Chapter 11

PRIVACY PROTECTION IN COLLABORATIVE FILTERING BY ENCRYPTED COMPUTATION

Wim F.J. Verhaegh, Aukje E.M. van Duijnhoven, Pim Tuyls, and Jan Korst

Abstract We present a method to protect users' privacy in collaborative filtering by performing the computations on encrypted data. We focus on the commonly-used memory-based approach, and show that the two main steps in collaborative filtering, being the determination of similarities and the prediction of ratings, can be performed on encrypted profiles. We discuss both user-based and item-based collaborative filtering, and for a number of variants of the similarity measures and prediction formulas described in literature, we show how they can be computed using encrypted data only. Although we consider collaborative filtering in this chapter, the techniques of comparing profiles using encrypted data only is useful in a much wider range of applications.

Keywords Collaborative filtering, privacy, encryption.

11.1 Introduction

One of the key characteristics of ambient intelligence [Aarts & Marzano, 2003] is personalization, which ensures tailored applications and services to users. In order to realize personalization, electronic systems need user profiles, indicating the specific characteristics and preferences of users. If such personalization is realized by a stand-alone device, there is no issue, but if the personalization is offered by a service on the internet, the privacy of users may be at stake. Although most internet services, such as Amazon, have a privacy statement on their web site, users may be reluctant to give their personal data away, for several reasons. First, they may not trust every server. Secondly, they may trust the server, but do not want to run the risk that it gets hacked. Thirdly, the server may be reliable, but if it goes bankrupt, the user profiles represent valuable information that may be sold to third parties. Of course the privacy concern depends on the kind of data, e.g. preferences for books may be less sensitive information than users' medical records.

Wim F.J. Verhaegh et al. (Eds.), Intelligent Algorithms in Ambient and Biomedical Computing, 169-184.

In order to protect the users' privacy, we investigate the possibilities that encryption techniques offer. The idea is that a user only releases personal information in an encrypted form, and that all the computations necessary for the personalization are done on the encrypted data. In the end, the user will receive an encrypted personalization result, which he can decrypt and use.

In this chapter we show how the above can be realized for recommendation services based on collaborative filtering [Herlocker et al., 1999; Shardanand & Maes, 1995], as a first personalization application that we select. Collaborative filtering is a well-known technique to recommend e.g. new music or books to users, and helps users in coping with the overload of content that is available through the internet. Based on a user's likes and dislikes for previously encountered content, it estimates to what extent he would like other content that is available. To this end, collaborative filtering uses the preferences of a community of users.

We can distinguish two global types of collaborative filtering approaches: memory-based [Herlocker et al., 1999] and model-based [Canny, 2002]. Memory-based collaborative filtering is the most commonly used approach. In this approach, which is a *lazy learning* approach in machine learning terms, the preferences (in the form of ratings for content) of a community of users are collected at a web server. Then, a similarity measure is computed between each pair of users based on the content they jointly rated. Next, recommendations for a particular user can be made by considering users that are similar to him, and checking for content that they liked but that has not yet been rated by the user or that is not yet in the user's collection.

Model-based approaches pursue a more active learning strategy. First, the collected preference data is processed to build a model of the users' profile space. For instance, Canny [2002] describes a factor-analysis approach, which first distills a basis of user preference profiles and expresses the individual users' profiles in terms of this basis. Next, this model is used to make predictions.

As mentioned, we want to develop a system that prevents any information about a user's preferences to become known to others. This not only means that we want to keep the user's ratings for items secret, but even the information of what items he has rated. Furthermore, we do not even want to reveal this kind of information anonymously, as we do not want to run the risk that the identity of the user is traced back somehow, after which his data is in the clear. Finally, as similarities between users also give information about a user's preferences, we also want to keep this data secret.

In addition to the above requirements from a user's perspective, we add the requirement that the server should maintain some control over the service, i.e., it should not be possible for a user to trivially retrieve valuable gathered data to set up a recommendation service too.

Whereas Canny [2002] focuses on model-based collaborative filtering, we discuss in this chapter how the more commonly used memory-based collaborative filtering technique can be performed on encrypted data. This holds for all variants of similarity measures and prediction formulas that we describe.

The remainder of this chapter is organized as follows. First, in Section 11.2 we discuss the procedures and formulas behind memory-based collaborative filtering, where we distinguish user-based and item-based approaches. Next, in Section 11.3 we briefly describe the proposed encryption system and its beneficial properties. Then, we discuss how the above requirements can be met, by describing how to perform the collaborative-filtering computations on encrypted data for the user-based and item-based approaches in Sections 11.4 and 11.5, respectively.

Although we focus in this chapter on encryption of preference information in collaborative filtering, the techniques we present are applicable in a much broader context, as many more ambient intelligence applications will use some form of matching profiles. Also these applications may be much better accepted by users if private information can be protected. We will however not elaborate on this.

