Learning Bidirectional Similarity for Collaborative Filtering

Bin Cao[1], Jian-Tao Sun[2], Jianmin Wu[2], Qiang Yang[1], and Zheng Chen[2]

[1] The Hong Kong University of Science and Technology, Hong Kong
{caobin, qyang}@cs.ust.hk
[2] Microsoft Research Asia, 49 Zhichun Road, Beijing, China
{jtsun,i-jiwu,zhengc}@microsoft.com

Abstract. Memory-based collaborative filtering aims at predicting the utility of a certain item for a particular user based on the previous ratings from similar users and similar items. Previous studies in finding similar users and items are based on user-defined similarity metrics such as Pearson Correlation Coefficient or Vector Space Similarity which are not adaptive and optimized for different applications and datasets. Moreover, previous studies have treated the similarity function calculation between users and items separately. In this paper, we propose a novel *adaptive bidirectional similarity metric* for collaborative filtering. We automatically learn similarities between users and items simultaneously through matrix factorization. We show that our model naturally extends the memory based approaches. Theoretical analysis shows our model to be a novel generalization of the SVD model. We evaluate our method using three benchmark datasets, including MovieLens, EachMovie and Netflix, through which we show that our methods outperform many previous baselines.

1 Introduction

Personalized services are becoming increasingly indispensable nowadays ranging from providing searching result to product recommendation. Collaborative filtering aims at predicting the preference of items for a particular user based on the items previously rated by other users. Examples of successful applications of collaborative filtering include recommending products at Amazon.com[1], movies by Netflix[2], etc. Memory-based methods are a set of widely used approaches for collaborative filtering which are simple and effective [1]. They usually fall into two classes: user-based approaches [4,10] and item-based approaches [7,17]. To predict a rating for an item from a user, user-based methods find other similar users and leverage their ratings to the item for prediction, while item-based methods use the ratings to other similar items from the user instead.

[1] http://www.amazon.com
[2] http://www.netflix.com/

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 178–194, 2008.

Despite their success, memory-based methods suffer from several serious problems. First, missing data is a major problem in collaborative filtering, causing the so-called sparseness problem [24]. This is because there are usually millions of users and items in existence. But a single user can only rate a relatively small number of items. When the data are extremely sparse, it is difficult to find similar users or items accurately. Second, in memory based approaches, similar users and items are found by calculating a given similarity metric, including Pearson Correlation Coefficient (PCC) [16] and Vector Space Similarity (VSS) [4]. However, these metrics are not adaptive to the application domains and the data sets. Once given, they are not changeable. Third, the classical PCC and VSS have trouble in distinguishing different importance of items. To cope with these problems, many variations of similarity metrics, weighting approaches, combination measures, and rating normalization methods have been developed [9]. Although they can capture the correlation between users or items to a certain extent, for these adaptations to work, there is no consensus as to which choice of a technique is the most appropriate for a real world situation [9]. Finally, many previous studies in collaborative filtering consider the similarities between users and items separately. However, similarities between users and items in reality are interdependent and can be used to reinforce each other. Therefore, it would be more appropriate if the similarities between users and items be jointly learned automatically.

In this paper, we propose a novel model to learn both the item and user similarities together. Our model enables the similarity learning based collaborative filtering (SLCF). We show that the joint similarity learning can be formulated as a problem of matrix factorization with missing values. The learned similarities between users as well as items can be regarded as being influenced by some latent factors. Different from some previous latent factor models such as singular value decomposition (SVD) [25] and Aspect Model [11], our model provides a more flexible scheme that does not require the number of factors underling the user space and the item space to be the same. Theoretical analysis shows that our model corresponds to a novel generalization of the SVD model, thus allowing a number of nice theoretical properties to be inherited from SVD research. In addition, we provide algorithms for rating prediction with different strategies based on learned similarity. We evaluate our model using three widely used benchmark datasets, the MovieLens, EachMovie and Netflix data sets. Experiment results show that our method outperforms many of the well known baselines.

2 Related Work

In the past, many researchers have explored memory-based approaches to collaborative filtering. Many of them can be regarded as improving the definition of similarity metric [4,6,9,14]. A drawback of these methods is that these similarity metrics are not adaptive to different datasets or contain some parameters

needed to be tuned but not learned. Another set of related work consider how to utilize the user-based and item-based approaches together [14,21]. In [21], Wang et al. proposed a probabilistic fusion model to combine user-based method with item-based method. They found the fact that fusing all the ratings in the user-item matrix can help solve the data sparseness problem. However, they still estimate the user-based ratings and item-based ratings independently and omit the relationship between them. Ma et al. in [14] proposed a method to fill in the missing value first before prediction but it has the same drawback with [21]. One particular work which addressed learning similarity is in [12] where Jin et al. proposed an automatic weighting scheme for items. Their method aims at finding the optimal weights that can form a clustered distribution for user vectors in the item space by bringing similar users closer and dissimilar users far away. But they only considered the similarity weights for items, not users simultaneously.

