

Using Singular Value Decomposition Approximation for Collaborative Filtering

Sheng Zhang, Weihong Wang, James Ford, Fillia Makedon
Department of Computer Science, Dartmouth College, Hanover, NH 03755
{clap, whwang, jford, makedon}@cs.dartmouth.edu

Justin Pearlman
Departments of Medicine & Radiology, Dartmouth Medical School, Lebanon, NH 03766
justin.pearlman@dartmouth.edu

Abstract

Singular Value Decomposition (SVD), together with the Expectation-Maximization (EM) procedure, can be used to find a low-dimension model that maximizes the log-likelihood of observed ratings in recommendation systems. However, the computational cost of this approach is a major concern, since each iteration of the EM algorithm requires a new SVD computation. We present a novel algorithm that incorporates SVD approximation into the EM procedure to reduce the overall computational cost while maintaining accurate predictions. Furthermore, we propose a new framework for collaborating filtering in distributed recommendation systems that allows users to maintain their own rating profiles for privacy. A server periodically collects aggregate information from those users that are online to provide predictions for all users. Both theoretical analysis and experimental results show that this framework is effective and achieves almost the same prediction performance as that of centralized systems.

Keywords: Collaborative Filtering, SVD Approximation, EM Procedure.

1. Introduction

Collaborative Filtering analyzes a user preferences database to predict additional products or services in which a user might be interested. The goal is to predict the preferences of a user based on the preferences of others with similar tastes. There are two general classes of collaborative filtering algorithms. Memory based algorithms [2, 11, 14] operate over the entire database to make predictions. Model based algorithms [1, 4, 6, 12, 13] use the database to estimate or learn a model for predictions.

Low-dimension linear models are a popular means to describe user preferences. The following are representative

state-of-the-art collaborative filtering algorithms: [1, 4, 6] directly assume that the user preferences database is generated from a linear model, matrix factorization based collaborative filtering methods [3, 9, 15, 16] obtain an explicit linear model to approximate the original user preferences matrix, and [2, 11] use the Pearson correlation coefficient, which is equivalent to a linear fit.

If we assume that users' ratings are generated from a low-dimension linear model together with Gaussian-distributed noise, the Singular Value Decomposition (SVD) technique can be used to find the linear model that maximizes the log-likelihood of the rating matrix, assuming it is complete. If the rating matrix is incomplete, as is the case in real-world systems, SVD cannot be applied directly. The Expectation-Maximization (EM) procedure [4, 16] can be used to find the model that maximizes the log-likelihood of the available ratings, but this requires a SVD computation of the whole matrix for each EM iteration. As the size of the rating matrix is usually huge (due to large numbers of users and items in typical recommendation systems), the computational cost of SVD becomes an important concern. Deterministic SVD methods for computing all singular vectors on an m -by- n matrix take $O(mn^2 + m^2n)$ time, and the Lanczos method requires roughly $O(kmn \log(mn))$ time to approximate the top k singular vectors [10].

In this work, we present a novel algorithm based on using an SVD approximation technique to reduce the computational cost of SVD based collaborative filtering. The basic idea is that in each iteration of EM procedure, a set of user rating profiles (rows) are sampled in a way such that the top k right singular vectors of the resulting sub-matrix approximate the top k right singular vectors of the original matrix. When this is done, SVD can be conducted on this sub-matrix instead of the whole matrix while still accurately updating all elements in the matrix.

We further extend this idea to a distributed collaborative filtering scenario, where users maintain their own rating

profiles for privacy and a server periodically collects aggregate information from online users in order to provide rating predictions on demand. We show that with a high probability, the rating profiles of a random small set of users are capable of making almost the same predictions as the rating profiles of all users.

The organization of this paper is as follows. Section 2 describes the linear model used for rating, explains why SVD is useful, and explains how the EM procedure can be used in practical systems. Section 3 presents our algorithm, which incorporates the SVD approximation into the EM procedure, and gives a theoretical analysis of it. Section 4 proposes our framework for collaborative filtering in distributed recommendation systems and presents algorithms for computing aggregates and making predictions. We also discuss algorithm stability and survey the security schemes that can be used for preserving privacy. Section 5 presents experimental results obtained from real data sets. Finally, Section 6 concludes the paper.