11.2 Memory-based collaborative filtering

Most memory-based collaborative filtering approaches work by first determining similarities between users, by comparing their jointly rated items. Next, these similarities are used to predict the rating of a user for a particular item, by interpolating between the ratings of the other users for this item. Typically, all computations are performed by the server, upon a user request for a recommendation.

Next to the above approach, which is called a *user-based* approach, one can also follow an *item-based* approach. Then, first similarities are determined between items, by comparing the ratings they have gotten from the various users, and next the rating of a user for an item is predicted by interpolating between the ratings that this user has given for the other items.

Before discussing the formulas underlying both approaches, we first introduce some notation. We assume a set U of users and a set I of items. Whether a user $u \in U$ has rated item $i \in I$ is indicated by a boolean variable b_{ui} which equals one if the user has done so and zero otherwise. In the former case, also a rating r_{ui} is given, e.g. on a scale from 1 to 5. The set of users that have rated an item i is denoted by U_i, and the set of items that have been rated by a user u is denoted by I_u.

11.2.1 The user-based approach

User-based algorithms are probably the oldest and most widely used collaborative filtering algorithms [Breese, Heckerman & Kadie, 1998; Herlocker et al., 1999; Resnick et al., 1994; Sarwar et al., 2000]. As described above, there are two main steps: determining similarities and calculating predictions. For both we discuss commonly used formulas, of which we show later that they all can be computed on encrypted data.

Similarity measures. Quite a number of similarity measures have been presented in the literature before. We distinguish three kinds: correlation measures, distance measures, and counting measures.

Correlation measures. A common similarity measure used in literature is the so-called *Pearson correlation coefficient* (see e.g. [Sarwar et al., 2000]), given by[1]

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_u \cap I_v} (r_{vi} - \bar{r}_v)^2}}, \tag{11.1}$$

where $\bar{r}_u$ denotes the average rating of user u for the items he has rated. The numerator in this equation gets a positive contribution for each item that is either rated above average by both users u and v, or rated below average by both. If one user has rated an item above average and the other user below average, we get a negative contribution. The denominator in the equation normalizes the similarity, to fall in the interval $[-1,1]$, where a value 1 indicates complete correspondence and -1 indicates completely opposite tastes.

Related similarity measures are obtained by replacing $\bar{r}_u$ in (11.1) by the middle rating (e.g. 3 if using a scale from 1 to 5) or by zero. In the latter case, the measure is called vector similarity or cosine, and if all ratings are non-negative, the resulting similarity value will then lie between 0 and 1.

Distance measures. Another type of measures is given by distances between two users' ratings, such as the mean-square difference [Shardanand & Maes, 1995] given by

$$\frac{\sum_{i \in I_u \cap I_v} (r_{ui} - r_{vi})^2}{|I_u \cap I_v|}, \tag{11.2}$$

or the normalized Manhattan distance [Aggarwal et al., 1999] given by

$$\frac{\sum_{i \in I_u \cap I_v} |r_{ui} - r_{vi}|}{|I_u \cap I_v|}. \tag{11.3}$$

[1] Note that if $I_u \cap I_v = \emptyset$, then the similarity $s(u,v)$ is undefined, and it should be discarded in the prediction formulas (11.9)–(11.11).

Such a distance is zero if the users rated their overlapping items identically, and larger otherwise. A simple transformation converts a distance into a measure that is high if users' ratings are similar and low otherwise.

Counting measures. Counting measures are based on counting the number of items that two users rated (nearly) identically. A simple counting measure is the majority voting measure [Nakamura & Abe, 1998] of the form

$$s(u,v) = (2-\gamma)^{c_{uv}}\gamma^{d_{uv}}, \tag{11.4}$$

where γ is chosen between 0 and 1, $c_{uv} = |\{i \in I_u \cap I_v \mid r_{ui} \approx r_{vi}\}|$ gives the number of items rated 'the same' by u and v, and $d_{uv} = |I_u \cap I_v| - c_{uv}$ gives the number of items rated 'differently'. The relation $\approx$ may here be defined as exact equality, but also nearly-matching ratings may be considered sufficiently equal.