Model based approaches do not predict ratings based on some ad-hoc heuristic rules, but rather, they are based on a *model* learned from the data using statistical and machine learning techniques. Viewed as a missing value prediction problem, collaborative filtering can also be solved through matrix factorization. SVD based approaches [3,20,25] can be regarded as latent factor models where the eigenvectors correspond to the latent factors. Users and items are mapped into a low dimensional space formed by the learned latent factors. Similar models also include [5,11]. A drawback of these models is that they all use the same latent factors to model users and items. An underlying assumption is that the numbers of latent factors that influence users and items are the same. Since a user may have diverse interests and an item may have multiple aspects, it is desirable to allow both items and users to be in a more flexible scheme. Si and Jin in [19] proposed a flexible mixture model for collaborative filtering. They are among the first to relax the restriction that users and items fall into the same classes. However, their probabilistic model regarded the ratings as discrete values. They also ignored the relation between ratings. As such, they did not consider scores of 3 and 2 to be closer to each other than scores of 5 and 1.

2.1 Memory-Based Collaborative Filtering

We review memory-based and SVD-based approaches for collaborative filtering (CF) in this and the next subsections. We construct a rating matrix R with rows representing users and columns representing movies. Since only part of the elements are known, we use X to denote the sparse matrix with elements known and use Y to denote the sparse matrix with elements we want to estimate. Both X and Y are subsets of rating matrix R. We define the problem of collaborative filtering as predicting the values in Y based on X.

User-based collaborative filtering predicts a target user u's interest in a test item m based on rating information from similar users.

$$r_{um} = \sum_{v \in C_u} s_{uv} r_{vm} \quad \text{for} \quad r_{um} \in Y \tag{1}$$

where r_{um} represents the rating for an item m from a user u and C_u is the set of nearest neighbors of the user u within which a user v has influence weight s_{uv} on u and s_{uv} can be calculated by normalizing Pearson Correlation Coefficient [16]. Hence $s_{uv} = PPC(u,v)/\sum_{w \in C_u} PPC(u,w)$, where

$$PCC(u,v) = \frac{\sum_{i \in R_u \cap R_v} (r_{ui} - \overline{r}_u) \cdot (r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in R_u \cap R_v} (r_{ui} - \overline{r}_u)^2} \cdot \sqrt{\sum_{i \in R_u \cap R_v} (r_{vi} - \overline{r}_v)^2}} \tag{2}$$

or Vector Space Similarity [4], so $s_{uv} = VSS(u,v)/\sum_{w \in C_u} VSS(u,w)$, where

$$VSS(u,v) = \frac{\sum_{i \in R_u \cap R_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in R_u \cap R_v} r_{ui}^2} \cdot \sqrt{\sum_{i \in R_u \cap R_v} r_{vi}^2}} \tag{3}$$

where R_u is the set of items rated by the user u.

Similar to user-based approach, we write item-based approaches as

$$r_{um} = \sum_{n \in C_m} s_{mn} r_{un} \quad \text{for} \quad r_{um} \in Y \tag{4}$$

where C_m is the set of nearest neighbors of the item m within which the item n has influence weight s_{mn} on m and s_{mn} can also be calculated using PCC or VSS as in the above equations.

2.2 SVD-Based Collaborative Filtering

Singular value decomposition (SVD)-based methods are also explored by many researchers for collaborative filtering [3,20,25]. SVD seeks a low-ranked matrix that minimizes the sum squared distance to the rating matrix R. Since most of the entries in R are missing, the sum-squared distance is minimized with respect to the partially *observed entries* of the rating matrix, which is X. So the loss function we optimize is

$$l = |I_X \odot (X - UV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2)$$

where $\odot$ stands for element-wise multiplication, $||\cdot||_F^2$ denotes the Frobenius norm, and I_X is the indicator function, with element $I_X(i,j)$ taking on value 1 if the user i rated the movie j, and 0 otherwise. U is a lower dimensional representation for users and V is a lower dimensional representation for items. The diagonal matrix Σ in traditional SVD is merged into U and V for simplicity. The last term is a regularization term which prevents the model from overfitting. *Unobserved entries* Y are then predicted by $Y = I_Y \odot (UV^T)$. The regularized SVD method has been shown to be successful in the competition of Netflix Prize [8,22].

Another adaptation of SVD-based method is using the EM algorithm to solve the missing value problem [25]. The basic idea is to iteratively estimate the

missing ratings and conduct SVD decomposition. However, since the matrix is no longer sparse in this approach, it quickly runs up against practical computational limits.

