information to the server while still allowing the server and the future active users to compute p'_{aq} jointly.

1. The server decides on a range $[-\alpha, \alpha]$, and let each user know. How the server decides on the value of α will be discussed later.
2. Each user i computes his/her mean vote, standard deviation, and then calculates the z-scores z_{ij} for the items that he/she has rated.
3. Each user i creates n_i uniform random numbers r_{ij} in the range $[-\alpha, \alpha]$, where n_i is the total number of items that user has rated; note that n_i might be different from user to user because every user might have rated different number of items. User i then adds those random numbers to its z-score ratings and generates the disguised z-scores $z'_{ij} = z_{ij} + r_{ij}$. User i then sends the results (n_i disguised z-scores) to the server.

After getting all the disguised z-scores z'_{ij} from many users, the server can now provide collaborative filtering services to active users based on the following facts from Eq. (7) and (8):

$$\begin{aligned} \sum_{i=1}^n z_{ik} \cdot z_{iq} &\approx \sum_{i=1}^n z'_{ik} \cdot z'_{iq} \\ \sum_{i=1}^n z_{ik} &\approx \sum_{i=1}^n z'_{ik} \end{aligned} \quad (12)$$

To get a recommendation for item q , the active user computes the z-scores z_{ak} for those items that he/she has rated before. Then, the server sends the results of $\sum_{i=1}^n z'_{ik} \cdot z'_{iq}$ and $\sum_{i=1}^n z'_{ik}$ for all k to the active user who uses Eq. (11) and (12) to compute p'_{aq} and p_{aq} , the predicted rating for the active user a on item q .

3.3 Selecting the Range of the Random Numbers

To protect the private data, the range of the random numbers is critical. If the range is too small, the perturbed data still discloses significant amount of information; if the range is too large, we hypothesize that the accuracy of the results will be very low. To understand how the range affects the accuracy, we associate the range with the distribution of the original data.

Let $X = (x_1, \dots, x_n)$ be a vector, where $x_i \in [c, d]$ for $i = 1 \dots n$. To disguise the values of the vector X using the randomized perturbation technique, we generate n uniform random numbers from $[-\alpha, \alpha]$ and add them to the values of the vector X .

Percentile (γ)	Range of random numbers
95%	[-1.95, 1.95]
85%	[-1.43, 1.43]
75%	[-1.15, 1.15]
50%	[-0.67, 0.67]

Table 1: Examples

Assume the distribution (probability density function) of the random variable x for the sequence in X is $f(x)$. We calculate the percentile of the distribution, and find the range for this percentile. For example, for 95 percentile, we find a range in $f(x)$, such that 95% of the data of this distribution falls into the range.

When the distribution is standard normal distribution, i.e. the distribution is a normal distribution with the mean (μ) being 0 and the standard deviation (σ) being 1, the ranges for certain percentiles can be looked up from a table. Fig. 2 shows the meaning of various percentiles for the standard normal distribution, and Table 1 shows the ranges of the random numbers for various percentiles in standard normal distribution.

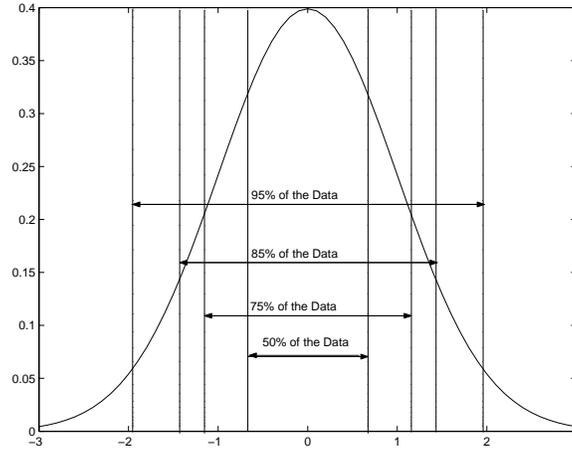


Figure 2: Standard Normal Distribution (mean 0, standard deviation 1)

From now on, we use γ to represent the percentile. After deciding the γ values, we can decide the range of the random numbers based on the distribution of the data.

3.4 Various Ways for Data Disguising

There are several ways we can disguise the original data using the random perturbation techniques.

- *Fixed Range:* We use a fixed range to generate random numbers. After deciding on the range of the random numbers, we generate random numbers uniformly within this range.
- *Random Range:* In this scheme, after deciding on the range of the random numbers, each user randomly generates a number α within this range, and then uses $[-\alpha, \alpha]$ as the range to generate uniform random numbers to disguise all his/her private data.

