A Scalable Collaborative Filtering Framework based on Co-clustering

Thomas George
Department of Computer Science
Texas A & M University
tgeorge@cs.tamu.edu

Srujana Merugu
Department of Electrical and Computer Engg.
University of Texas at Austin
merugu@ece.utexas.edu

Abstract

Collaborative filtering-based recommender systems, which automatically predict preferred products of a user using known preferences of other users, have become extremely popular in recent years due to the increase in web-based activities such as e-commerce and online content distribution. Current collaborative filtering techniques such as correlation and SVD based methods provide good accuracy, but are computationally very expensive and can only be deployed in static off-line settings where the known preference information does not change with time. However, a number of practical scenarios require dynamic real-time collaborative filtering that can allow new users, items and ratings to enter the system at a rapid rate. In this paper, we consider a novel collaborative filtering approach based on a recently proposed weighted co-clustering algorithm [3] that involves simultaneous clustering of users and items. We design incremental and parallel versions of the co-clustering algorithm and use it to build an efficient real-time collaborative filtering framework. Empirical evaluation of our approach on large movie and book rating datasets demonstrates that it is possible to obtain an accuracy comparable to that of the correlation and matrix factorization based approaches at a much lower computational cost.

1 Introduction

The evolution of the internet during the last decade has resulted in an overwhelming increase in web-based activities such as e-commerce and online content distribution where users are often forced to choose from a large number of products or content items. To aid users in the decision making process, it has become increasingly important to design recommender systems that automatically identify the likely choices of the users. Collaborative filtering techniques [9] that identify the likely preferences of a user based on the known preferences of other users have been shown to be very effective for generating high quality recommendations. These techniques not only provide superior performance compared to content-based filtering methods [6], but also do not require the users to explicitly state their interests.

Most of the existing collaborative filtering methods based on correlation criteria, singular value decomposition (SVD) and non-negative matrix factorization (NNMF) have been shown to provide highly accurate predictions of ratings. However, these techniques suffer from one main drawback, which is the presence of a computationally expensive training component. As a result, these techniques can only be deployed in static settings where the known preferences do not vary much with time. However, a number of practical scenarios such as real-time news personalization require dynamic collaborative filtering that can handle new users, items and ratings entering the system at a rapid rate. In such situations, it is imperative for the recommender system to dynamically adapt its predictions using the new information, which in turn requires a fast and efficient training algorithm.

In this paper, we consider a novel collaborative filtering approach based on a recently proposed weighted co-clustering algorithm [3]. The key idea is to simultaneously obtain user and item neighborhoods via co-clustering and generate predictions based on the average ratings of the co-clusters (user-item neighborhoods) while taking into account the individual biases of the users and items. Based on this approach, we design an efficient real-time collaborative filtering framework and make two new contributions.

- First, we propose a dynamic collaborative filtering approach that can support the entry of new users, items and ratings using a hybrid of incremental and batch versions of the co-clustering algorithm. Empirical comparison of our approach with SVD, NNMF and correlation-based collaborative filtering techniques indicates comparable accuracy at a much lower computational effort.

- Second, we design a scalable, real-time collaborative filtering system by developing parallel versions of the co-clustering, prediction and incremental training routines.

The rest of the report is organized as follows: Section 2 describes related work. Section 3 contains a formal definition of the recommendation problem. Section 4 explains the proposed collaborative filtering approach while

Section 5 describes a scalable framework based on parallel co-clustering and prediction algorithms. Section 6 provides the results of our empirical evaluation and Section 7 contains the conclusions and future work.

Notation: Upper case alphabets, e.g., A are used to represent matrices with the subscripted versions, e.g., A_{ij} denoting the corresponding matrix elements. Sets are represented by calligraphic upper-case alphabets, e.g., $\mathcal{X}, \mathcal{Y}$. and enumerated as $\{\mathbf{x}_i\}_{i=1}^{n}$ where $\mathbf{x}_i$ are the elements of the set.

2 Related Work

In this section, we briefly discuss some of the existing literature on recommender systems, collaborative filtering (CF) and co-clustering.

Automated recommender systems can be broadly classified into two groups – (i) content-based filtering systems [6, 4] that predict preferences based on the content of the items and the explicit interests of the users, and (ii) collaborative filtering systems [15] that make predictions based solely on the known ratings of similar users. The relative performance of these techniques depends on the sparsity of the known ratings. However, recent advances in internet and database technology have made it feasible to collect large number of user-product ratings with little or no effort on part of users, thus making collaborative filtering a superior and more practical approach [15].

A number of collaborative filtering techniques have been proposed in the literature, of which the most popular ones are those based on correlation criteria [12, 6] and matrix factorization [13, 17]. The correlation-based techniques use similarity measures such as Pearson correlation [12] and cosine similarity [16] to determine a neighborhood of like-minded users for each user and then predict the user's rating for a product as a weighted average of ratings of the neighbors. Though reasonably accurate, correlation-based techniques have much reduced coverage since they cannot detect item synonymy. They are also computationally very expensive as the correlation between every pair of users needs to be computed during the training phase.