3 Learning Similarity Functions

We present our main contributions in this section. To begin with, we consider memory-based approaches in matrix form and extend it to one-directional similarity learning.

3.1 One-Directional Similarity Learning

Memory-based collaborative filtering methods are usually separated from model-based approaches and regarded as heuristic-based approaches [1]. In this paper we provide a novel way to model memory-based methods from matrix point of view.

Equation (1) can be written in a matrix form,

$$Y = \widehat{S}_1 X \tag{5}$$

where $\widehat{S}_1$ denotes the similarity matrix of row vectors corresponding to users with $\widehat{S}_1(u, v)$ defined by

$$\widehat{S}_1(u,v) = \begin{cases} s_{uv}, & v \in C_u, \quad (6) \\ 0, & \text{otherwise.} \quad (6') \end{cases}$$

Similar to the user-based approach, item-based methods can be represented in matrix form as

$$Y = X\widehat{S}_2 \tag{7}$$

where $\widehat{S}_2$ denotes the similarity matrix of the column vectors corresponding to items with $\widehat{S}_2(m, n)$ defined by

$$\widehat{S}_2(m,n) = \begin{cases} s_{mn}, & n \in C_m, \quad (8) \\ 0, & \text{otherwise.} \quad (8') \end{cases}$$

Noticing that X and Y are both subsets of the rating matrix R, Equations (5) and (7) can actually be seen as matrix reconstruction equations with respect to R. By replacing Y on the left side of the equation with R, we can obtain matrix factorization formulas for similarity matrix learning.

$$R = S_1 X \quad \text{and} \quad R = X S_2 \tag{9}$$

In the above formulas, the similarity matrices S_1 and S_2 are no longer predefined as in previous memory based approaches. Instead, they are the variables that can be learned from the data.

To reduce the number of parameters in a similarity matrix S, we can factorize S with $S = UV^T$. This means similarity matrices S_1 and S_2 can be non-symmetric since the influence between users may not be symmetric. Then we have a factorization problem with missing values,

$$R = UV^T X \tag{10}$$

If we ignore the missing values and replace R with X, this will lead to a new factorization problem

$$X = UV^T X \tag{11}$$

Matrix factorization in this form is also discussed in [23] where it is solved for document clustering.

If we assume the similarity matrices S_1 and S_2 are symmetric, we can reduce the number of parameters further and reformulate Equation (10) as

$$R = UU^T X \tag{12}$$

This is one-directional similarity learning model. In next subsection we extend it to bi-directional case.

3.2 Bi-Directional Similarity Learning

One-directional similarity learning considers users and items separately. In this section, we extend the learning problem to a bi-directional similarity learning problem that can learn the row and column similarities together. Recent studies [14,21] have found that the combination of user-based and item-based approaches can indeed boost the performance of collaborative filtering. However, these recently proposed methods still conduct user-based prediction and item-based prediction separately. In this section, we show how to integrate them together to take the advantage of both.

Based on previous subsection, a natural way to combine user-based and item-based approach can be stated as

$$r_{um} = \sum_{v,n} s_{uv} s_{mn} r_{vn} \quad \text{for} \quad r_{um} \in Y \tag{13}$$

In this formula, we extend the neighborhood to all users and all items. This indicates that all ratings are interconnected: the prediction for a target user and item can benefit from ratings of other users and items, and vice versa.

The above equation can be re-written in matrix form

$$Y = S_1 X S_2 \tag{14}$$

where S_1 and S_2 are also variables we need to learn. S_1 represents the row (user) similarity matrix and S_2 represents the column (item) similarity matrix. Similar

to one-directional similarity learning, we have a similarity learning problem in matrix factorization form.

$$R = S_1 X S_2 \tag{15}$$

With the assumption that the similarity matrices S_1 and S_2 are symmetric, the problem can be converted to

$$R = UU^T XVV^T \tag{16}$$

where U is a rank-K_U matrix and V is a rank-K_V matrix with K_U denoting the number of latent factors for users and K_V be the number of latent factors for items.

We can also extend the model to nonsymmetric similarity matrix, but in that case we have more parameters to learn. Symmetric assumption can significantly decrease the number of variables we need to learn. Another advantage of using this trick is that it guarantees the similarity matrix to be positive semi-definite naturally. Therefore, we still follow the symmetric assumption in this paper.