4 Experimental Results

4.1 Datasets

We use Jester and MovieLens datasets in our experiments to evaluate the accuracy of our randomized-perturbation-based collaborative filtering scheme. We compare the predictions based on original data with the predictions calculated from randomized data using our scheme. We describe the details of the datasets in the following:

- Jester is a web-based joke recommendation system, developed at University of California, Berkeley [9]. The database has 100 jokes and records of 17,988 users. The ratings range from -10 to +10, and the scale is continuous. Some users end up reading and rating all the jokes, so Jester is much more dense than the other datasets we used. Almost 50% of all possible ratings are present.
- MovieLens data were collected by the GroupLens Research Project at the University of Minnesota (www.cs.umn.edu/research/GroupLens). There are two datasets available. The first one (called MovieLens Public Data) consists of 100,000 ratings for 1,682 movies by 943 users. The second one (called MovieLens Million Data) consists of approximately 1 million ratings for approximately 3,500 movies made by 7,463 users. Each user has rated at least 20 movies. Ratings are made on a 5-star scale.

4.2 Evaluation Criteria

Several evaluation criteria for collaborative filtering have been used in literature [5, 10, 17]. The most common criteria are the *Mean Absolute Error (MAE)* and the *standard deviation (σ)*. We also use these two criteria in our evaluation.

If p_1, p_2, \dots, p_d are predicted values from undisguised data, and p'_1, p'_2, \dots, p'_d are predicted values from disguised data, then $E = \{\xi_1, \xi_2, \dots, \xi_d\} = \{p'_1 - p_1, p'_2 - p_2, \dots, p'_d - p_d\}$ represents errors. Therefore, the MAE and the standard deviation of the errors are computed using the following equations:

$$\bar{E} = \frac{\sum_{i=1}^d |\xi_i|}{d} \quad \text{and} \quad \sigma = \sqrt{\frac{\sum_{i=1}^d (E - \bar{E})^2}{d - 1}}$$

4.3 Methodology

The outline of our procedure is described in the following:

1. **Selecting training and testing datasets.** The MovieLens public dataset contains 943 users and 1,682 items. We randomly divided the dataset into a training set (900 users) and a testing set (43 users). For the Jester dataset, we randomly selected 5,000 users from the dataset for training data and 500 users for testing data; for the MovieLens million dataset, we randomly selected 3,000 users for training and 300 users for testing.
2. **Prediction for active users.** For each active user selected randomly from the testing dataset, we randomly select an item and use our randomized-perturbation-based scheme and the original algorithm, respectively, to predict the ratings on this item for this active user. We then calculate the difference of these two ratings. We run this prediction procedure for 100 times and calculate the mean absolute error and standard deviation of the errors.

We hypothesize that the privacy and accuracy depend on several factors including the selection of α , the γ values that affect the range of the random numbers (α), the total number of users, and the total number of items. Therefore, we conducted the following experiments:

- *Fixed α vs. Random α .* We experimented with two different strategies regarding the range of the random numbers (which is used to disguise users' private data): one requires users to select the random numbers from a fixed range $[-\beta, \beta]$, where β is a constant number; the other strategy requires users to select the random numbers from a changing range $[-\alpha, \alpha]$, where α is a random number ranging from $[0, \beta]$.
- *Dense Datasets vs. Sparse Datasets:* We conducted experiments on both dense and sparse datasets. Although Jester and MovieLens are sparse datasets, we converted them into dense datasets. There are different methods to deal with missing ratings including using the user mean, the item mean, or the overall mean vote [5]. We used the item mean votes [16] to convert sparse training datasets into dense datasets.
- *The Selection of γ .* The value of γ decides the range of the random numbers. It is critical to the performance of our scheme. We conducted experiments for various γ values, including 95%, 85%, 75%, and 50%, but we only showed the results when γ is 95% and 50%.
- *Sets of Experiments:* We hypothesize that the randomized perturbation techniques give more accurate results when the number of users and/or items increases. To test this hypothesis, we conducted three sets of experiments: for the first and the second sets, we try to keep the number of users the same while changing the number of items; for the second and the third sets, we try to keep the number of items the same while changing the number of users. Since the third set needs to involve a large number of users, we conduct the third set of experiments on dense datasets (we need to convert the sparse training datasets into dense sets using the item mean votes).

4.4 Experimental Results

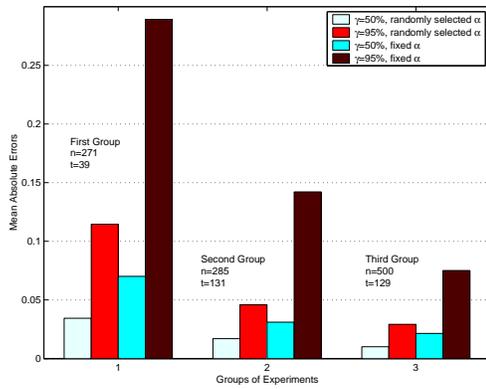
To evaluate our proposed schemes, we have conducted several experiments; we then compared the prediction results from randomized data (using our schemes) with the results from the original data. Fig. 3, Fig. 4, and Fig. 5 depict our results on three different datasets.