2. Background

The fundamental problem addressed by linear models is how to efficiently represent a large matrix of user ratings. Denote the rating matrix as A (m users-by- n items); then $A_{(i)}$ (the i th row of A) is user i 's rating profile and A_{ij} represents the rating given by user i on item j . Define matrix X (m -by- n and with rank at most k) as a low-dimension linear model that approximates the rating matrix A . Hence, each element A_{ij} is equal to X_{ij} plus an error that is from a Gaussian distribution with zero mean and standard deviation σ_{ij} . For simplicity, we assume that all σ_{ij} are equal to σ . Therefore, $A_{ij} \sim \mathcal{N}(X_{ij}, \sigma)$, and the log-likelihood of A_{ij} given X_{ij} follows as

$$\log \Pr(A_{ij}|X_{ij}) = -\frac{1}{2\sigma^2}(A_{ij} - X_{ij})^2 + C.$$

In this and following equations, C is a constant. If we assume that $\Pr(A_{ij}|X_{ij})$ are independent, the log-likelihood of the whole rating matrix A given the linear model X is

$$\begin{aligned} \log \Pr(A|X) &= \log \prod_{ij} \Pr(A_{ij}|X_{ij}) \\ &= -\frac{1}{2\sigma^2} \sum_{ij} (A_{ij} - X_{ij})^2 + C. \end{aligned}$$

Thus, if A is completely known, finding the X that maximizes $\log \Pr(A|X)$ is equivalent to finding the X that minimizes $\sum_{ij} (A_{ij} - X_{ij})^2$, which is a least square problem and can be solved by SVD. If A is factorized in the form $A = USV^T$ via SVD, then the product of the top k left singular vectors U_k , the diagonal matrix of the top k singular values S_k , and the transpose of the top k right singular vectors V_k^T satisfies $\|A - U_k S_k V_k^T\|_F \leq \|A - M\|_F$

for any matrix M with rank less than or equal to k , where $\|\cdot\|_F$ denotes the Frobenius Norm. So given a completely known rating matrix A , $X = U_k S_k V_k^T$ can be considered as the most probable or the most accurate k -dimensional linear model that describes A .

In real recommendation systems, the rating matrix A is incomplete, which means that some ratings are available while others are unknown (unrated). To deal with an incomplete rating matrix, one can impute missing elements using some simple methods such as using user averages or item averages, and then find the linear model that fits the filled-in rating matrix best. However, imputed values obtained by these naive methods are not likely to be true, and thus the linear model found may deviate significantly from the true model. A better way to deal with missing ratings, described in [4, 16], is to find the model that maximizes the log-likelihood of all observed ratings (denoted as A^o) using an EM procedure.

The **Expectation** step of the t th iteration in the EM procedure updates the expected expression of the complete-data log-likelihood with respect to the unknown data (denoted as A^u) given the observed data A^o and the current parameter estimates $X^{(t-1)}$, which can be written as $\mathbb{E}[\log \Pr(A^o, A^u|X)|A^o, X^{(t-1)}]$. If A_{ij} is observed,

$$\mathbb{E}[\log \Pr(A_{ij}|X)|A^o, X^{(t-1)}] = -\frac{1}{2\sigma^2}(A_{ij} - X_{ij})^2 + C.$$

If A_{ij} is unknown, we have $\mathbb{E}[A_{ij}|X_{ij}^{(t-1)}] = X_{ij}^{(t-1)}$. It is because $A_{ij} \sim \mathcal{N}(X_{ij}, \sigma)$, and the expected expression of A_{ij} given the current parameter estimate $X_{ij}^{(t-1)}$ is thus equal to that estimate. Therefore,

$$\begin{aligned} &\mathbb{E}[\log \Pr(A_{ij}|X)|A^o, X^{(t-1)}] \\ &= \mathbb{E}[\log \Pr(A_{ij}|X_{ij})|X_{ij}^{(t-1)}] \\ &= -\frac{1}{2\sigma^2} \mathbb{E}[(A_{ij} - X_{ij})^2|X_{ij}^{(t-1)}] + C \\ &= -\frac{1}{2\sigma^2} (\mathbb{E}[A_{ij}^2|X_{ij}^{(t-1)}] - 2\mathbb{E}[A_{ij}X_{ij}|X_{ij}^{(t-1)}] + X_{ij}^2) + C \\ &= -\frac{1}{2\sigma^2} ((X_{ij}^{(t-1)})^2 + \sigma^2 - 2X_{ij}^{(t-1)}X_{ij} + X_{ij}^2) + C \\ &= -\frac{1}{2\sigma^2} (X_{ij}^{(t-1)} - X_{ij})^2 + C. \end{aligned}$$