Another counting measure is given by the weighted kappa statistic [Cohen, 1968], which is defined as the ratio between the observed agreement between two users and the maximum possible agreement, where both are corrected for agreement by chance. More formally, the measure is given by

$$s(u,v) = \frac{o_{uv} - e_{uv}}{1 - e_{uv}}. \tag{11.5}$$

Here, o_{uv} is the observed fraction of agreement, given by

$$o_{uv} = \frac{\sum_{i \in I_u \cap I_v} w(r_{ui}, r_{vi})}{|I_u \cap I_v|}, \tag{11.6}$$

where weights $w(x,y)$, with $0 \leq w(x,y) = w(y,x) \leq 1$ and $w(x,x) = 1$, indicate the degree of correspondence between ratings x and y. The offset e_{uv} is the expected fraction of agreement, and is given by

$$e_{uv} = \sum_{x \in X} \sum_{y \in X} p_u(x) p_v(y) w(x,y), \tag{11.7}$$

where X is the set of possible ratings, and $p_u(x)$ is the fraction of items that u has given a rating x, i.e.,

$$p_u(x) = \frac{|\{i \in I_u \mid r_{ui} = x\}|}{|I_u|}. \tag{11.8}$$

Prediction formulas. The second step in collaborative filtering is to use the similarities to compute a prediction for a certain user-item pair. Also for this step several variants exist. For all formulas, we assume that there are users that have rated the given item; otherwise no prediction can be made.

Weighted sums. The first prediction formula, as used by Herlocker et al. [1999], is given by

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in U_i} s(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in U_i} |s(u,v)|}. \tag{11.9}$$

So, the prediction is the average rating of user u plus a weighted sum of deviations from the averages. In this sum, all users are considered that have rated item i. Alternatively, one may restrict them to users that also have a sufficiently high similarity to user u, i.e., we sum over all users in $U_i(t) = \{v \in U_i \mid s(u,v) \geq t\}$ for some threshold t.

An alternative, somewhat simpler prediction formula is given by

$$\hat{r}_{ui} = \frac{\sum_{v \in U_i} s(u,v) r_{vi}}{\sum_{v \in U_i} |s(u,v)|}. \tag{11.10}$$

Note that if all ratings are positive, then this formula only makes sense if all similarity values are non-negative, which may be realized by choosing a non-negative threshold.

Maximum total similarity. A second type of prediction formula is given by choosing the rating that maximizes a kind of total similarity, as is done in the majority voting approach, given by

$$\hat{r}_{ui} = \arg\max_{x \in X} \sum_{v \in U_i^x} s(u,v), \tag{11.11}$$

where $U_i^x = \{v \in U_i \mid r_{vi} \approx x\}$ is the set of users that gave item i a rating similar to value x. Again, the relation $\approx$ may be defined as exact equality, but also nearly-matching ratings may be allowed. Also in this formula one may use $U_i(t)$ instead of U_i to restrict oneself to sufficiently similar users.

Time complexity. The time complexity of user-based collaborative filtering is as follows.

For the first step, there are $O(|U|^2)$ pairs of users between which a similarity has to be computed. Similarity measures (11.1)–(11.4) require sums of $O(|I|)$ items, hence giving a total time complexity of $O(|U|^2|I|)$ to determine all similarities. The computation of the weighted kappa statistic (11.5) requires per pair of users $O(|I|)$ steps to compute (11.6) and $O(|X|^2)$ steps to compute (11.7), where for the latter one needs to compute (11.8) in $O(|I|)$ steps once per user and per value in X. So, this gives a total time complexity of $O(|U|^2|I| + |U|^2|X|^2 + |U||I||X|)$ for the kappa statistic. As $|X|$ is typically bounded by a small constant, say between 5 and 10, this reduces to the same time complexity $O(|U|^2|I|)$ as for the other measures.

If for all users all items with a missing rating are to be given a prediction, then this requires $O(|U||I|)$ predictions to be computed. Prediction formulas (11.9) and (11.10) can be computed in $O(|U|)$ steps, where (11.9) requires $O(|I|)$ steps once per user u to compute his average ratings $\bar{r}_u$. So, this gives a total complexity of $O(|U|^2|I|)$ to compute all predictions. Prediction formula (11.11) however requires $O(|U||X|)$ steps per prediction, thereby giving a total complexity of $O(|U|^2|I||X|)$ to compute all predictions. Again, if $|X|$ is bounded by a constant, this time complexity reduces to $O(|U|^2|I|)$.

11.2.2 The item-based approach

As mentioned, item-based algorithms [Karypis, 2001; Sarwar et al., 2001] first compute similarities between items, e.g. by using a similarity measure

$$s(i,j) = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_u)^2 \sum_{u \in U_i \cap U_j} (r_{uj} - \bar{r}_v)^2}}. \tag{11.12}$$

Note that the exchange of users and items as compared to (11.1) is not complete, as still the average rating $\bar{r}_u$ is subtracted from the ratings. The reason to do so is that this subtraction compensates for the fact that some users give higher ratings than others, and there is no need for such a correction for items.

The standard item-based prediction formula to be used for the second step is given by

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in I_u} s(i,j)(r_{uj} - \bar{r}_j)}{\sum_{j \in I_u} |s(i,j)|}. \tag{11.13}$$

The other similarity measures and prediction formulas we presented for the user-based approach can in principle also be turned into item-based variants, but we will not show them here.