For both MovieLens datasets (MovieLens public data and MovieLens million data), when we choose $\gamma = 95\%$ and the fixed- α scheme, the mean absolute error in our experiments is below 0.29. Since the rating range is from 1 to 5, $MAE = 0.29$ indicates our results are very close to the results generated from the original data. As we can see from both Fig. 3 and Fig. 4, the results get much better when we use the random- α scheme to generate the random numbers; the results also get better when we choose a smaller γ value, or increase the value of n (the total number of users) and t (the total number of items). We will discuss the effects of those changes later.

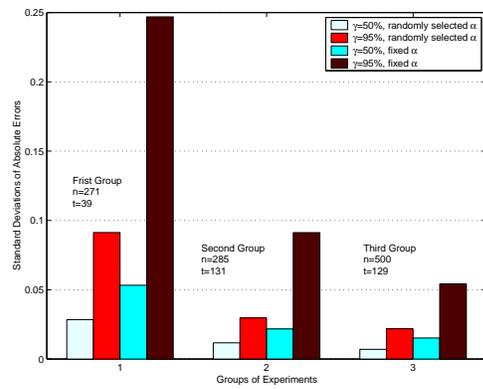
For the Jester dataset, when we choose $\gamma = 95\%$ and the fixed- α scheme, the mean absolute error in our experiments can reach as high as 1.4. However, in Jester dataset, the rating scale is from -10 to 10; an error of 1.4 is equivalent to 0.28 in a 1–5 scale. Therefore, the results for Jester dataset is similar to those from the MovieLens datasets.

We now show how the α selection scheme, the value of γ , and the value of n and t affect the accuracy of the results.

- *Value of n and t :* Results on all three datasets show that the accuracy is improved when either of the following increases: the total number of users (n) or the total number of items (t)

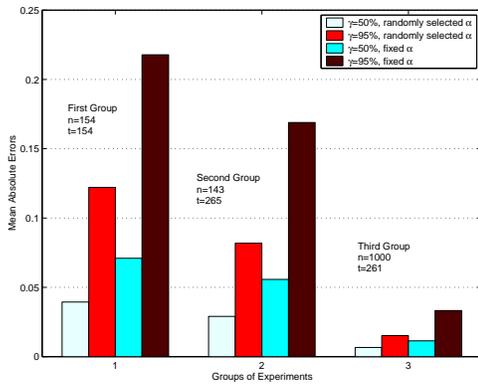


(a) The Mean Absolute Errors (rating range: 1-5)

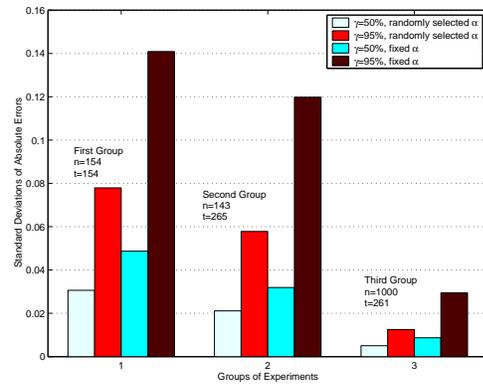


(b) Standard Deviations of the Absolute Errors

Figure 3: MovieLens Public Dataset



(a) The Mean Absolute Errors (rating range: 1-5)