The matrix factorization approaches include SVD [13] and NNMF-based [17, 10] filtering techniques that predict the unknown ratings based on a low rank approximation of the original ratings matrix. The missing values in the original matrix are filled using average values of the rows or columns. Unlike correlation-based methods, the matrix factorization techniques treat the users and items symmetrically and hence, handle item synonymy and sparsity in a better fashion. However, the training component of these techniques is computationally intensive making it impractical to have frequent re-training. Incremental versions of SVD based on folding-in and exact rank-1 updates [14, 5] partially alleviate this problem. However, the update operations are also not very efficient since the effects of small updates to the ratings matrix are not localized. In addition to the two classes of techniques described above, there has also been work on CF approaches based on user-clustering [18] and horting [2] – a graph based approach for identifying the neighbors of a user.

The CF approach we consider in this paper is based on a special case of the weighted Bregman co-clustering algorithm presented in [3]. There has been a lot of work on co-clustering [8, 7], most of which was focused on handling sparsity and dimensionality reduction issues in text and micro-array analysis. A crucial difference between [3] and earlier co-clustering work is the formulation of the co-clustering problem as a matrix approximation problem with non-uniform weights on the input matrix elements. Since the prediction of unknown ratings (weight=0) can be posed as a weighted matrix approximation problem, it is possible to now employ a CF approach based on co-clustering.

The co-clustering approach is similar to the correlation and clustering-based techniques in the sense that neighborhoods are employed for prediction, the main difference being that both users and items are clustered so that item synonymy ceases to be a problem. Further, as in the case of SVD and NNMF, the co-clustering algorithm also optimizes the approximation error of a low parameter reconstruction of the ratings matrix. However, unlike SVD and NNMF, the effects of changes in the ratings matrix are localized which makes it possible to have efficient incremental updates.

3 Problem Definition

We now formulate the recommendation problem in terms of predicting the unknown ratings. Using a matrix representation, we transform this problem to a weighted matrix approximation problem and motivate the co-clustering based approach for solving it.

Let $\mathcal{U} = \{u_i\}_{i=1}^{m}$ be the set of users such that $|\mathcal{U}| = m$ and $\mathcal{P} = \{p_j\}_{j=1}^{n}$ be the set of items such that $|\mathcal{P}| = n$. Let A be the $m \times n$ ratings matrix such that A_{ij} is the rating of the user u_i for the item p_j and let W be the $m \times n$ matrix corresponding to the confidence of the ratings in A. In the absence of explicit confidence information, we assume $W_{ij} = 1$ when the rating is known and 0 otherwise.

In a static setting, the set of users, the set of items and the known ratings are fixed and the main problem in recommender systems involves guessing an unknown rating for an existing user and item pair, i.e., one of the missing elements in A. The predicted ratings can then be used to generate the top-N recommendations if required. A natural approach to this rating prediction problem is to assume that the matrix has a certain low parameter structure, deduce the parameters of the structure based on the available ratings so that a certain loss function is minimized, and then use a matrix reconstruction based on this structure for predicting the missing values. To make this approach concrete, we need to specify the class of allowed structures as well as the loss function used to measure the approximation error. Usually, the loss function is chosen to be the squared error and the class of chosen structures determines the technique being adopted. For example, when the imposed structure is a low rank approximation with no constraints, the best approximation is obtained using SVD (assuming the missing values are suitably handled), thus leading to the SVD-based approach [13]. Similarly, when the imposed structure is a low rank k approximation that can be factorized into non-negative matrices of sizes $m \times k$ and $k \times n$, the best approximation is obtained using the NNMF algorithm [11]. Though powerful in terms of the approximation ability, these two approaches suffer from the drawback that small changes in the original matrix A could lead to changes in most of the parameters in the approximation (i.e. changes are not localized), making it difficult to design incremental training algorithms.

To address these concerns, we consider low parameter approximations based on simultaneous clustering (co-clustering) of users and items in the ratings matrix A. Let $\rho : \{1, \cdots, m\} \mapsto \{1, \cdots k\}$ and $\gamma : \{1, \cdots, n\} \mapsto \{1, \cdots, l\}$ denote the user and item clustering where k and l are number of user and item clusters. The simplest approximation scheme based on co-clustering is the one where each missing rating is approximated by the average value in the corresponding co-cluster. In this case, the approximate matrix has only kl parameters and might not be able to provide accuracy comparable to the SVD and NNMF-based approaches. Hence, we consider a more complex approximation that incorporates the biases of the individual users and items by including the terms (user average - user cluster average) and (item average - item cluster average) in addition to the co-cluster average. The approximate matrix $\hat{A}$ is given by

$$\hat{A}_{ij} = A^{COC}_{gh} + (A^R_i - A^{RC}_g) + (A^C_j - A^{CC}_h) \quad (3.1)$$