3.3 Algorithms for Bi-directional Similarity Learning

Now the loss function we are going to minimize is

$$l = ||I_X \odot (R - UU^T XVV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{17}$$

Since $I_X \odot R = I_X \odot X$, l can be converted to

$$l = ||I_X \odot (X - UU^T XVV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2)$$

The last term in l is a regularization term which prevents the model from overfitting. Let $E = I_X \odot (X - UU^T XVV^T)$, then the loss function is simplified by

$$l = ||E||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{18}$$

We use gradient approaches to solve the minimization problem. We have the derivation of U and V in matrix form:

$$\frac{\partial l}{\partial U} = -2(EVV^T X^T U + XVV^T E^T U) + 2\alpha U \tag{20}$$

$$\frac{\partial l}{\partial V} = -2(E^T UU^T XV + X^T UU^T EV) + 2\alpha V \tag{20'}$$

There are a lot of gradient based algorithms which have been developed for optimization problems such as conjugate gradient [15] and SMD [18]. In this paper we use adaptive gain gradient decedent algorithm [2] to minimize the loss function. The algorithm is described in Algorithm 1. The advantage of adaptive gain gradient decedent algorithm includes easy implementation and fast convergence speed.

Algorithm 1. Bi-directional Similarity Learning using Adaptive Gain

Input: training data X, parameters μ, K_U, K_V and T
Output: U and V
Initialization: Random initialize U and V
FOR $t = 1$ **TO** T**:**
Update U**:** $U^{(t+1)} = U^{(t)} - \eta_U^{(t)} \odot \frac{\partial l}{\partial U}^{(t)}$
Update V**:** $V^{(t+1)} = V^{(t)} - \eta_V^{(t)} \odot \frac{\partial l}{\partial V}^{(t)}$
Update η_U**:**
$\eta_U^{(t)} = \eta_U^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_U^{(t)} \odot \frac{\partial l}{\partial U}^{(t-1)} \odot \frac{\partial l}{\partial U}^{(t)})$
Update η_V**:**
$\eta_V^{(t)} = \eta_V^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_V^{(t)} \odot \frac{\partial l}{\partial V}^{(t-1)} \odot \frac{\partial l}{\partial V}^{(t)})$

Another point we should notice is that although the similarity matrices S_1 and S_2 are large and dense, we can avoid computing them in the algorithm by carefully choosing the order of matrices multiplication.

3.4 Relation to SVD

In this section, we discuss the relation between our model and SVD model.

Theorem 1. *If we disregard the missing data and require that the ranks of U and V are the same, SVD is the solution to $X = UU^TXVV^T$.*

Proof. Suppose that $X = U\Sigma V^T$. By plugging it into UU^TXVV^T, we obtain $UU^TXVV^T = UU^TU\Sigma V^TVV^T = U\Sigma V^T = X$.

The equivalence of our model and SVD models can be established under the condition that there are no missing values and U and V have equal ranks. However, when there are missing values, the two models are not equivalent anymore even when we have $K_U = K_V = K$. We can see this point in the experiment part again.

Another difference between our model and SVD is seen from the rank of approximation matrix. SVD seeks the optimal rank-K approximation to the original matrix. But in our problem, we are not explicitly given rank restriction of the reconstructed matrix. The rank of reconstructed matrix is determined by the ranks of S_1, S_2 and X itself.

From the dimension-reduction point of view, SVD seeks a K dimensional space for row vectors and column vectors. However, in our model, we look for two different ranked spaces for row vectors and column vectors. Therefore, our model can also be regarded as bi-dimension-reduction-based method for row vectors and column vectors with different dimensions. We also can find the relation between the two spaces as two basis sets satisfy the following equation

$$U \cdot B_1 = B_2 \cdot V \tag{21}$$

where the users' basis $B_1 = U^TXVV^T$ and the items' basis $B_2 = UU^TXV$.

4 Rating Prediction Based on Bidirectional Similarity Learning

Different strategies can be used for collaborative filtering based on our learned similarity. In this section, we discuss three types of similarity learning based collaborative filtering strategies.

4.1 Matrix Reconstruction Strategy

Model-based approaches keep the user profiles in a more compressed data structure than memory based methods. The prediction for a user's interests is based on the user's profile that is learned during a training process. In our model, the user u's profile corresponds to row u in matrix U denoted by U_u and the item i's profile corresponds to column i in V, i.e. V_i^T. With our learned model, we predict a rating to the item i by the user u,

$$r_{ui} = \sum_{v,j} s_{vu} s_{ij} r_{vj} = U_u U^T X V V_i^T \quad \text{for} \quad r_{ui} \in Y$$

This can be done when both u and i show up in the training data X. We refer to this prediction strategy as matrix reconstruction strategy for SLCF (R-SLCF).

Matrix reconstruction strategy for collaborative filtering has the new user and new item problem. It can only predict the rating for existing users and items during training process. A naive solution to this problem is to retrain the whole new dataset and then make prediction for the new users and items. This procedure is clearly too time-consuming and often infeasible. In the next sub-section, we will use another strategy to solve this problem.