(b) Standard Deviations of the Absolute Errors

Figure 4: MovieLens Million Dataset

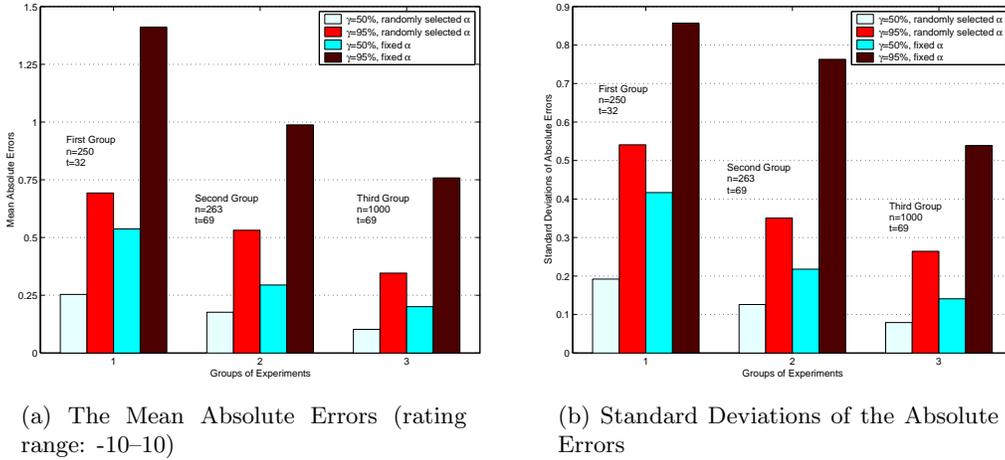


Figure 5: Jester Dataset

involved in the prediction calculation. The improvement trend can be observed from the differences of Group 1, 2, and 3. The reason of the improvement can be quite straightforwardly explained: Our scheme is based on the fact that the sum and the scalar product of the perturbed data are approximately the same as those of the original data. As we know, the more data we have for these two computations, the more accurate the approximation will be. In our scheme, increasing n and t is equivalent to increasing the amount of data involved in the sum and the scalar product computations.

- *Selection of γ Value:* Our results clearly show that the range of the random numbers affects the accuracy of our scheme. For example, in all of our result figures, results for $\gamma = 50\%$ are always better than those for $\gamma = 95\%$. The range of random numbers is $[-0.67, 0.67]$ when $\gamma = 50\%$, whereas it is $[-1.95, 1.95]$ when $\gamma = 95\%$. As we know, when the range is small, the randomness also becomes smaller; thus the accuracy can be improved.
- *Fixed α vs. Random α :* Our results show that the random- α scheme yields better performance than the fixed- α scheme. For example, in Fig. 3, when $\gamma = 95\%$, the error for the random- α scheme is almost half of the error for the fixed- α scheme. The reason for this phenomenon is that when the range of the random numbers is random from $[0, \beta]$, the distribution of all the random numbers is not uniform in range $[-\beta, \beta]$; the probability of choosing a number near 0 is larger than the probability of choosing a number near β or $-\beta$. On the other hand, in the fixed- α scheme, the generated random numbers are uniformly distributed in range $[-\beta, \beta]$. As we have already discussed, the bigger the random number, the less accurate the result. Therefore, choosing randomly selected α as the range for the random numbers produces better results than the fixed- α scheme.

5 Conclusion and Future Work

We have presented a solution to the privacy-preserving collaborative filtering problem using the randomized perturbation scheme. Our solution makes it possible for servers to collect private data from users for collaborative filtering purposes without compromising users' privacy requirements.

Our experiments have shown that our solution can achieve accurate prediction compared to the prediction based on the original data.

We believe that accuracy of our scheme can be further improved if more aggregate information is disclosed along with the disguised data, especially those aggregate information whose disclosure does not compromise much of users' privacy. These types of information include mean, standard deviation, distribution, true data in a permuted order, etc. We will study how these kinds of aggregate data disclosure affects the accuracy and the privacy.

We will also study other collaborative filtering algorithms, and investigate whether we can extend our techniques to other memory-based and model-based algorithms to achieve privacy-preserving collaborative filtering.

References

- [1] Anonymizer.com: <http://www.anonymizer.com>.
- [2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD on Management of Data*, pages 439–450, Dallas, TX USA, May 15 - 18 2000.
- [3] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, Madison, WI, July 1998.
- [4] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, May 2002.
- [5] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, Tampere, Finland, August 2002.
- [6] L. F. Cranor, J. Reagle, and M. S. Ackerman. Beyond concern: Understanding net users' attitudes about online privacy. Technical report, AT&T Labs-Research, April 1999. Available from <http://www.research.att.com/library/trs/TRs/99/99.4.3/report.htm>.
- [7] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. 35(12):61–70, 1992.
- [8] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- [9] D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg. Jester 2.0: A new linear-time collaborative filtering algorithm applied to jokes. In *Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, CA.
- [10] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.
- [11] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 194–201, 1995.
- [12] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: Applying collaborative filtering to usenet news. In *Communications of the ACM*, pages 77–87, March 1997.
- [13] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model based approach. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 473–480, Morgan Kaufmann, San Francisco, 2000.

- [14] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transaction. *ACM Transactions on Information and System Security*, 1(1):Pages 66–92, 1998.
- [15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.
- [16] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system—a case study. In *ACM WebKDD 2000 Web Mining for E-commerce Workshop*, 2000.
- [17] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating ”word of mouth”. In *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.
- [18] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of 1997 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 5-7 1997.
- [19] A. F Westin. Freebies and privacy. Technical report, Opinion Research Corporation, July 1999. Available from <http://www.privacyexchange.org/iss/surveys/sr990714.html>.