where $g = \rho(i)$, $h = \gamma(j)$ and A^R_i, A^C_j are the average ratings of user u_i and item p_j, and A^{COC}_{gh}, $A^R_g C$ and A^{CC}_h are the average ratings of the corresponding co-cluster, user-cluster and item-cluster respectively, i.e.,

$$A^{COC}_{gh} = \frac{\sum_{i'|\rho(i')=g} \sum_{j'|\gamma(j')=h} A_{i'j'}}{\sum_{i'|\rho(i')=g} \sum_{j'|\gamma(j')=h} W_{i'j'}}$$

$$A^{RC}_g = \frac{\sum_{i'|\rho(i')=g} \sum_{j'=1}^{n} A_{i'j'}}{\sum_{i'|\rho(i')=g} \sum_{j'=1}^{n} W_{i'j'}}$$

$$A^{CC}_h = \frac{\sum_{i'=1}^{m} \sum_{j'|\gamma(j')=h} A_{i'j'}}{\sum_{i'=1}^{m} \sum_{j'|\gamma(j')=h} W_{i'j'}}$$

$$A^R_i = \frac{\sum_{j'=1}^{n} A_{ij'}}{\sum_{j'=1}^{n} W_{ij'}} A^C_j = \frac{\sum_{i'=1}^{m} A_{i'j}}{\sum_{i'=1}^{m} W_{i'j}}$$

It can be shown [3] that the approximation matrix $\hat{A}$ is the least squares solution that preserves the user, item and co-cluster averages, or in other words, it is the best additive approximation to A given these summary statistics. Using $\hat{A}$, we can now pose the prediction of unknown ratings as a co-clustering problem where we seek to find the optimal user and item clustering (ρ, γ) such that the approximation error of $\hat{A}$ (which is a function of (ρ, γ)) with respect to the known ratings of A is minimized, i.e.,

$$\min_{(\rho,\gamma)} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(A_{ij} - \hat{A}_{ij})^2 \quad (3.2)$$

where W_{ij} ensures that only the known ratings contribute to the loss function. [3] presents an alternate minimization based algorithm that is guaranteed to provide a locally optimal solution for the co-clustering problem (3.2). The resulting co-clustering (ρ, γ) can then be used to compute the various averages required to estimate $\hat{A}$. In case of dynamic scenarios, the known ratings matrix A varies with time and hence, we require $\hat{A}$ to adapt to the changes in A in an efficient manner. Unlike SVD and NNMF, it is possible to devise a fast incremental update for (3.2) by keeping (ρ, γ) fixed since the availability of new ratings results in changes to only a few parameters in $\hat{A}$, i.e., the averages of the corresponding user, item and their respective clusters.

4 Collaborative Filtering via Co-clustering

In this section, we describe the three main components of our collaborative filtering approach based on co-clustering — (i) *static training*, which involves co-clustering the ratings matrix and computing the various summary statistics used for prediction, (ii) *prediction*, which consists of estimating an unknown rating from the various summary statistics, and (iii) *incremental training*, which involves updating the summary statistics based on incoming ratings.

4.1 Static Training

The main objective of the static training component is to compute all the parameters that are required for fast prediction of the unknown rating. In our co-clustering based approach, this essentially involves solving the problem (3.2) assuming the ratings matrix A is fixed and estimating the averages A^{COC}, A^{RC}, A^{CC}, A^R and A^C using the optimal co-clustering (ρ, γ). We perform this using the appropriate instantiation of the Bregman co-clustering algorithm presented in [3], which is guaranteed to provide a locally optimal solution. The key idea in this algorithm is to assume some initial co-clustering and then alternately optimize over the row (user) and column (item) clustering till convergence is achieved. Since the objective function in (3.2) can be written as the sum of contributions from the individual rows (or columns), the optimal row clustering (or column clustering) can be readily obtained by finding the appropriate cluster assignment for each row (or column). Algorithm 1 provides the details of the various steps.

Since the ratings matrix A is likely to be sparse, it is possible that there are no known ratings in certain co-clusters and in such cases, we replace the co-cluster average by the global average value. It is also important to note that the row and column assignment steps of Algorithm 1 can be implemented efficiently by pre-computing the invariant parts of the update cost functions. For example, the matrix A^{tmp1} defined by $A^{tmp1}_{ij} = A_{ij} - A^R_i - A^C_j$ can be computed at the very beginning. Now, for any $\{x_i\}_{i=1}^n$ and $\mu = \frac{1}{n}\sum_{i=1}^n x_i$, we note that $\sum_{i=1}^n (x_i - c)^2 = \sum_{i=1}^n (x_i - \mu)^2 + n(\mu - c)^2$ so that minimizing the L.H.S is equivalent to minimizing $(\mu - c)^2$. Using the above property, it can be shown that minimizing the row update cost function is equivalent to minimizing $A^{tmp2}_{i\gamma(j)} - A^{COC}_{g\gamma(j)} + A^{RC}_g$ where A^{tmp2} is given by $A^{tmp2}_{ih} = \frac{\sum_{j'|\gamma(j')=h} A^{tmp1}_{ij'}}{\sum_{j'|\gamma(j')=h} W_{ij'}} + A^{CC}_h$ and can be computed before the row cluster assignment step. The column cluster assignment step can be similarly simplified by computing a $k \times n$ matrix A^{tmp3} such that $A^{tmp3}_{gj} = \frac{\sum_{i'|\rho(i')=g} A^{tmp1}_{i'j}}{\sum_{i'|\rho(i')=g} W_{i'j}} + A^{RC}_g$. Further, since A is a sparse matrix and the objective function is based only on the known ratings (i.e., $W_{ij} \neq 0$), it is possible to speed up the co-clustering algorithm by storing only the relevant values in A^{tmp1}, A^{tmp2} and A^{tmp2}.