Also in the time complexity for item-based collaborative filtering the roles of users and items interchange as compared to the user-based approach, as expected. For the first step, $O(|I|^2)$ similarity measures (11.12) have to be computed, each of which takes $O(|U|)$ steps. The prediction formula (11.13) requires $O(|I|)$ steps for each user and each item, where the average rating $\bar{r}_i$ takes $O(|U|)$ steps once per item i. As a result, the total time complexity is given by $O(|U||I|^2)$.

If the number $|U|$ of users is much larger than the number $|I|$ of items, the time complexity of the item-based approach is favorable over that of user-based collaborative filtering. Another advantage in this case is that the similarities are generally based on more elements, which gives more reliable measures. A further advantage of item-based collaborative filtering, as argued by Sarwar et al. [2001], is that correlations between items may be more stable than correlations between users, but we will not elaborate on this.

11.3 Encryption

In the next section we show how the presented formulas for collaborative filtering can be computed on encrypted ratings. Before doing so, we present the encryption system we use, and the specific properties it possesses that allow for the computation on encrypted data.

11.3.1 A public-key cryptosystem

The cryptosystem we use is the public-key cryptosystem presented by Paillier [1999]. We will not describe it in full detail, for which we refer to the chapter, but we briefly describe how data is encrypted.

First, encryption keys are generated. To this end, two large primes p and q are chosen randomly, and we compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. Furthermore, a *generator* g is computed from p and q (for details, see [Paillier, 1999]). Now, the pair (n, g) forms the public key of the cryptosystem, which is sent to everyone, and λ forms the private key, to be used for decryption, which is kept secret.

Next, a sender who wants to send a message $m \in \mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ to a receiver with public key (n, g) computes a ciphertext $\varepsilon(m)$ by

$$\varepsilon(m) = g^m r^n \bmod n^2, \tag{11.14}$$

where r is a number randomly drawn from $\mathbb{Z}_n^* = \{x \in \mathbb{Z} \mid 0 < x < n \wedge \gcd(x, n) = 1\}$. This r prevents decryption by simply encrypting all possible values of m (in case it can only assume a few values) and comparing the end result. The Paillier system is hence called a *randomized* encryption system.

Decryption of a ciphertext $c = \varepsilon(m)$ is done by computing

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n,$$

where $L(x) = (x-1)/n$ for any $0 < x < n^2$ with $x \equiv 1 \pmod{n}$. During decryption, the random number r cancels out.

Note that in the above cryptosystem the messages m are integers. Nevertheless, rational values are possible by multiplying them by a sufficiently large number and rounding off [Fouque, Stern & Wackers, 2002]. For instance, if we want to use messages with two decimals, we simply multiply them by 100 and round off. Usually, the range $\mathbb{Z}_n$ is large enough to allow for this multiplication.

11.3.2 Properties

The above presented encryption scheme has the following nice properties. The first one is that

$$\varepsilon(m_1)\varepsilon(m_2) \equiv g^{m_1} r_1^n g^{m_2} r_2^n \equiv g^{(m_1+m_2)}(r_1 r_2)^n \equiv \varepsilon(m_1+m_2) \pmod{n^2},$$

which allows us to compute sums on encrypted data. Secondly,

$$\varepsilon(m_1)^{m_2} \equiv (g^{m_1} r_1^n)^{m_2} \equiv g^{m_1 m_2}(r_1^{m_2})^n \equiv \varepsilon(m_1 m_2) \pmod{n^2},$$

which allows us to compute products on encrypted data. An encryption scheme with these two properties is called a *homomorphic* encryption scheme. The Paillier system is one homomorphic encryption scheme, but more ones exist.

We can use the above properties to calculate sums of products, as required for the similarty measures and predictions, using

$$\prod_j \varepsilon(a_j)^{b_j} \equiv \prod_j \varepsilon(a_j b_j) \equiv \varepsilon(\sum_j a_j b_j) \pmod{n^2}. \tag{11.15}$$

So, using this, two users a and b can compute an inner product between a vector of each of them in the following way. User a first encrypts his entries a_j and sends them to b. User b then computes (11.15), as given by the left-hand term, and sends the result back to a. User a next decrypts the result to get the desired inner product. Note that neither user a nor user b can observe the data of the other user; the only thing user a gets to know is the inner product.

A final property we want to mention is that

$$\varepsilon(m_1)\varepsilon(0) \equiv g^{m_1} r_1^n g^0 r_2^n \equiv g^{m_1}(r_1 r_2)^n \equiv \varepsilon(m_1) \pmod{n^2}.$$

This action, which is called *(re)blinding*, can be used also to avoid a trial-and-error attack as discussed above, by means of the random number $r_2 \in \mathbb{Z}_n^*$. We will use this in Section 11.4.2.