By combining these two cases,

$$\begin{aligned} &\mathbb{E}[\log \Pr(A^o, A^u|X)|A^o, X^{(t-1)}] \\ &= \mathbb{E}[\log(\Pr(A^o|X) \Pr(A^u|X))|A^o, X^{(t-1)}] \\ &= -\frac{1}{2\sigma^2} \left(\sum_{A_{ij} \in A^o} (A_{ij} - X_{ij})^2 \right. \\ &\quad \left. + \sum_{A_{ij} \in A^u} (X_{ij}^{(t-1)} - X_{ij})^2 \right) + C. \end{aligned} \quad (1)$$

The **Maximization** step of the t th iteration computes the updated model $X^{(t)}$ that maximizes the expected expression obtained in the Expectation step. By equation (1), we know that $X^{(t)}$ should be the matrix that minimizes

$$\sum_{A_{ij} \in A^o} (A_{ij} - X_{ij})^2 + \sum_{A_{ij} \in A^u} (X_{ij}^{(t-1)} - X_{ij})^2.$$

Denote as $A^{(t)}$ a filled-in matrix whose unknown entries have been filled in from $X^{(t-1)}$, *i.e.*,

$$A_{ij}^{(t)} = \begin{cases} A_{ij} & \text{if } A_{ij} \text{ is observed} \\ X_{ij}^{(t-1)} & \text{if } A_{ij} \text{ is unknown} \end{cases}$$

. Thus, the objective becomes finding the model $X^{(t)}$ that minimizes $\sum_{ij} (A_{ij}^{(t)} - X_{ij})^2$. As shown above, this problem can be solved by performing SVD on $A^{(t)}$ to calculate the best rank k approximation to it.

The EM procedure is ensured to converge [8], which means that the log-likelihood of all observed ratings given the current model estimate is always nondecreasing. Because A_{ij} is from a Gaussian distribution with X_{ij} as its mean, given X_{ij} $\Pr(A_{ij}|X_{ij})$ obtains its maximum when A_{ij} is equal to X_{ij} . Thus, X_{ij} is the best prediction for user i 's rating on item j .

3. SVD Approximation in Centralized Recommendation Systems

In this section, we discuss how to use SVD approximation to reduce the computational cost of the SVD based collaborative filtering in traditional centralized recommendation systems where the server keeps all users' rating profiles. If the server uses the EM procedure shown in Section 2, the computational cost will be $l \cdot O(mn^2 + m^2n)$, in which l is the number of iterations of the EM procedure and $O(mn^2 + m^2n)$ is the computational cost of performing deterministic SVD on an m (users)-by- n (items) matrix. This cost is expensive because m and n can be very large in a recommendation system, from thousands to millions.

The SVD Approximation technique of Drineas *et al.* [5] shows that for any matrix A (m -by- n), if c rows are sampled and scaled appropriately, the top right singular vectors of the new matrix C (c -by- n) approximate the top right singular vectors of A . More formally, assume that in picking a certain row for C , the probability that the i th row in A is picked (denoted as p_i) is such that $p_i \geq \beta \|A_{(i)}\|^2 / \|A\|_F^2$, where β is a constant and $\|\cdot\|$, denoting a vector's length, is equal to the squared root of the sum of the squares of all its elements, and suppose that if $A_{(i)}$ is picked, $A_{(i)}/\sqrt{cp_i}$ will be included as a row in C . Denote H (n -by- k) as the matrix formed by the top k right singular vectors of C and

A_k ($A_k = U_k S_k V_k^T$) as the best rank k approximation to A . It follows that for any $c \leq n$ and $\delta > 0$,

$$\|A - AHH^T\|_F^2 \leq \|A - A_k\|_F^2 + 2(1 + \sqrt{8 \ln(2/\delta)}) \sqrt{\frac{k}{\beta c}} \|A\|_F^2. \quad (2)$$

In addition, if $p_i = \|A_{(i)}\|^2 / \|A\|_F^2$, which implies that β is equal to one,

$$\|A - AHH^T\|_F^2 \leq \|A - A_k\|_F^2 + 2(1 + \sqrt{8 \ln(2/\delta)}) \sqrt{\frac{k}{c}} \|A\|_F^2. \quad (3)$$

The proof of both inequalities can be found in [5].

As discussed in Section 2, the objective of the maximization step in the t th iteration of the EM procedure is to compute the best rank k approximation to the filled-in matrix $A^{(t)}$. Using SVD approximation, the server can sample c users' rating profiles (rows in A) with probability proportional to their length squared and form the matrix C after scaling. Note here that the c samples are not necessarily from c different users, and that a user's rating profile might be sampled more than once according to the sampling method. After computing the top k right singular vectors of C and obtaining the matrix H , the server can use $A^{(t)}HH^T$ to approximate $A_k^{(t)}$. Then in the expectation step of the $(t+1)$ th iteration, the unknown entry A_{ij} is calculated as $(A^{(t)}HH^T)_{ij}$.