Assuming the above efficient implementation of the algorithm, the computational time for obtaining matrices A^{tmp1}, A^{tmp2} and A^{tmp3} is linear in the number of non-zeros in A. Each iteration of the algorithm further involves obtaining the optimal row and column assignments. For the row cluster assignment, the computational time required is $O(mkl)$ since each row of the $m \times l$ matrix A^{tmp2} needs to be compared with the k possible alternatives corresponding to the different row clusters. Similarly, the computational time for column assignments is $O(nkl)$. Therefore, assuming a constant number of iterations, the overall computation time is $O(W^{glob} + mkl + nkl)$ where W^{glob} is the number of non-zeros in A. Table 1 compares the computational time required for the various collaborative filtering approaches.

4.2 Prediction

In our approach, the prediction of unknown ratings is performed by suitably combining the various summary statistics. When the requested rating is for an existing user and item, the prediction is straightforward and determined by (3.1). When the user is new and the item is an existing one, the predicted value is the item average. Similarly, when the item is new and the user is an existing one, the predicted value the user average. When both the user and item are new, there is no specific information and we just return the global average of all the known ratings. Algorithm 2 shows the details of prediction process, which can be readily seen to be a constant time ($O(1)$) operation.

Algorithm 1 Static Training via Co-clustering

```
Input: Ratings Matrix $A$, Non-zeros matrix $W$, #row clusters
  $l$, #col. clusters $k$.
Output: Locally optimal co-clustering $(\rho, \gamma)$ and averages
  $A^{COC}$, $A^{RC}$, $A^{CC}$, $A^R$, and $A^C$.
Method:
  1. Randomly initialize $(\rho, \gamma)$.
  repeat
    2a. Compute averages $A^{COC}$, $A^{RC}$, $A^{CC}$, $A^R$ and $A^C$.
    2b. Update row cluster assignments
    $\rho(i) = \underset{1\le g\le k}{\text{argmin}} \sum_{j=1}^n W_{ij}(A_{ij} - A^{COC}_{g\gamma(j)} - A^R_i + A^{RC}_g - A^C_j + A^{CC}_{\gamma(j)})^2, \ 1 \le i \le m$
    2c. Update column cluster assignments
    $\gamma(j) = \underset{1\le h\le l}{\text{argmin}} \sum_{i=1}^m W_{ij}(A_{ij} - A^{COC}_{\rho(i)h} - A^R_i + A^{RC}_{\rho(i)} - A^C_j + A^{CC}_h)^2, \ 1 \le j \le n$
  until convergence
```

Table 1. Comparison of computational times of various CF algorithms. In case of SVD and NNMF, k is the rank of the approximation matrix and for correlation-based methods, k=#neighbors

Algorithm	Static Training	Prediction	Incremental Training
Co-clustering	$O(W^{glob} + mkl + nkl)$	$O(1)$	$O(1)$
SVD	$O(m^2n + n^3)^1$	$O(k)$	$O(W^{glob})$
NNMF	$O(W^{glob}k)$	$O(k)$	—
Correlation	$O(mW^{glob})$	$O(k)$	$O(W^{glob})$

Algorithm 2 Prediction

```
Input: $A^R, A^{RC}, A^C, A^{CC}, A^{CoC}, A^{glob}$, co-clustering $(\rho, \gamma)$
  user set $\mathcal{U}$, item set $\mathcal{P}$, user $u_i$, item $p_j$.
Output: Rating $r$
Method:
  Case $u_i \in \mathcal{U}$ and $p_j \in \mathcal{P}$: { old user-old item}
  $g \leftarrow \rho(i)$, $h \leftarrow \gamma(j)$
  $r \leftarrow {A_i}^R + {A_j}^C - {A_g}^{RC} - {A_h}^{CC} + {A_{gh}}^{CoC}$
  Case $u_i \in \mathcal{U}$ and $p_j \notin \mathcal{P}$: { old user-new item}
  $g \leftarrow \rho(i)$, $r \leftarrow {A_i}^R$
  Case $u_i \notin \mathcal{U}$ and $p_j \in \mathcal{P}$: {new user-old item}
  $h \leftarrow \gamma(j)$, $r \leftarrow {A_j}^C$
  Case $u_i \notin \mathcal{U}$ and $p_j \notin \mathcal{P}$: {new user-new item}
  $r \leftarrow A^{glob}$
```