11.3.3 A threshold version of the cryptosystem

The Paillier cryptosystem can also be implemented in a threshold version [Fouque, Poupard & Stern, 2000], in which the decryption key is shared among a number l of users, and a ciphertext can only be decrypted if more than a threshold t of users cooperate. In this version, the generation of the keys is somewhat more complicated, as well as the decryption mechanism. Without further going into details, for which we refer to Fouque, Poupard & Stern [2000], we briefly discuss the decryption procedure in the threshold cryptosystem. For this, first a subset of at least $t+1$ users is chosen that will be involved in the decryption. Next, each of these users receives the ciphertext and computes a decryption share, using his own share of the key. Finally, these decryption shares are combined to compute the original message. As long as at least

$t+1$ users have combined their decryption share, the original message can be reconstructed.

11.4 Encrypted user-based algorithm

Having all ingredients in place, we now explain how memory-based collaborative filtering can be performed on encrypted data, in order to compute a prediction $\hat{r}_{ui}$ for a certain user u and item i. Note that although the computations are done on encrypted data, the outcome is of course identical to that of the original collaborative filtering algorithm.

We consider a setup as depicted in Figure 11.1, where user u communicates with other users v through a server. Furthermore, each user has generated his own key, and has published the public part of it. As we want to compute a prediction for user u, the steps below will use the keys of u.

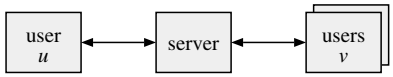

Figure 11.1. The setup for the user-based algorithm.

11.4.1 Computing similarities on encrypted data

First we take the similarity computation step, for which we start with the Pearson correlation given in (11.1). Although we already explained in Section 11.3 how to compute an inner product on encrypted data, we have to resolve the problem that the iterator i in the sums in (11.1) only runs over $I_u \cap I_v$, and this intersection is not known to either user. Therefore, we first introduce

$$q_{ui} = \begin{cases} r_{ui} - \bar{r}_u & \text{if } b_{ui} = 1, \text{ i.e., user } u \text{ rated item } i, \\ 0 & \text{otherwise,} \end{cases}$$

and rewrite (11.1) into

$$s(u,v) = \frac{\sum_{i \in I} q_{ui} q_{vi}}{\sqrt{\sum_{i \in I} q_{ui}^2 b_{vi} \sum_{i \in I} q_{vi}^2 b_{ui}}}. \tag{11.16}$$

The idea that we used is that any $i \notin I_u \cap I_v$ does not contribute to any of the three sums because at least one of the factors in the corresponding term will be zero. Hence, we have rewritten the similarity into a form consisting of three inner products, each between a vector of u and one of v.

The protocol now runs as follows. First, user u calculates encrypted entries $\varepsilon(q_{ui})$, $\varepsilon(q_{ui}^2)$, and $\varepsilon(b_{ui})$ for all $i \in I$, using (11.14), and sends them to the server. The server forwards these encrypted entries to each other

user $v_1,\ldots,v_m$. Next, each user v_j, $j=1,\ldots,m$, computes $\varepsilon(\sum_{i\in I} q_{ui}q_{v_ji})$, $\varepsilon(\sum_{i\in I} q_{ui}^2 b_{v_ji})$, and $\varepsilon(\sum_{i\in I} q_{v_ji}^2 b_{ui})$, using (11.15), and sends these three results back to the server, which forwards them to user u. User u can decrypt the total of $3m$ results and compute the similarities $s(u,v_j)$, for all $j=1,\ldots,m$. Note that user u now knows similarity values with the other m users, but he need not know who each user $j=1,\ldots,m$ is. The server, on the other hand, knows who each user $j=1,\ldots,m$ is, but it does not know the similarity values.

For the other similarity measures, we can also derive computation schemes using encrypted data only. For the mean-square distance, we can rewrite (11.2) into

$$\frac{\sum_{i\in I_u\cap I_v}(r_{ui}^2-2r_{ui}r_{vi}+r_{vi}^2)}{|I_u\cap I_v|}=\frac{\sum_{i\in I} r_{ui}^2 b_{vi}+2\sum_{i\in I} r_{ui}(-r_{vi})+\sum_{i\in I} r_{vi}^2 b_{ui}}{\sum_{i\in I} b_{ui}b_{vi}}, \tag{11.17}$$

where we additionally define $r_{ui}=0$ if $b_{ui}=0$ in order to have well-defined values. So, this distance measure can also be computed by means of four inner products.