By inequality (3), the server has a high confidence that given the same filled-in matrix $A^{(t)}$, the rank k model $A^{(t)}HH^T$ obtained *via* SVD approximation is close to the best rank k approximation $A_k^{(t)}$. Therefore, with a high probability, the EM procedure is likely to calculate more and more likely true values for missing entries. This implies that although the EM procedure with SVD approximation is not guaranteed to converge on an optimal solution, the log-likelihood of observed ratings will generally increase.

Algorithm 1 EM Procedure *via* SVD approximation

- 1: Set initial values $X_{ij}^{(0)}$ for unknown entries A_{ij} .
 - 2: **while** in the t th iteration of EM procedure **do**
 - 3: Fill in A by replacing unknown entries A_{ij} with $X_{ij}^{(t-1)}$, denote the filled-in matrix as $A^{(t)}$.
 - 4: Set $p_i = \|A_{(i)}^{(t)}\|^2 / \|A^{(t)}\|_F^2$, and pick c rows from A .
 If the i th row is picked, include $A_{(i)}^{(t)} / \sqrt{cp_i}$ in C .
 - 5: Compute the top k right singular vectors of C and form these k vectors into the matrix H .
 - 6: $X^{(t)} = A^{(t)}HH^T$.
 - 7: **end while**
-

Algorithm 1 shows the details of applying SVD approximation in centralized recommendation systems. Note that

to compute the top k right singular vectors of C (line 5) directly by performing SVD on C takes $O(c^2n + cn^2)$ time. A more efficient approach is to compute CC^T first ($O(c^2n)$ time), and then perform SVD on CC^T ($O(c^3)$ time). From basic linear algebra, we know that the left singular vectors of C^TC are the same as the left singular vectors of C , and that the singular values of C^TC are the squares of the singular values of C . Thus, the top k right singular vectors of C can be computed *via* its top k left singular vectors and top k singular values ($O(cnk)$ time). As generally k is much smaller than c and c is much smaller than n , the total computational cost of performing SVD on C in the above way is $O(c^2n)$.

4. SVD Approximation in Distributed Recommendation Systems

Traditional centralized recommendation systems have problems such as users losing their privacy, retail monopolies being favored, and diffusion of innovations being hampered [3]. Distributed collaborative filtering systems, where users keep their rating profiles to themselves, have the potential to correct these problems. However, in the distributed scenario there are two new problems that need to be dealt with. The first problem is how to ensure that users' data are not revealed to the server and other users. The second problem is how to ensure that users can get as accurate predictions as they do in the centralized scenario. This paper is mainly focused on the second problem, and consequently we rely mechanisms shown in [3, 7] to address the first problem.

Since the server cannot directly see users' rating profiles, it needs to compute an *aggregate* (a learning result based on user information) for making predictions. Figure 1 shows our framework for collaborative filtering in distributed recommendation systems. At a certain time point t , the server securely computes the aggregate (denoted as G_t) from those users who are online at that time point (denoted as U_t); "securely" here means that users' rating profiles are not disclosed to the server and other users. Between time point t and $t + 1$, when a certain user (no matter whether she is in U_t or not) needs predictions, the server computes predictions based on this user's rating profile and the aggregate G_t .

The reason of computing aggregates periodically is that users' rating profiles are dynamic. For any given user, the probability that he is in U_t is independent of the probability that he is in U_{t+1} , so U_t and U_{t+1} would be expected to have few users in common (given sufficiently many users). Therefore, it is hard to find a way to combine aggregates computed at different time points for predictions. A more minor concern in this framework is how the server picks time points for aggregate computations. Of course, time

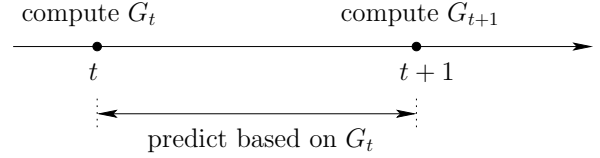


Figure 1. The framework for collaborative filtering in distributed recommendation systems.

points at which more users are online seem preferable, since having more users generally lead to more accurate learning results. On the other hand, according to the following theoretical analysis and experimental results, a small subset (about 5%) of all users is often enough for good predictions.

4.1 Algorithms and Theoretical Analysis

We first present algorithms for computing aggregates and generating predictions and then a theoretical analysis of their performance. Assume that there are c online users at time point t , and that their rating profiles are denoted $A_{(1)}$ to $A_{(c)}$. Algorithm 2 shows how to generate the aggregate.