4.3 Incremental Training

As mentioned earlier, we require our collaborative filtering approach to be applicable to dynamic settings where ratings are being continuously updated. Co-clustering algorithm (Algorithm 1) described earlier,

```
Algorithm 3 Incremental Training
Input: Matrix averages $A^R, A^{RC}, A^C, A^{CC}, A^{CoC}, A^{glob}$,
  Non-zero counts $W^R, W^{RC}, W^C, W^{CC}, W^{CoC}, W^{glob}$,
  co-clustering $(\rho, \gamma)$ user set $\mathcal{U}$, item set $\mathcal{P}$, user $u_i$, item $p_j$.
Output: Updated matrix averages and non-zero counts
Method:
  1. Identify user-item clusters
     Case $u_i \in \mathcal{U}$ and $p_j \in \mathcal{P}$: {old user-old item }
       $g \leftarrow \rho(i), h \leftarrow \gamma(j)$
     Case $u_i \in \mathcal{U}$ and $p_j \notin \mathcal{P}$: {old user-new item}
       $g \leftarrow \rho(i), h \leftarrow 0, \gamma(j) \leftarrow 0, \mathcal{P} \leftarrow \mathcal{P} \bigcup \{p_j\}$
     Case $u_i \notin \mathcal{U}$ and $p_j \in \mathcal{P}$: {new user-old item}
       $g \leftarrow 0, h \leftarrow \gamma(j), \rho(i) \leftarrow 0, \mathcal{U} \leftarrow \mathcal{U} \bigcup \{u_i\}$
     Case $u_i \notin \mathcal{U}$ and $p_j \notin \mathcal{P}$: {new user-new item}
       $g \leftarrow 0, h \leftarrow 0, \rho(i) \leftarrow 0, \gamma(j) \leftarrow 0$
     $\mathcal{U} \leftarrow \mathcal{U} \bigcup \{u_i\}, \mathcal{P} \leftarrow \mathcal{P} \bigcup \{p_j\}$
  2. Update the matrix averages
  3. Increment non-zero counts
```

though quite efficient, is primarily designed for a static setting where the users, items and the ratings are fixed.

However, completely ignoring the information in the new ratings could result in poor performance. To address this problem, we consider an incremental partial update mechanism. The key idea here is that since the prediction of the ratings depends only on the summary statistics (matrix averages), updating these statistics using the new ratings will incorporate most of the information in the new ratings. When the new rating corresponds to an existing user and item, it is straightforward to update the corresponding averages. However, when either the user or item is new, then there is no cluster assignment for this new entity. To resolve this problem, we temporarily assign the new entity to a global transitional cluster (indexed by 0) and update the corresponding averages. During the next run of the co-clustering algorithm, the users in the global user cluster and the items in the global item cluster are reassigned to one of regular user and item clusters. Algorithm 3 shows the details of the partial update procedure, which is also a constant time ($O(1)$) operation.

5 Scalable Collaborative Filtering System

In this section, we describe a real-time collaborative filtering system based on the co-clustering approach presented in Section 4. We also propose parallel versions of co-clustering, prediction and incremental training routine to further improve the scalability of our approach.

5.1 CF-System Description

From Section 4, we note that there are three main routines in our collaborative filtering approach. Of these, the prediction and incremental training routines can be performed extremely fast while the static training process is relatively slow. Since a user requires a fast response during the prediction and incremental training phases, it is essential to separate these from the static training process. Therefore, we design a simple collaborative filtering system (shown in Figure 1) assuming two parallel processors P_1 and P_2 (which could be threads in a single processor as well) where P_1 handles the prediction and incremental training and P_2 is responsible for the static training. Processor P_1, therefore, mainly interacts only with the summary statistics (say S) , i.e., reads the summary statistics during prediction and updates it during incremental training. During incremental training P_1, also updates the raw ratings. On the other hand, processor P_2 repeatedly performs co-clustering by reading the current ratings matrix A and updating S when done. To ensure consistency, the data objects A and S are stored in two parts — (a) stable values $A^{(t)}, S^{(t)}$ at the end of t^{th} co-clustering, and (b) increments $\Delta A^{(t)}, \Delta S^{(t)}$. At the end of each co-clustering run, the two parts are merged to obtain a new set of stable values.

Though reasonably efficient, the system shown in Figure 1 might not be adequate when the ratings matrix and the summary statistics are too large to be stored in the main memory of a single processor, for example, in the case of a news personalization application involving millions of users and articles. In such a situation, a natural solution is to use a distributed memory representation for the data objects so that each of the processors P_1 and P_2 are in fact clusters of processors. In order to maintain scalability of such a CF system, it is important to design efficient parallel versions of the various CF operations.