The computation of normalized Manhattan distances is somewhat more complicated. Given the set X of possible ratings, we first define for each $x\in X$,

$$b_{ui}^x=\begin{cases}1 & \text{if } b_{ui}=1\wedge r_{ui}=x,\\ 0 & \text{otherwise,}\end{cases}$$

and

$$a_{ui}^x=\begin{cases}|r_{ui}-x| & \text{if } b_{ui}=1,\\ 0 & \text{otherwise.}\end{cases}$$

Now, (11.3) can be rewritten into

$$\frac{\sum_{i\in I}\sum_{x\in X} b_{ui}^x a_{vi}^x}{\sum_{i\in I} b_{ui}b_{vi}}=\frac{\sum_{x\in X}\sum_{i\in I} b_{ui}^x a_{vi}^x}{\sum_{i\in I} b_{ui}b_{vi}}. \tag{11.18}$$

So, the normalized Manhattan distance can be computed from $|X|+1$ inner products. Furthermore, for the numerator a user v can compute $\prod_{x\in X}\varepsilon(\sum_{i\in I} b_{ui}^x a_{vi}^x)\equiv\varepsilon(\sum_{x\in X}\sum_{i\in I} b_{ui}^x a_{vi}^x)$, and send this result, together with the encrypted denominator, back to user u.

The majority-voting measure can also be computed in the above way, by defining

$$a_{ui}^x=\begin{cases}1 & \text{if } b_{ui}=1\wedge r_{ui}\approx x,\\ 0 & \text{otherwise.}\end{cases} \tag{11.19}$$

Then, c_{uv} used in (11.4) is given by

$$c_{uv}=\sum_{x\in X}\sum_{i\in I} b_{ui}^x a_{vi}^x, \tag{11.20}$$

which can again be computed in a way as described above. Furthermore,

$$d_{uv} = \sum_{i \in I} b_{ui} b_{vi} - c_{uv}.$$

Finally, we consider the weighted kappa measure (11.5). Again, o_{uv} can be computed by defining

$$a^x_{ui} = \begin{cases} w(x, r_{ui}) & \text{if } b_{ui} = 1, \\ 0 & \text{otherwise,} \end{cases}$$

and then calculating

$$o_{uv} = \frac{\sum_{x \in X} \sum_{i \in I} b^x_{ui} a^x_{vi}}{\sum_{i \in I} b_{ui} b_{vi}}. \tag{11.21}$$

Furthermore, e_{uv} can be computed in an encrypted way if user u encrypts $p_u(x)$ for all $x \in X$ and sends them to each other user v, who can then compute

$$\prod_{x \in X} \prod_{y \in Y} \varepsilon(p_u(x))^{p_v(y) w(x,y)} \equiv \varepsilon(e_{uv}), \tag{11.22}$$

and send this back to u for decryption.

11.4.2 Computing predictions on encrypted data

For the second step of collaborative filtering, user u can calculate a prediction for item i in the following way. First, we rewrite the quotient in (11.9) into

$$\frac{\sum_{v \in U} s(u,v) q_{vi}}{\sum_{v \in U} |s(u,v)| b_{vi}}. \tag{11.23}$$

So, first user u encrypts $s(u,v_j)$ and $|s(u,v_j)|$ for each other user v_j, $j = 1, \ldots, m$, and sends them to the server. The server then forwards each pair $\varepsilon(s(u,v_j)), \varepsilon(|s(u,v_j)|)$ to the respective user v_j, who computes $\varepsilon(s(u,v_j))^{q_{v_j i}} \varepsilon(0) \equiv \varepsilon(s(u,v_j) q_{v_j i})$ and $\varepsilon(|s(u,v_j)|)^{b_{v_j i}} \varepsilon(0) \equiv \varepsilon(|s(u,v_j)| b_{v_j i})$, where he uses reblinding to prevent the server from getting knowledge from the data going back and forth to user v_j by trying a few possible values. Each user v_j next sends the results back to the server, which then computes

$$\prod_{j=1}^{m} \varepsilon(s(u,v_j) q_{v_j i}) \equiv \varepsilon(\sum_{j=1}^{m} s(u,v_j) q_{v_j i})$$

and

$$\prod_{j=1}^{m} \varepsilon(|s(u,v_j)| b_{v_j i}) \equiv \varepsilon(\sum_{j=1}^{m} |s(u,v_j)| b_{v_j i}),$$

and sends these two results back to user u. User u can then decrypt these messages and use them to compute the prediction. The simple prediction formula of (11.10) can be handled in a similar way.

The maximum total similarity prediction as given by (11.11) can be handled as follows. First, we rewrite

$$\sum_{v \in U_i^x} s(u,v) = \sum_{j=1}^{m} s(u,v_j) a_{v_j i}^x, \tag{11.24}$$

where $a_{v_j i}^x$ is as defined by (11.19). Next, user u encrypts $s(u,v_j)$ for each other user v_j, $j = 1,\ldots,m$, and sends them to the server. The server then forwards each $\varepsilon(s(u,v_j))$ to the respective user v_j, who computes $\varepsilon(s(u,v_j))^{a_{v_j i}^x} \varepsilon(0) \equiv \varepsilon(s(u,v_j) a_{v_j i}^x)$, for each rating $x \in X$, using reblinding. Next, each user v_j sends these $|X|$ results back to the server, which then computes

$$\prod_{j=1}^{m} \varepsilon(s(u,v_j) a_{v_j i}^x) \equiv \varepsilon(\sum_{j=1}^{m} s(u,v_j) a_{v_j i}^x), \tag{11.25}$$

for each $x \in X$, and sends the $|X|$ results to user u. Finally, user u decrypts these results and determines the rating x that has the highest result.