Algorithm 2 Computing the aggregate G_t

- 1: **for** each user i , to each unknown entry A_{ij} **do**
 - 2: If A_{ij} has been predicted before, replace A_{ij} with the latest prediction.
 - 3: Else replace A_{ij} with the average of user i 's ratings.
 - 4: **end for**
 - 5: The server securely performs SVD on the matrix C (c -by- n) formed by filled-in rating profiles.
 - 6: Aggregate G_t is the matrix (n -by- c) formed by the top k right singular vectors of C .
-

When a user i asks for predictions, the server generates predictions as follows using the aggregate G_t .

Algorithm 3 Generating predictions for user i

- 1: For each unknown entry A_{ij} , if A_{ij} has been predicted before, replace A_{ij} with the latest prediction.
 - 2: Else replace A_{ij} with the average of user i 's ratings.
 - 3: Multiply the filled-in rating profile vector (1-by- n) by $G_t G_t^T$ to generate predictions.
-

For analysis, we make the following two assumptions.

Assumption 1: there exists a constant β such that for any m and for any user i , the filled-in rating profile vector (denoted as $A_{(i)}^*$) satisfies $\sum_{j=1}^m \|A_{(j)}^*\|^2 / (m \cdot \|A_{(i)}^*\|^2) \geq \beta$. Recall that m is the total number of users. The soundness of this assumption is shown in Appendix A.

Assumption 2: there is a uniform, constant probability that any user is online at any time point.

Assuming that c users are online (sampled) at a time point according to Assumption 2, then such a sampling method is similar to another sampling method that picks also c samples in the following way: to choose a sample, the probability that any user is picked is uniform (*i.e.*, $1/m$). Note that picking c users at random is not the same as sequentially picking c users with a uniform probability. The difference is that each user will be picked at most once in the first case, while a user might appear more than once in the second. Taking into account the fact that the probability of repetitions is small when c is considerably smaller than n , for simplicity these two sampling methods can be regarded as equivalent in practice.

This observation, together with Assumption 1, leads to the following conclusion. At time point t , denote A^* as the filled-in rating matrix after each user imputes unknown entries and A_k^* as the best rank k approximation to A^* . Assume that there are c online users and that if user i is online, $A_{(i)}^*/\sqrt{c/m}$ is a row in C . Then by inequality (2), the G_t formed from the top k right singular vectors of C satisfies the following inequality: for any $\delta > 0$.

$$\|A^* - A^*G_tG_t^T\|_F^2 \leq \|A^* - A_k^*\|_F^2 + 2(1 + \sqrt{8\ln(2/\delta)})\sqrt{\frac{k}{\beta c}}\|A^*\|_F^2. \quad (4)$$

Since each row in C has the same scalar $\sqrt{c/m}$, it can be omitted because it will not affect the resulting singular vectors. Therefore, online users' filled-in rating profiles can be directly combined to form C using Algorithm 2.

The above inequality says that with a high probability the computed aggregate implies a highly accurate linear model behind the current filled-in rating matrix. For an unknown entry A_{ij} , if the server makes predictions for user i between time point t and $t + 1$, then its corresponding value A_{ij}^* at time point $t + 1$ will be imputed based on the model estimate obtained at time point t . Therefore, the aggregate computation can be taken as the model re-estimation process in the EM procedure maximization step, and prediction generation can be taken as the calculation of expected expressions in the expectation step. Some online users involved in the aggregate computation may not ask for predictions since the last aggregate computation, which implies that their unknown rating entries have not been imputed based on the updated model estimate. Nonetheless, we can require them to compute predictions first based on G_t before they join the computation of G_{t+1} . Therefore, the server may sometimes require more times of aggregate computations to obtain the accurate model, but the final prediction accuracy will not be affected a lot.

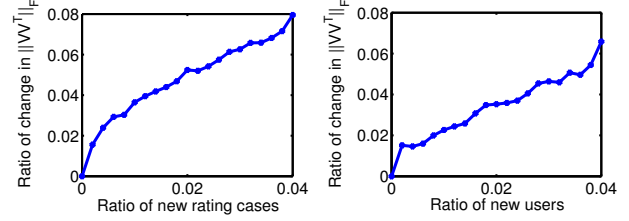


Figure 2. The impact of new ratings or new users on the ratio of change in $\|V_k V_k^T\|_F$, that is $\|V_k V_k^T - \hat{V}_k \hat{V}_k^T\|_F / \|V_k V_k^T\|_F$, here $k = 10$.