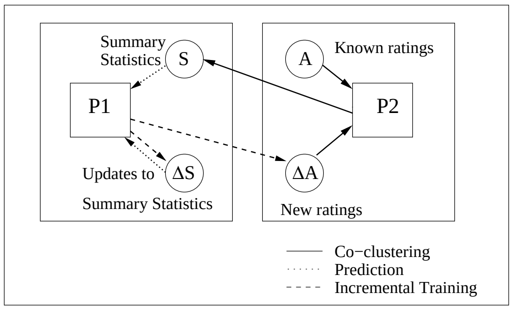

Figure 1. Collaborative Filtering System

5.2 Parallel Collaborative Filtering

The parallelization of P_1 and P_2 are independent of each other and depend on the distribution of S and A respectively. In fact, it might often be more practical to parallelize the operations of only one of the processors depending on the application.

Parallel Co-clustering (P_2). To obtain a parallel version of the co-clustering algorithm, we note that there are three main steps in each iteration of the co-clustering algorithm -(i) computing the matrix averages, (ii) obtaining the row cluster assignments, and (iii) obtaining the column cluster assignments. Further, given the matrix averages, the separability of the row and column update cost functions allows us to separately find the optimal assignment for each row and column. Using this inherent

Algorithm 4 Parallel Co-clustering

```
Input: Ratings Matrix $A$, Non-zeros matrix $W$, #row clusters
    $l$, #col. clusters $k$.
Output: Locally optimal co-clustering $(\rho, \gamma)$ and averages
    $A^{COC}$, $A^{RC}$, $A^{CC}$, $A^{R}$, and $A^{C}$.
Method:
  Processor p gets sub-matrices $A^{rp}$ ($m_p \times n$) & $A^{cp}$ ($m \times n_p$)
  1. Randomly initialize $(\rho^p, \gamma^p)$.
  repeat
    2a. Compute partial contributions to matrix averages and
    non-zero counts based on $A^{rp}$ and $A^{cp}$.
    2b. Accumulate the partial contributions to obtain overall
    matrix averages.
    2c. Update row assignments $\rho^p$ for all rows in $A^{rp}$.
    2d. Update column assignments $\gamma^p$ for all columns in $A^{cp}$.
  until convergence
  3. Concatenate the cluster maps $(\rho^p, \gamma^p)$ to obtain $(\rho, \gamma)$
```

data parallelism, we propose a parallel version of the co-clustering algorithm (Algorithm 4). The key idea is to divide the rows and columns among the processors so that the steps (ii) and (iii) can be completely performed in parallel. For step(i), the different processors compute their partial contributions which are then combined to obtain the overall matrix averages. To analyze the computational complexity, we assume that the matrix is dense and equally partitioned among N processors so that processor p owns $m_p = m/N$ rows and $n_p = n/N$ columns. Since each processor only owns a fraction of A, the computation of the partial contributions to the various cluster averages takes only $O(mn/N)$ operations while the accumulation of these contributions requires $O(Nkl)$ operations. The row and cluster assignment operation further require $O(m_p kl + n_p kl)$ operations. Hence, the overall computation time of the algorithm is $O(\frac{mn+mkl+nkl}{N} + Nkl)$ which corresponds to almost linear speedup assuming $N < \sqrt(m+n)$ and ignoring the communication costs. For a sparse matrix, one might have to use a more sophisticated partitioning scheme to ensure that the load is evenly balanced in order to obtain a good speedup.

Parallel Prediction and Incremental Training (P_1). When the summary statistics cannot be stored on a single processor, we can parallelize the prediction and incremental training routines by distributing the summary statistics based on the user and item clusters. The key idea is to separate out the tasks of determining the user and item clusters corresponding to a request (or submission) and performing the actual computation. When a rating is requested (or submitted) for a particular user u_i and item p_j, the root processor first determines the clusters to which these belong using the locally stored cluster mappings and directs the request to a slave processor that owns these clusters. In case of new users and items, the root processor maps them to the global transitional cluster and directs them to an appropriate slave processor. The slave processor then performs the actual computation involved in the prediction or incremental training.

6 Experimental Results

In this section, we provide empirical evidence showing that our co-clustering-based approach can provide high quality predictions on unknown ratings with much lower computational effort as compared to traditional collaborative filtering techniques. We also study the scalability benefits of parallelizing the co-clustering.

6.1 Datasets and Algorithms

We used two publicly available datasets (`MovieLens` and `BookCrossing` dataset) for our experiments. The `MovieLens` dataset (http://www.grouplens.org/data/) consists of 100,000 ratings(1-5) by 943 users on 1682 movies whereas the `BookCrossing` dataset [19] consists of 269392 explicit ratings(1-10) by 47034 users for 133438 books. We also created 10 subsets (`Movie1-Movie10`) of the `MovieLens` dataset each containing a fraction (10-100%) of the total ratings. We compared the performance of our co-clustering based approach with SVD [13], NNMF [10] and classic correlation-based collaborative filtering [12]. An incremental SVD-based approach [14] using a *folding in* technique was also implemented in order to evaluate the prediction accuracy in dynamic scenarios with changing ratings. The algorithms were all implemented in C++ and MPI (for parallel version). For implementing the SVD-based approach, we used a highly fine-tuned Fortran library LAPACK [1] to get reliable timing measurements.