11.4.3 Time complexity revisited

The effect of encryption on the time complexity of computing the similarities and predictions, is as follows.

The time complexity to compute (11.16) and (11.17) is determined by the server, which has to forward for each user u, $O(|I|)$ encrypted items to all users v_j. This can be done in a total of $O(|U|^2|I|)$ steps, which equals the time complexity of the unencrypted case. For (11.18) and (11.20), $O(|I||X|)$ encrypted items have to be forwarded for each user u to all users v_j, giving a total of $O(|U|^2|I||X|)$ steps, which is a factor $O(|X|)$ more than in the unencrypted case. Finally, for the weighted kappa statistic we first need $O(|I||X|)$ steps per pair u,v to compute o_{uv} in (11.21), and secondly we need to compute e_{uv}. The latter takes for each u, $O(|U||X|)$ steps for the server to forward the encrypted items $\varepsilon(p_u(x))$ to all other users v_j, and $O(|X|^2)$ steps for all users v_j to compute (11.22). Note that the latter users can do so in parallel, and hence the total time complexity of computing all kappa statistics is given by $O(|U|^2|I||X| + |U|^2|X| + |U||X|^2)$.

For the prediction formulas, encryption does not have an effect on the time complexity. Formulas (11.23) and the encrypted version of (11.10) still require $O(|U|)$ steps per user u and item i for the work done by the user u himself and the server, giving a total time complexity of $O(|U|^2|I|)$. The time complexity to compute (11.24) in an encrypted way is determined by the server, which has to calculate $O(|X|)$ products (11.25) over $O(|U|)$ entries for each user u and item i, giving a total time complexity of $O(|U|^2|I||X|)$.

Although the time complexity is not much affected, we note that the run time of course will increase, because of the more demanding computations on

encrypted data. This overhead is determined by the length of the encryption key.

11.5 Encrypted item-based algorithm

Also item-based collaborative filtering can be done on encrypted data, using the threshold system of Section 11.3.3. The general working of the item-based approach is slightly different than the user-based approach, as first the server determines similarities between items, and next uses them to make predictions.

So, first the server considers each pair $i, j \in I$ of items to determine their similarity as given in (11.12). Again, we first have to resolve the problem that the iterator, u in this case, does not run over the entire set U. To this end, we rewrite (11.12) into

$$\frac{\sum_{u\in U} q_{ui}q_{uj}}{\sqrt{\sum_{u\in U} q_{ui}^2 b_{uj} \sum_{u\in U} q_{uj}^2 b_{ui}}}.$$

So, we have to compute three sums, where each user $u \in U$ can compute his own contributions to them. Each of the three sums can be computed in the following way. First, the users compute their contributions, encrypt them using the threshold encryption scheme, and send the encrypted contributions to the server. The server multiplies all encrypted contributions, and gets in this way an encrypted version of the respective sum. Next, the server sends this encrypted sum back to the users, who each compute their decryption share of it. These decryption shares are sent back to the server, and if the server has received more shares than the threshold, it can decrypt the respective sum.

Next, we consider the prediction formula (11.13). For this, first the server computes the average rating $\bar{r}_i$ of each item i, which can be written as

$$\bar{r}_i = \frac{\sum_{u\in U} r_{ui}}{\sum_{u\in U} b_{ui}},$$

where again we define $r_{ui} = 0$ if $b_{ui} = 0$. The two sums in this quotient can again be computed by the server in the same way as described above.

Secondly, user u can compute a prediction for item i if we rewrite (11.13) into

$$\frac{\sum_{j\in I} b_{uj}|s(i,j)|\bar{r}_i + \sum_{j\in I} r_{uj}s(i,j) + \sum_{j\in I} b_{uj}s(i,j)(-\bar{r}_j)}{\sum_{j\in I} b_{uj}|s(i,j)|}. \tag{11.26}$$

Inspecting the four sums in this equation reveals that they are inner products between vectors of u and of the server. Note that the similarity values and item averages are valuable data to the server, so it does not want to share this with the users. So, user u encrypts his entries b_{uj} and r_{uj} and sends them to the server. The server then computes the sums in the encrypted domain, and

sends the results back to u. User u then can decrypt the sums, and compute the prediction. Note that the server need not send all four encrypted sums back, as it can compute the denominator in (11.26) in the encrypted domain. It then only sends the encrypted denomimator and numerator back to u.