4.2 Stability of Predictions

A potential problem in a rating system is that users, items, and ratings are all dynamic and subject to change at any time. As a result of this, the rating matrix may change between two aggregate computations as a result of the following three situations: 1) users giving new (updated) ratings on items; 2) new users being registered in the system; and 3) new items being added to the system. An altered rating matrix typically requires a different linear model to best describe it. However, the linear model should not be disturbed to a significant extent when the rating matrix undergoes a small change—otherwise predictions made between two aggregate computation will probably be inaccurate. Since the computed aggregate in Algorithm 2 approximates the top k right singular vectors (V_k) of the current filled-in rating matrix, the core concern is an instability in $V_k V_k^T$.

We conducted two preliminary experiments on a 5000-by-1427 rating matrix from the EachMovie data set to assess the impact of rating matrix changes on $V_k V_k^T$. In the first experiment, 90% of the entries are randomly picked to form the original rating matrix A , and the rest of the entries are progressively added to form \hat{A} . In the second experiment, rating profiles from 90% of the users are used to form the initial matrix A and the remaining users are progressively added to create \hat{A} . The case where new items are added is not considered here because the size of $V_k V_k^T$ will change as the number of items is changed, and this will make it difficult to assess its effect. Figure 2 shows that the difference between $V_k V_k^T$ and $\hat{V}_k \hat{V}_k^T$ is in a small range (6%) when the number of rating cases or the number of users is increased within a small range (3%). In a real system, it is reasonable that changes in the number of rating cases and the number of users will be in this range between two consecutive aggregate computations.

Another issue related to the stability of results is how frequently the aggregate should be computed. Note that $G_t G_t^T$ (the product of aggregate and its transpose) has a fixed size

(n -by- n) no matter how many online users join the aggregate computations. The frequency of aggregate computations can thus be decided based on the disturbance of $G_t G_t^T$. When the change is very small, the interval between aggregate computations can be made longer, and vice versa.

4.3 Preserving Privacy

This work is not focused on improvements in preserving privacy in the distributed scenario, so to address the privacy issue in Algorithm 2 and 3, we apply security schemes proposed in [3, 7]. In Algorithm 2, a distributed secure SVD computation is needed to ensure that users' rating profiles are not revealed to other users or the server. Canny's paper [3] proposed a scheme to achieve this objective. The idea is to reduce the SVD computation to an iterative calculation requiring only the addition of vectors of user data, and use homomorphic encryption to allow the sums of encrypted vectors to be computed and decrypted without exposing individual data.

In Algorithm 3, the multiplication of a user's rating profile by $G_t G_t^T$ should be securely computed both so that the server cannot learn the rating profile and so that the user cannot learn $G_t G_t^T$. Moreover, the multiplication result should not be revealed to the server either, as in that case the server could easily compute the user's rating data. The multiplications of a vector by a matrix can be considered as a group of scalar products. There are several privacy preserving schemes for calculating scalar products in the security literature and one of them is presented in [7]. It also has an asymmetric version to let only one side know the final result.

5 Results

The objectives of our experiments were to test the performance of SVD approximation in two situations: in centralized recommendation systems (Algorithm 1) and in distributed recommendation systems (Algorithm 2 and 3).

Two data sets, known as *Jester* and *EachMovie*, were used in experiments. *Jester* is a web based joke recommendation system developed at the University of California, Berkeley [9]. The data set contains 100 jokes, with user ratings ranging from -10 to 10 . The *EachMovie* set is from a research project at the Compag Systems Research Center. User ratings for movies are recorded on a numeric six-point scale (0, 0.2, 0.4, 0.6, 0.8, 1.0). The density (or fraction of the rating matrix that is filled) of the *Jester* set is about 50%, while that of *EachMovie* is only 3%.

For each experiment, we pick a 5000-by-100 (users-by-items) rating matrix from the *Jester* data set and a 5000-by-1427 rating matrix from *EachMovie*. For each rating matrix, 90% of the observed ratings are randomly picked as