6.2 Evaluation Methodology

To study the performance of our collaborative filtering approach, we focused on three evaluation metrics corresponding to prediction accuracy, static training time and prediction time. The prediction accuracy was measured using the mean absolute error (MAE), which is the average of the absolute values of the errors over all the predictions. The static training time was estimated in terms of the CPU time taken for the core training routines (viz. co-clustering and SVD) while the prediction time was estimated by averaging over the response time taken for all the predictions. All the timing measurements were obtained on a Linux Beowulf cluster consisting of 256 AMD Opteron Model 240 processors (1.4Ghz) on 128 compute nodes with 384GB RAM.

For evaluating the prediction accuracy, we created ten 80-20% random train-test splits of the datasets and averaged the results over the various splits. We considered two scenarios, —(i) static testing, where the known ratings do not change, and (ii) dynamic testing, where the ratings are updated incrementally. In the case of static testing, the prediction accuracy was obtained for all the four algorithms, whereas the dynamic testing accuracy was obtained only for the co-clustering and SVD-based methods since the other techniques do not have an incremental learning component. The training timing measurements were also restricted to the co-clustering and

Table 2. Mean absolute error using various CF algorithms in a static testing scenario

Data	SVD	NNMF	CORR	COCLUST
`Mov1`	0.83 ± 0.06	0.82 ± 0.04	0.84 ± 0.04	0.84 ± 0.06
`Mov2`	0.83 ± 0.03	0.85 ± 0.03	0.84 ± 0.02	0.82 ± 0.03
`Mov3`	0.82 ± 0.03	0.83 ± 0.03	0.85 ± 0.02	0.81 ± 0.03

Table 3. Mean absolute error using various CF algorithms in a dynamic scenario

Data	SVD	COCLUST
`Mov1`	0.796 ± 0.082	0.801 ± 0.067
`Mov2`	0.781 ± 0.069	0.784 ± 0.059
`Mov3`	0.788 ± 0.072	0.778 ± 0.061

SVD based approach since our implementation of the NNMF and the correlation-based CF techniques was not necessarily efficient and resulted in large training times.

The parameters of the different algorithms were varied based on the experiment. In all our experiments, the number of row and column clusters and the rank parameter of the SVD, NNMF algorithms was chosen to be identical. The number of neighbors in the correlation-based CF algorithm was set to a fixed number. Since co-clustering and the NNMF algorithms only provide locally optimal results, we performed multiple (5 runs) of these algorithms and picked the best solution.

6.3 Results and Discussion

We now present the experimental results showing how the prediction accuracy, the average prediction time and the training time depend on the algorithm, size of the datasets and the number of prediction parameters.

Prediction accuracy vs. Choice of CF-algorithm. Tables 2 and 3 show the mean absolute error obtained using the different collaborative filtering algorithms for static and dynamic scenarios respectively. In all the cases, the number of row and column clusters and the rank (for SVD,NNMF) was set to 3. From the tables, we note that all the algorithms including the co-clustering-based approach provide approximately the same accuracy for the static scenario. For the dynamic scenario, the SVD-based approach is the only other technique with an incremental training component and again, the accuracies obtained are comparable. It should be noted that the co-clustering approach required fewer parameters and less training time compared to all the other methods.

Prediction accuracy vs. #Prediction parameters. Figures 2 (a) and (b) show the variations of the mean absolute error with the number of row/column clusters (equal to rank k) and the number of prediction parameters respectively for the different algorithms on `Movie3` dataset. The number of prediction parameters corre-

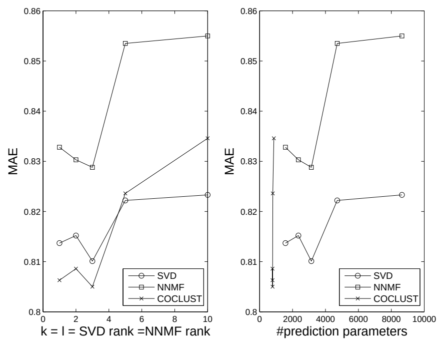

Figure 2. Variation of mean absolute error with #prediction parameters

Table 4. Average prediction time (in μs)

	SVD	NNMF	COCLUST
k=2	2.94	2.96	2.28
k=10	3.23	3.16	2.29

sponds to the extra storage required in addition to the raw ratings matrix. In case of the co-clustering approach, the prediction parameters are the various summary statistics ($(m+n+kl-k-l)$ values) whereas in case of SVD and NNMF, these are the matrix factors ($(m+n)(k+l)$ values). From the figure, we note that prediction accuracy is optimal at a certain k and then deteriorates with increasing k. The figure also clearly shows the difference in the number of prediction parameters required for the co-clustering and the other matrix factorization approaches.