The time complexity of encrypted item-based collaborative filtering is not affected by encryption, apart from the computational overhead of computing with encrypted numbers. For the similarity measure, the server has to collect $O(|U|)$ contributions for three sums, for each pair i, j, giving a total time complexity of $O(|U||I|^2)$. Securely computing the average scores $\bar{r}_i$ requires $O(|U|)$ steps per item i, giving $O(|U||I|)$ in total. Finally, computing the predictions in an encrypted way takes $O(|I|)$ steps per user u and item i, giving also a total of $O(|U||I|^2)$.

11.6 Conclusion

We have shown how collaborative filtering can be done on encrypted data. In this way, sensitive information about a user's preferences, as discussed in the introduction, is kept secret, as it only leaves the user's system in an encrypted form. We have listed a number of variants of the similarity measures and prediction formulas described in literature, and showed for each of them how they can be computed using encrypted data only, without affecting their results.

Compared to the original set-up of collaborative filtering, the new set-up requires a more active role of the users' devices. This means that instead of a (single) server that runs an algorithm, we now have a system running a distributed algorithm, where all the nodes are actively involved in parts of the algorithm. The time complexity of the algorithm basically stays the same, except for an additional factor $|X|$ (typically between 5 and 10) for some similarity measures, and the overhead from computing with large encrypted numbers.

Although we showed that collaborative filtering can in principle be done on encrypted data, there are a few more issues to be resolved for a practical implementation. For instance, one should take into account the computational and communication overhead required due to the encryption and decryption of data. Furthermore, the system should be made robust against more complex forms of attacks, e.g. an attack where a user repeatedly computes similarities to other users, each time using a profile with only one item. These issues are topic of further research.

Although we only discussed collaborative filtering, the technique of computing similarities between profiles on encrypted data only is interesting for other applications as well, such as user matching and service discovery. In the vision of ambient intelligence, where much more (sensitive) profiling will be

used in the future, this may play a crucial role in getting these applications accepted by a wide audience. This is also topic of further research.

References

Aarts, E., and S. Marzano [2003]. *The New Everyday: Visions of Ambient Intelligence*. 010 Publishing, Rotterdam, The Netherlands.

Aggarwal, C., J. Wolf, K.-L. Wu, and P. Yu [1999]. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. *Proceedings ACM KDD'99 Conference*, pages 201-212.

Breese, J., D. Heckerman, and C. Kadie [1998]. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52.

Canny, J. [2002]. Collaborative filtering with privacy via factor analysis. *Proceedings ACM SIGIR'02*, pages 238–245, 2002.

Cohen, J. [1968]. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70:213–220.

Fouque, P.-A., G. Poupard, and J. Stern [2000]. Sharing decryption in the context of voting or loteries. *Proceedings of the 4th International Conference on Financial Cryptography, Lecture Notes in Computer Science*, 1962:90–104.

Fouque, P.-A., J. Stern, and J.-G. Wackers [2002]. Cryptocomputing with rationals. *Proceedings of the 6th International Conference on Financial Cryptography, Lecture Notes in Computer Science*, 2357:136–146.

Herlocker, J., J. Konstan, A. Borchers, and J. Riedl [1999]. An algorithmic framework for performing collaborative filtering. *Proceedings ACM SIGIR'99*, pages 230–237.

Karypis, G. [2001]. Evaluation of item-based top-n recommendation algorithms. *Proceedings 10th Conference on Information and Knowledge Management*, pages 247–254.

Nakamura, A., and N. Abe [1998]. Collaborative filtering using weighted majority prediction algorithms. *Proceedings 15th International Conference on Machine Learning*, pages 395–403.

Paillier, P. [1999]. Public-key cryptosystems based on composite degree residuosity classes. *Proceedings Advances in Cryptology − EUROCRYPT'99, Lecture Notes in Computer Science*, 1592:223–238.

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl [1994]. GroupLens: An open architecture for collaborative filtering of netnews. *Proceedings ACM CSCCW'94 Conference on Computer-Supported Cooperative Work*, pages 175–186.

Sarwar, B., G. Karypis, J. Konstan, and J. Riedl [2000]. Analysis of recommendation algorithms for e-commerce. *Proceedings 2nd ACM Conference on Electronic Commerce*, pages 158–167.

Sarwar, B., G. Karypis, J. Konstan, and J. Riedl [2001]. Item-based collaborative filtering recommendation algorithms. *Proceedings 10th World Wide Web Conference (WWW10)*, pages 285–295.

Shardanand, U., and P. Maes [1995]. Social information filtering: Algorithms for automating word of mouth. *Proceedings CHI'95*, pages 210–217.

van Duijnhoven, A.E.M. [2003]. Collaborative filtering with privacy. Master's thesis, Technische Universiteit Eindhoven.