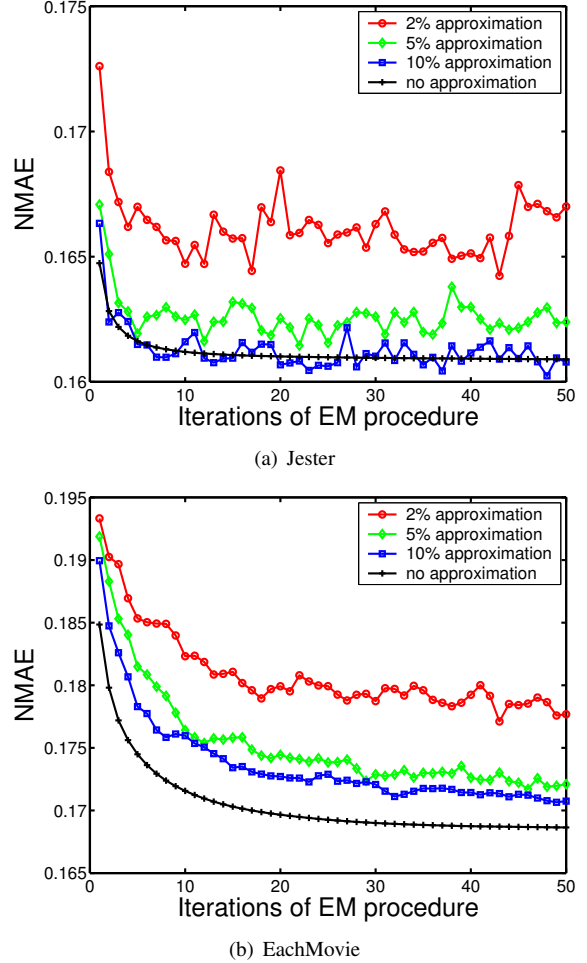


Figure 3. Comparison of Algorithm 1 (with three approximation ratios of 2%, 5%, and 10%) with the standard EM algorithm ("no approximation").

available and the other observed ratings are regarded as test cases. Algorithms are performed on the available cases to make predictions for the test cases. For all algorithms, the reduced linear dimension (k) is equal to 4. This value was selected empirically based on both prediction results and convergence rate. It is consistent with the results in [16].

The experimental measure used for comparisons is *Normalized Mean Absolute Error (NMAE)*, which is the average of the absolute values of the difference between the real ratings and the predicted ratings divided by the ratings range.

5.1 Experiment 1

Experiment 1 examines the performance of applying SVD approximation in the centralized recommendation sys-

tem scenario. The selected approximation ratios (the number of rows (c) in Algorithm 1 over the number of users) are 2%, 5%, and 10%. Figure 3 displays NMAE results of Algorithm 1 with these three approximation ratios and the standard EM procedure without SVD approximation. As predicted analytically in Section 3, the log-likelihood of observed ratings in EM procedure with SVD approximation is generally increasing, although it is not monotonically increasing. Thus, Figure 3 is a verification that the prediction accuracy of Algorithm 1 generally increases when more iteration are used in the EM procedure. In both data sets, the NMAE of Algorithm 1 with an approximation ratio of 5% is less than 2% higher than the NMAE for the standard algorithm. Moreover, the convergence rate of both algorithms is nearly identical. Algorithm 1 takes only about one tenth the time for each iteration compared with the standard algorithm. All these points support the conclusion that our algorithm is practical for real-world systems.

5.2 Experiment 2

In experiment 2, the performance of Algorithm 2 and 3 when applied in the distributed recommendation system scenario is compared with the performance of the standard EM procedure in the centralized scenario. For algorithms in the distributed scenario, a random set (2%, 5%, and 10%) of the user rating profiles are picked for each time-step’s aggregate computation. We assume that 50% of the users ask for predictions during two aggregate computations, so a random 50% of the rows are picked, in which unknown entries are imputed based on the current aggregate. In order to compare the centralized EM algorithm with the other cases, each iteration in it is regarded as one aggregate computation, and all missing entries are imputed based on the current model estimate after each iteration. Figure 4 gives the NMAE results obtained in these two scenarios. It shows that our algorithms for distributed recommendation systems are comparable to the EM procedure algorithm in centralized recommendation systems under these conditions. With only 5% of the users online in each time-step’s aggregate computation, the distributed scenario achieves a prediction accuracy that is at most 2% worse than that of the centralized scenario.

6 Conclusion

This paper presents novel collaborative filtering algorithms that incorporate SVD approximation technique into an SVD based EM procedure. In centralized recommendation systems, the standard SVD based EM procedure algorithm takes $O(m^2n + mn^2)$ time for each SVD computation, while our new algorithm with SVD approximation takes only $O(s^2n)$ time. Experiments on existing data sets