Prediction time vs. Choice of CF-algorithm. Table 4 shows the average prediction time for the various CF algorithms on the `MovieLens` dataset. In all the cases, the prediction time is reasonably low, though the SVD, NNMF-based approaches show a small increase with increasing k.

Training time vs. Data set size. Figure 3 shows the variation of the static training time with the size of the dataset (#known ratings) for the co-clustering and SVD-based methods on the `MovieLens` dataset. From the figure, we observe that the co-clustering approach is much faster than the SVD-based method in spite of the fact that the SVD-based method uses an optimized library while the co-clustering algorithm did not. In our experiments, the NNMF and correlation based methods were found to be even slower than SVD, but we do not report the results due to the variations in the implementation. The results also indicate that the training time of co-clustering approach increases at a much lower rate than that of the SVD-based approach.

Training time vs. Number of processors. Figure 4 shows how the computational time of the co-clustering algorithm varies with the number of processors on four different datasets. From the figures, we note that there is reasonable (though sub-linear) speedup due to par-

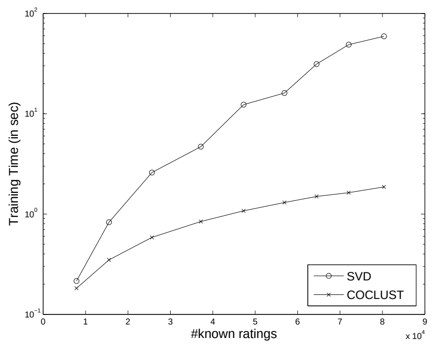

Figure 3. Variation of training time with size of the data set

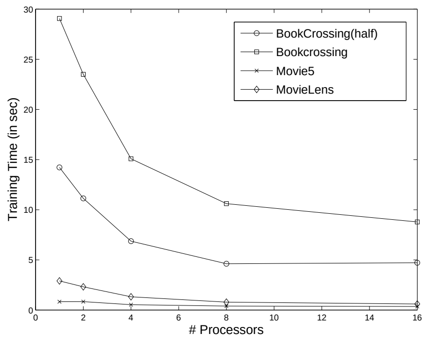

Figure 4. Variation of co-clustering time with number of processors

allelization. In particular, the trends on `Movie5` and `MovieLens` indicate that parallelization is more beneficial for larger datasets since the computational costs are much higher than the communication costs in that case.

7 Conclusion and Future Work

Recommender systems based on collaborative filtering are proving to be extremely useful for a number of online activities such as e-commerce. However, there remain important challenges that need to be addressed, specially with regard to scalability and deployment for dynamic scenarios where new users, items and ratings enter the system at a rapid rate. In this paper, we presented a new dynamic collaborative filtering approach based on simultaneous clustering of users and items. Empirical results indicate that our approach can provide high quality predictions at a much lower computational cost compared to traditional correlation and SVD-based approaches. We also proposed parallel versions of the various components of our collaborative filtering approach to make it scalable for large datasets containing millions of users and items. In future, we plan to extend our work in three main directions — (i) exploring a divisive hierarchical clustering setup that could be more suitable for real-time adaptation as well as for identifying the appropriate number of user/item clusters, (ii) adapting our collaborative filtering approach for streaming data items such as news articles by modifying the weight matrix W based on time progression, and (iii) investigating the suitability of co-clustering algorithms based on loss functions other than squared error.

References

[1] LAPACK- linear algebra package, 1999. http://www.netlib.org/lapack.

[2] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting hatches an egg: a new graph-theoretic approach to collaborative fi ltering. In *KDD '99*, pages 201–212, 1999.

[3] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *KDD*, pages 509–514, 2004.

[4] D. Billsus and M. J. Pazzani. Learning collaborative information fi lters. In *ICML*, pages 46–54, 1998.

[5] M. Brand. Fast online SVD revisions for lightweight recommender systems. In *SDM*, pages 37–48, 2003.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative fi ltering. In *UAI*, pages 43–52, 1998.

[7] Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.

[8] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative fi ltering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[10] T. Hofmann. Latent semantic models for collaborative fi ltering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[11] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of ACM Conf. on Computer Supported Cooperative Work*, pages 175–186, 1994.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *WebKDD Workshop.*, 2000.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental SVD-based algorithms for highly scaleable recommender systems. In *Proc. of the 5th Intl. Conf. on Computer and Information Technology*, 2002.

[15] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative fi ltering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[16] U. Shardanand and P. Maes. Social information fi ltering: Algorithms for automating "word of mouth". In *Proc. of ACM Conf. on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[17] N. Srebro and T. Jaakkola. Weighted low rank approximation. In *ICML*, pages 720–728, 2003.

[18] L. Ungar and D. Foster. Clustering methods for collaborative fi ltering. In *Workshop on Recommendation Systems*, 1998.

[19] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversifi cation,. In *WWW*, 2005.