4.2 Projection Strategy

In this section, we discuss projection based strategy P-SLCF in our new framework which can bring new users and items into the model without retraining on the whole dataset. The key issue is how to introduce new users and items into the previous model and predict ratings for these new users and items based on previous models.

Suppose that there are some new users who arrive with new rating information $\widehat{Y}$ and $\widehat{Y}$ is to be included into the previous user rating matrix X. Then we have a new rating matrix with $X' = \begin{pmatrix} X \\ \widehat{Y} \end{pmatrix}$. Let U_Y be a representation of new users. Hence we have

$$\widehat{Y} = U_{\widehat{Y}} \cdot U^T X V V^T = U_{\widehat{Y}} \cdot B_1 \tag{22}$$

By solving the above linear equation, we find $U_{\widehat{Y}}$ with

$$U_{\widehat{Y}} = \widehat{Y} \cdot ((I_{\widehat{Y}} \odot B_1) \cdot (I_{\widehat{Y}} \odot B_1)^T + \lambda \mathbb{I})^{-1} \tag{23}$$

where $\mathbb{I}$ is identity matrix. We can regard the user as being projected to a lower dimensional space spanned by the matrix B_1. Then, all new users are projected

into this space. The last term $\lambda\mathbb{I}$ is introduced to guarantee that the inverse operation is more stable [13].

Similar to adding new users, we can consider the new items as being projected to a lower dimensional space spanned by B_2. Suppose that there are some new items that arrive with new rating information $\widehat{Y}$ and $\widehat{Y}$ is included into the previous user rating matrix X to give $X' = (X, \widehat{Y})$. We can update $V_{\widehat{Y}}$ by

$$\widehat{Y} = UU^T XV \cdot V_{\widehat{Y}}^T = B_2 \cdot V_{\widehat{Y}}^T \tag{24}$$

$$U_{\widehat{Y}} = \widehat{Y} \cdot ((I_{\widehat{Y}} \odot B_2) \cdot (I_{\widehat{Y}} \odot B_2)^T + \lambda\mathbb{I})^{-1} \tag{25}$$

Then similar to R-SLCF, we can predict the rating by

$$R_{\widehat{Y}} = U_{\widehat{Y}} U^T XVV_{\widehat{Y}}^T$$

Although we need to calculate inverse of matrices in projection based strategy, but since the matrices are of rather small scale and can be computed efficiently.

4.3 Improved Memory-Based Strategy

Memory-based methods can also be adapted to use our learned similarity. The idea is to use the learned similarity matrices S_1 and S_2 to find the nearest neighbors. Then we can use the memory based methods for prediction. We refer to this strategy as M-SLCF. This strategy is especially helpful for comparing our learned similarity with the user-defined similarity such as PCC. We show the results of comparison in Section 5.3.

5 Experiment

In this section, we will introduce data sets, evaluation metric and experiment results of our similarity learning-based collaborative filtering. In Section 5.3, M-SLCF is used and in Section 5.4, P-SLCF is used for comparison purpose. In other parts, R-SLCF is used for experiments.

5.1 Datasets

Three benchmark datasets are used in our experiments.

- MovieLens[3] is a widely used movie recommendation dataset. It contains 100,000 ratings with scale 1-5. The ratings are given by 943 users on 1,682 movies. The public dataset only contains users who have at least 20 ratings.
- EachMovie[4] is another popular used dataset for collaborative filtering. It contains 2,811,983 ratings from 72,916 users on 1,628 movies with scale 1-6.

[3] http://www.grouplens.org/

[4] http://www.cs.cmu.edu/ lebanon/IR-lab.htm

Table 1. Optimal K_V Given K_U

K_U	5	6	7	8	9	10	11	12	13	14	15
Opt K_V	14	14	14	14	12	10	8	8	8	6	5
MAE	0.7611	0.7606	0.7606	0.7607	0.7604	0.7606	0.7605	0.7603	0.7606	0.7607	0.7608

Table 2. Optimal K_U Given K_V

K_V	5	6	7	8	9	10	11	12	13	14	15
Opt K_U	13	13	12	12	12	11	9	9	9	7	6
MAE	0.7608	0.7607	0.7606	0.7603	0.7606	0.7606	0.7606	0.7604	0.7607	0.7606	0.7610

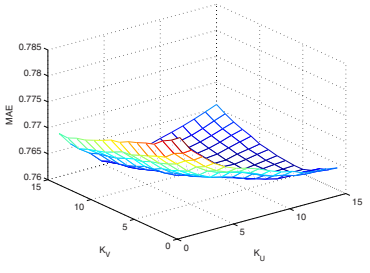

Fig. 1. MAE surface of R-SLCF on MovieLens dataset. Numbers of user factor (K_U) and item factor (K_V) are varied simultaneously.