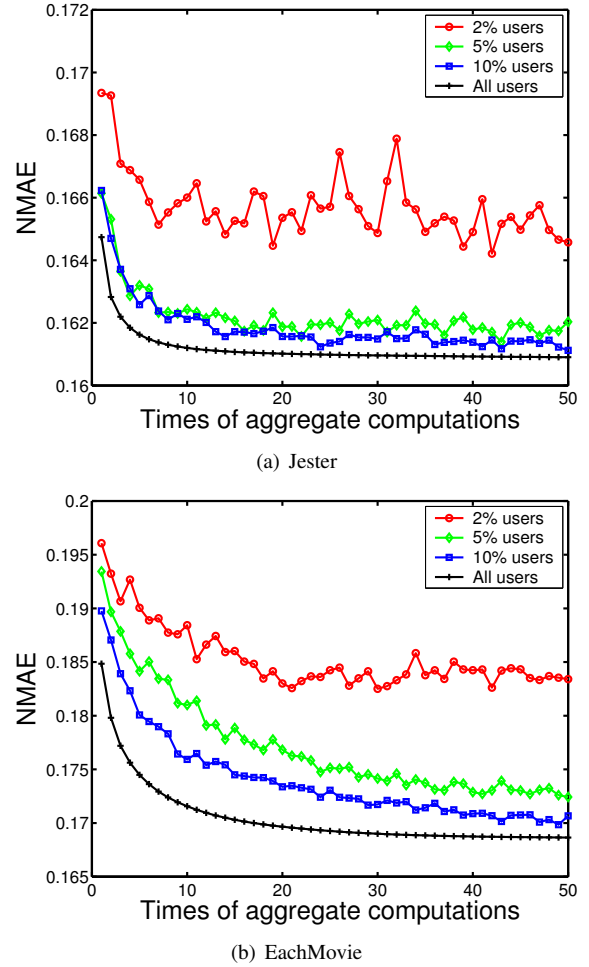


Figure 4. Comparison of algorithms in distributed recommendation systems (when the number of online users in each aggregate computation is 2%, 5%, or 10% of the total number of users) with the standard EM algorithm as applied in centralized recommendation systems (“All users”).

show that our algorithm is very promising. Its prediction accuracy is almost the same as that of the standard algorithm even if a small approximation ratio is used. We also propose a new framework for collaborative filtering *via* SVD approximation in distributed recommendation systems in which the server periodically calculates an aggregate from online users’ rating profiles and makes predictions for all users based on it. Our experiments show that if the number of online users is at least five percent of the total number of users, the prediction accuracy in this distributed scenario is almost the same as what is obtained in the centralized

scenario.

In the proposed distributed framework, there is still a central server that computes aggregates and provides predictions. This framework is suitable for online collaborative filtering service providers if users are concerned about the privacy of their rating profiles. However, a recommendation system based on a central server is less suited for peer-to-peer environments. Therefore, one possible project of future work is to extend the current server-based distributed framework to a peer-to-peer environment, where we need to consider additional issues such as the availability of peers that keep aggregates, the validity of aggregates, and limitations on peer resources.

Acknowledgements

This work was supported by the National Science Foundation under grant IDM 0308229, and also in part by FAMRI through a Center of Excellence.

References

- [1] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, 2001.
- [2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.
- [3] J. Canny. Collaborative filtering with privacy. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
- [4] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th ACM SIGIR Conference*, 2002.
- [5] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering of large graphs via the singular value decomposition. *IEEE Journal of Machine Learning*, 56(1-3):9–33, 2004.
- [6] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of the 34th ACM symposium on Theory of computing*, 2002.
- [7] W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, 2001.
- [8] Z. Ghahramani and M. I. Jordan. Learning from incomplete data. Technical report, MIT, 1994.
- [9] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. EigenTaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- [10] G. Golub and C. V. Loan. *Matrix Computations (3rd edition)*. Johns Hopkins University Press, 1996.
- [11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM SIGIR Conference*, 1999.
- [12] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.
- [13] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.
- [14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 1994.
- [15] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems—a case study. In *ACM WebKDD Web Mining for E-Commerce Workshop*, 2000.
- [16] N. Srebro and T. Jaakkola. Weighted low rank approximation. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

Appendix

A. Soundness of Assumption 1

To verify the soundness of Assumption 1, an experiment was performed on a 5000-by-1427 rating matrix from EachMovie. Missing entries are filled in using the average of that user for all available entries. Let $\beta^* = \min_i \sum_{j=1}^m \|A_{(j)}^*\|^2 / (m \cdot \|A_{(i)}^*\|^2)$. Table 1 displays the mean value and the standard deviation of β^* (from 20 trials) when m increases from 1000 to 5000. It shows that β^* is very stable when m increases.

Table 1. The mean value (“mean”) and the standard deviation (“std”) of β^* from 20 trials when m increases.

m	1000	2000	3000	4000	5000
mean	0.406	0.407	0.407	0.407	0.407
std	0.006	0.003	0.002	0.001	0.000