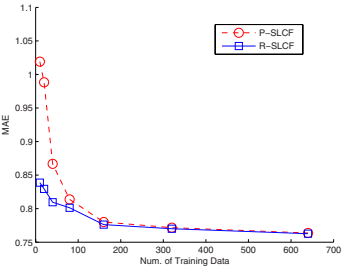

Fig. 2. Comparison of P-SLCF and R-SLCF. Evaluated by MAE on Movielens with 200 user as test data, $K_U = K_V = 10$ for SLCF.

– Netflix[5] is a pubic dataset used in Netflix Prize competition. It contains ratings from 480,000 users on nearly 18,000 movies with scale 1-5. In this paper, we use a subset of 367,348 ratings from 5,000 users and 2,000 movies for our experiments.

5.2 Evaluation Metrics

In this paper, we use Mean Absolute Error (MAE) for experiment evaluation.

$$MAE = \frac{\sum_{u,m} |r_{um} - \widehat{r}_{um}|}{N}$$

where r_{um} denotes the rating of the user u for the item m, and $\widehat{r}_{um}$ denotes the predicted rating for the item m of the user u. The denominator N is the number of tested ratings. Smaller MAE score corresponds with better prediction.

[5] http://www.netflixprize.com

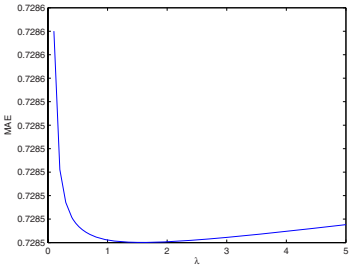

Fig. 3. Influence of parameter λ to the performance of P-SLCF

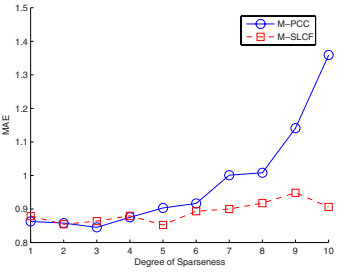

Fig. 4. Comparison of PCC and SLCF on similarity with different degree of sparseness. Evaluated using MAE on EachMovie with 50 nearest neighbors

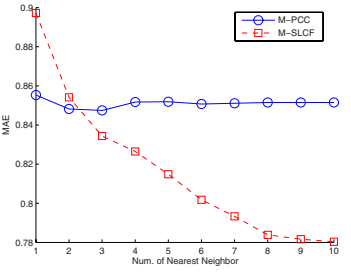

Fig. 5. Comparison of PCC and SLCF on similarity with different number of nearest neighbors. Evaluaed using MAE on EachMovie with sparseness degree 5.

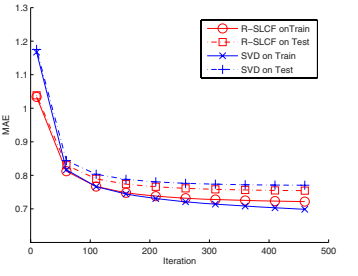

Fig. 6. Converge curves of R-SLCF and regularized SVD. Evaluated by MAE on Movielens with parameter $K = 10$ for regularized SVD, $K_U = K_V = 10$ for R-SLCF.

5.3 Empirical Study of our Approach

Impact of K_U and K_V. Two important parameters of our SLCF methods are the user similarity matrix rank K_U and the item similarity matrix rank K_V. In this experiment, we run experiments on MovieLens dataset to study the impact of K_U and K_V. Figure (1) shows the three dimensional MAE surface with K_U and K_V being changed simultaneously. We find that the best prediction result is achieved when K_U and K_V are neither too small nor too large. Table (1) shows the best K_V for given K_U and Table (2) shows the best K_U for given K_V. An interesting observation is that most of the best prediction results are achieved when $K_U + K_V \approx 20$. This means that the inherent information conveyed by latent user factors and item factors are complementary to each other. When fewer user factors are available, more item factors are required to characterize the inherent structure of rating matrix, and vice versa. From the MAE surface of Figure (1), the best result is obtained when both user and item factors are

considered ($K_U = 12$, $K_V = 8$). This verifies our motivation that user and item spaces should be modeled with different numbers of factors. Another parameter in our model is α which controls the balance between prediction error on training data and model complexity. After testing on different values, we use $\alpha = 0.0001$ in our experiments.

The Difference of R-SLCF and P-SLCF. Since it is costly to retrain the model when new users or items come, we provide the P-SLCF algorithm in Section 4.2. In this experiment, we compare the accuracy of prediction by R-SLCF and P-SLCF. Figure (2) shows the comparison results on MovieLens. In this experiment, we use 200 users as testing data. When training users are very few, P-SLCF is not as good as R-SLCF. But as the number of training users increases, the performances of P-SLCF and R-SLCF become very close.

An important parameter in P-SLCF is λ. Figure (3) shows the influence of λ to the prediction accuracy. After testing different values of λ, we find that $\lambda = 1$ to be a good choice which we use in our experiments.

Impact of Data Sparseness. In this sub-section, we show experiments on the impact of data sparseness on similarity learning using M-SLCF. For comparison purpose, we also use the predefined similarity PCC (Equation (2)) for selecting neighbors which we refer to as M-PCC. In both cases, Equation (1) with equal weights for neighbors is used for making predictions.

We first filter the EachMovie dataset by keeping the users who have rated different number of movies (from less than 50 to less than 5 in this experiment). In this way, we construct datasets with different degree of sparseness. We use user based method with neighbors found by SLCF and compare it with PCC. When the data are not that sparse, PCC can do good job in finding nearest neighbors. However, when the degree of sparseness increases, it does not work anymore. In Figure (4), we can clearly see that SLCF is able to find more accurate neighbors with the degree of sparseness increased. Figure (5) verifies our conclusion from the other side. It shows how SLCF and PCC perform with different number of nearest neighbors. We can see that PCC is good at finding the most similar users but SLCF has the advantage of finding the potentially similar users. That is, we can improve the recall of finding similar users. Therefore, when more nearest neighbors are used, our model performs much better.

5.4 Comparison with Other Approaches

The baselines we use include user-based method using PCC, item-based method using PCC and regularized SVD method. We also compare our method with another recent proposed state-of-the-art method [21] which also fusions the similarities of users as well as items. Although we also conduct experiments on Netflix dataset, our results are not comparable with the top results on the leaderboard since they are hybrid methods. We should notice regularized SVD, which is one of the best algorithms in the Netflix Prize competition [8], is also included in our baselines.

Table 3. MAE comparison of R-SLCF with SVD for different K. For R-SLCF1 we require $K_U = K_V = K$. For R-SLCF2 we require $K_U + K_V = 2K$.

Dataset	K	SVD	R-SLCF1	R-SLCF2
MovieLens	K=5	0.7665	**0.7534**	**0.7534**
	K=10	0.7676	0.7517	**0.7516**
	K=15	0.7785	0.7533	**0.7523**
	K=20	0.7906	0.7554	**0.7532**
EachMovie	K=5	0.8023	0.7902	**0.7901**
	K=10	0.8272	0.7855	**0.7845**
	K=15	0.8317	0.7920	**0.7912**
	K=20	0.8127	0.7932	**0.7920**
Netflix	K=5	0.7557	0.7505	**0.7501**
	K=10	0.7640	0.7490	**0.7480**
	K=15	0.7737	0.7498	**0.7498**
	K=20	0.7835	0.7571	**0.7569**

Table 4. MAE comparison of R-SLCF with memory-based method and item-based method. $N = 30$ means only users with ratings no larger than 30 are included.

Dataset	#Rating	I-based	U-based	R-SLCF
MovieLens	N=30	1.0936	0.8785	**0.8418**
	N=40	0.9587	0.8527	**0.8113**
	N=50	0.9144	0.8451	**0.8104**
	N=60	0.8648	0.8239	**0.8056**
EachMovie	N=30	1.7238	0.9919	**0.9347**
	N=40	1.6437	0.9908	**0.9297**
	N=50	1.7792	0.9836	**0.9338**
	N=60	1.6656	0.9886	**0.9327**
Netflix	N=30	0.9568	0.8804	**0.7974**
	N=40	0.8647	0.8390	**0.7782**
	N=50	0.8293	0.8114	**0.7672**
	N=60	0.7934	0.7774	**0.7439**

Table 5. Compare with results of SF on MovieLens

Num. of Training Users	100			200			300		
Num. of Ratings Given	5	10	20	5	10	20	5	10	20
P-SLCF	**0.838**	**0.770**	**0.771**	**0.799**	**0.768**	**0.763**	**0.787**	**0.753**	**0.739**
SF	0.847	0.774	0.792	0.827	0.773	0.783	0.804	0.761	0.769

Comparison with SVD-Based Approaches. Since our model is similar to SVD, in this section, we carefully compare our model with the regularized SVD model we introduced in Section 2.2 in different aspects. Figure (6) shows the convergence curves of our approach compared with regularized SVD. In this experiment, we use the same optimization algorithm (adaptive gain) with the same initial point[6] for U and V to run the algorithms and tune the best step length for each algorithm. We can see our approach converges faster than regularized SVD and finds better solution. It is also worthy to notice that in the last several iterations regularized SVD has smaller MAE on training data but larger MAE on test data when compared with R-SLCF. This indicates regularized SVD is more likely to be overfitting than our model. This may be due to that regularized SVD requires a strict rank-K approximation but we do not.

Table (3) shows a performance comparison of our model and regularized SVD model with various Ks. In this experiment, R-SLCF uses the same number of variables with regularized SVD for the fair of comparison. We can see our method clearly outperforms regularized SVD model. This experiment also indicates that

[6] Although the initial points are the same, the initial performance can be different.

when there are missing values our model is different from regularized SVD even $K_U = K_V$.

Comparison with Memory-Based Approaches. We compare our method with user-based(U-based) and item-based(I-based) approaches with results shown in Table (4). The experiment is carried out with different sparseness condition with $N = 30$ meaning only users who have ratings less than or equal to 30 are used. From this table we can see that our method clearly outperforms the baselines.

We also compare our method with another stat-of-the-arts algorithm Similarity Fusion (SF) [21] which also utilizes both user side and item side information. The difference between our approach and SF is that the similarities used in our algorithm is automatically learned rather than defined heuristically. To compare with their algorithm, we followed the exactly same experiment settings in the paper. Then, for the performance of their method, we quote their results from their publication. We can see that our approach outperforms SF significantly.

6 Conclusion and Future Work

We proposed a novel model learning user and item similarities simultaneously for collaborative filtering. We showed that our model can be regarded as a generalization of SVD model. We developed an efficient learning algorithm as well as three prediction strategies. The experiments showed our method could outperform baselines including memory-based approaches and SVD.

For future work, we plan to develop more efficient algorithms to learn our model in larger scale datasets. We also plan to relax the symmetry assumption. Although it brings more variables to learn, it is a more reasonable assumption. Although focused on collaborative filtering in this paper, our model is very general for sparse data which has matrix form. Therefore, we plan to apply our model to other kinds of data sets and tasks such as document clustering.

Acknowledgments

Bin Cao and Qiang Yang are supported by a grant from MSRA (MRA07/08. EG01). We thank the anonymous reviewers for their useful comments.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE TKDE 17(6), 734–749 (2005)
2. Almeida, L.B., Langlois, T., Amaral, J.D., Plakhov, A.: Parameter adaptation in stochastic optimization. On-line learning in neural networks, 111–134 (1998)

3. Brand, M.: Fast online svd revisions for lightweight recommender systems. In: Proc. of SIAM ICDM (2003)
4. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14^{th} Conf. on UAI, pp. 43–52 (1998)
5. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proc. of the 25th SIGIR, pp. 238–245. ACM, New York (2002)
6. Delgado, J.: Memory-based weightedmajority prediction for recommender systems. In: ACM SIGIR 1999 Workshop on Recommender Systems (1999)
7. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM TOIS 22(1), 143–177 (2004)
8. Funk, S.: Netflix update: Try this at home (December 2006), `http://sifter.org/~simon/journal/20061211.html`
9. Herlocker, J., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf. Retr. 5(4), 287–310 (2002)
10. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proc. of the 22nd SIGIR, pp. 230–237. ACM, New York (1999)
11. Hofmann, T.: Latent semantic models for collaborative filtering. ACM TOIS 22(1), 89–115 (2004)
12. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proc. of the 27th Annual International ACM SIGIR
13. Kirsch, A.: An introduction to the mathematical theory of inverse problems. Springer, New York (1996)
14. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: Proc. of the 30th SIGIR, pp. 39–46. ACM, New York (2007)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C, 2nd edn. The art of scientific computing. Cambridge University Press, Cambridge (1992)
16. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proc. of ACM 1994 Conf. on CSCW (1994)
17. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th international Conf. on WWW, pp. 285–295 (2001)
18. Schraudolph, N.N.: Local gain adaptation in stochastic gradient descent. Technical Report IDSIA-09-99, 8 (1999)
19. Si, L., Jin, R.: A flexible mixture model for collaborative filtering. In: Proc. of the Twentieth ICML (2003)
20. Vozalis, M.G., Margaritis, K.G.: Using SVD and demographic data for the enhancement of generalized collaborative filtering. Inf. Sci 177(15) (2007)
21. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proc. of the 29th SIGIR, pp. 501–508. ACM, New York (2006)
22. Wu, M.: Collaborative filtering via ensembles of matrix factorizations. In: Proc. of KDD Cup and Workshop (2007)
23. Xu, W., Gong, Y.: Document clustering by concept factorization. In: Proc. of the 27th SIGIR, pp. 202–209. ACM, New York (2004)

24. Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proc. of the 28th SIGIR, pp. 114–121. ACM, New York (2005)
25. Zhang, S., Wang, W., Ford, J., Makedon, F., Pearlman, J.: Using singular value decomposition approximation for collaborative filtering. In: Proc. of the Seventh IEEE CEC, pp. 257–264 (2005)